EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

SAĞLIK KAYITLARININ İZLENMESİNDE AKILLI KART KULLANIMI

Geylani KARDAŞ

Uluslararası Bilgisayar Anabilim Dalı
Bilim Dalı Kodu : 619.02.04
Sunuş Tarihi : 06.08.2003

Tez Danışmanı : Prof. Dr. Turhan Tunalı

BORNOVA - İZMİR

Geylani KARDAŞ tarafından YÜKSEK LİSANS TEZİ olarak sunulan "Sağlık Kayıtlarının İzlenmesinde Akıllı Kart Kullanımı" başlıklı bu çalışma E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 06.08.2003 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:	İmza:
Jüri Başkanı :	.....
Raportör Üye:	.....
Üye :	.....

ÖZET

SAĞLIK KAYITLARININ İZLENMESİNDE AKILLI KART KULLANIMI

KARDAŞ, Geylani

Yüksek Lisans Tezi, Uluslararası Bilgisayar Enstitüsü

Tez Yöneticisi: Prof. Dr. Turhan TUNALI

Ağustos 2003, 227 sayfa

Akıllı kartlar, içerisinde bilgi depolayan ve bu bilgiyi işleyen, taşınabilir birer entegre olarak bilgi sistemlerinde kullanılmaktadır. Özellikle sağladıkları güvenlik ve hızlı bilgi erişimi olanakları sayesinde bu kartların bir çok sektörde olduğu gibi sağlık sektöründe de bilgi taşıma medyası olarak yer alması son dönemde oldukça popüler hale gelmiştir.

Bu tez çalışmasında; akıllı kartlara dayalı bir sağlık sistemi tasarlanmış ve gerçekleştirilmiştir. Sistem bir hastane bünyesinde; bilgi akışında akıllı kartların kullanılmasını ve yine tez kapsamında hazırlanan dağıtık bir protokol üzerinden bilgi alışverişinin sağlanmasını içermektedir. Sistemde, hastaların ve doktorların sahip olacağı iki ayrı akıllı karta ait kart yazılımları hazırlanmıştır. Hastalara ait akıllı kartlar üzerinde hasta kişisel bilgileri dışında genel sağlık bilgileri de yer almakta; doktorlar da kendilerini sisteme tanıtan akıllı kartları sayesinde hasta kartlarındaki bilgileri kullanarak işlem yapabilmektedirler. Sistem dağıtık nesne protokolü üzerinde, istemciler ve veritabanı sunucuları arasındaki bilgi alışverişi sırasında kartlarda yer alan şifreleme ve imzalama anahtarları kullanılmaktadır. Sistem, nesneye dayalı mimari ve tasarım desenleri kullanılarak Java platformunda hazırlanmıştır.

Anahtar sözcükler: Akıllı kart, sağlık bilgi sistemleri, dağıtık nesne protokolü

ABSTRACT

USE OF SMART CARDS IN HEALTH RECORDS

KARDAŞ, Geylani

MSc. in International Computer Institute

Supervisor: Prof. Dr. Turhan TUNALI

August 2003, 227 pages

Smart cards are used in information technologies as portable integrated devices with data storage and data processing capabilities. As in other fields, smart card use in health systems became popular due to their increased capacity and performance. Their efficient use with easy and fast data access facilities leads to implementation particularly widespread in security systems.

In this thesis, a smart card based healthcare system is designed and implemented. The system includes smart card usage in data flow and provides data communication via a distributed protocol which is also designed in this thesis. Two smart card software modules are implemented which run on patient and healthcare professional smart cards respectively. In addition to personal information, general health information about the patient is also loaded to patient smart card. Health care providers use their own smart cards to be authenticated on system and to access data on patient cards. Encryption keys and digital signature keys stored on the system smart cards are used in secure and authenticated data communication between clients and database servers over system's distributed object protocol. System is developed on Java platform by using object oriented architecture and design patterns.

Keywords: Smart card, healthcare information system, distributed object protocol

TEŞEKKÜR

Öncelikle bu tez konusu üzerinde bana çalışma imkanı sunan tez danışmanım Prof. Dr. Turhan TUNALI'ya çalışma süresince deneyimi, bilgisi ve önerileriyle araştırma ve geliştirmeyi yönlendirmesi ve sağladığı kaynaklarla destek olmasından dolayı teşekkürü bir borç bilirim.

Çalışma sırasında ihtiyaç duyduğum tıbbi veri örneklerini sağlayan ve çalışma hakkında değerli görüşlerini esirgemeyen Doç. Dr. Tayfun DALBASTI 'ya (Ege Üniversitesi Tıp Fakültesi Nöroşirürji ABD), yine çalışma sırasında bilgi ve görüşlerinden yararlandığım ve yakın ilgisini esirgemeyen Dr. Şule YILDIRIM'a (Norveç Bilim ve Teknoloji Üniversitesi) ve manevi desteklerini esirgemeyen aileme ve arkadaşlarıma teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

ÇİZELGELER DİZİNİ

Çizelge Sayfa

KISALTMALAR

AID : Application Identifier (Uygulama Tanımlayıcısı)

AKSS : Akıllı Kart Sağlık Sistemi

APDU : Application Protocol Data Unit (Uygulama Protokolü Veri Birimi)

API : Application Programming Interface (Uygulama Programlama Ara yüzü)

CAD : Card Acceptance Device (Kart Kabul Cihazı)

CAP : Converted Applet (Çevrilmiş Küçük Uygulama)

CLA : Class of instruction (Komut Sınıfı)

CORBA : Common Object Request Broker Architecture (Genel Nesne İstek Aracı Mimarisi)

DBMS : Database Management System (Veritabanı Yönetim Sistemi)

DES : Digital Encryption Standard (Sayısal Şifreleme Standartı)

DF : Dedicated File (Atanmış Dosya)

DSA : Digital Signature Algorithm (Sayısal İmzalama Algoritması)

EDI : Electronic Document Interchange (Elektronik Doküman Değişimi)

EEPROM : Electrical Erasable Programmable Read Only Memory (Elektriksel Silinebilir Programlanabilir Salt Okunur Bellek)

EF : Elementary File (Temel Dosya)

EJB : Enterprise Java Bean (Gelişmiş Java Fasulyeleri)

HIC : Health Insurance Card (Sağlık Sigorta Kartı)

HPC : Health Professional Card (Sağlık Uzmanı Kartı)

ID : Identification (Tanımlama)

INS : Instruction code (Komut kodu)

J2EE : Java 2 Enterprise Edition (Java 2 Gelişmiş Sürüm)

JCF : Java Card Framework (Java Kart Çatısı)

JCRE : Java Card Runtime Enviroment (Java Kart Çalışma Zamanı Ortamı)

KISALTMALAR (devam)

JCVM : Java Card Virtual Machine (Java Kart Sanal Makinesi)
JDBC : Java Database Connectivity (Java Veritabanı Bağlanırlığı)
JFC : Java Foundation Classes (Java Altyapı Sınıfları)
JRE : Java Runtime Enviroment (Java Çalışma Zamanı Ortamı)
MF : Master File (Ana Dosya)
MVC : Model, View & Controller (Model, Görünüm & Denetleyici)
OCF : Open Card Framework (Açık Kart Çatısı)
ODBC : Open Database Connectivity (Açık Veritabanı Bağlanılırlığı)
OMG : Object Management Group (Nesne Yönetim Grubu)
PIN : Personal Identification Number (Kişisel Tanımlama Numarası)
PIX : Proprietary Indetifier Extension (Tescilli Kimlik Uzantısı)
PKI : Public Key Infrastructure (Genel Anahtar Altyapısı)
RFU : Reserved For Future Use (Gelecekte Kullanılmak Üzere Ayrılmış)
RID : Resource Identifier (Kaynak Tanımlayıcısı)
RMI : Remote Method Invocation (Uzak Metot Uyandırma)
ROM : Read Only Memory (Salt Okunur Bellek)
RPC : Remote Procedure Call (Uzak Yordam Çağrısı)
RSA : Rivest Shamir Adleman
SIM : Subscriber Identity Module (Abone Kimlik Modülü)
SQL : Sequential Query Language (Sıralı Sorgu Dili)
SSL : Secure Socket Layer (Güvenli Soket Katmanı)
TPDU : Transfer Protocol Data Unit (İletim Protokolü Veri Birimi)
XML : Extended Markup Language (Genişletilmiş Biçimleme Dili)
WWW : World Wide Web (Dünya Çapında Ağ)

1 GİRİŞ

Hastanelerde ve diğer tıp kurumlarında otomasyon sistemlerine olan ihtiyaç gün geçtikçe artmaktadır. Bu sistemler kullanılarak; sağlık bilgilerinin doğru ve hızlı bir şekilde elde edilmesi, hastalara daha kaliteli bir hizmetin sunulması ve sağlık personelinin görevini daha rahat ve etkin olarak yerine getirmesi hedeflenmektedir. Bilgi sistemleri ve bilgisayar ağlarındaki son gelişmeler sağlık kayıtlarının elde edilmesi, saklanması, işlenmesi ve kullanılmasını sağlamada hastane otomasyon sistemlerinin bir bilgi ağını içermesi sonucunu ortaya koymuştur.

(Tunalı et al., 2002)'de de belirtildiği gibi sağlık bilgi sistemlerinde hastalara ait bilgiler bir çok kaynaktan elde edilebilir. Bu kaynaklar arasında hastanın kendisi (hastalıklara ait belirtiler, vücut ısısı, vb.), uygulanan testler (örneğin kan grubu testi), çevrimiçi (online) hasta izleme sistemleri (kol bantları, hız ölçerler, vb.), hastalıklara ait doktor teşhisleri ve önceden saklanan hasta bilgileri sayılabilir. Elde edilen bilgilerin analizi ve değerlendirmesi tıp bilimini ilgilendirmesine karşın bu bilgilerin saklanması ve paylaşımı bilgi teknolojilerini ilgilendirmektedir. Yeni bulgular ile beraber hastanın önceki bilgilerine erişim özellikle doktorlar tarafından doğru teşhislerin konulmasında büyük önem taşımaktadır.

Hastaya ait sağlık bilgilerine hızlı erişim tedavide yine büyük önem taşımaktadır. Bir hastane bünyesinde yer alan veritabanlarından bu bilgileri elde etmek mümkündür ancak hizmet verilecek hasta sayısının artması ve özellikle hızlı bir biçimde bu bilgilere erişim söz konusu olduğunda sürekli veritabanı erişimi sorun olmaktadır. Bunun yanı sıra her zaman veritabanına erişmek için gerekli ağ bağlantısı bulunmayabilir veya bilgilere hastanın kayıtlı olmadığı bir hastanede veya kurumda ihtiyaç duyulabilir. Halihazırda bir çok hastanenin kendine özel bir bilgi sisteminin olması ile beraber bu sistemlerin birbirleri ile ortak bir protokol üzerinden haberleşmesi ciddi anlamda gerçekleştirilememektedir. Buna neden olarak bilgisayar ağ altyapısı, bilgi güvenliği ve sistemlerin çok özel çözümler olması sayılabilir.

Ortaya konan bu ihtiyaçların karşılanması ve problemlerin en aza indirilmesi hastane otomasyon sistemlerinin kapasitelerinin, gelişmiş bilgi depolama ve geri getirim mekanizmaları sayesinde arttırılması ile mümkündür. Bunlara ek olarak, deneyimler ve gelişmiş hasta hizmet kalitesine sahip sistemler de göstermektedir ki taşınabilir medya aygıtları sağlık otomasyon sistemlerinin bir parçası olmak zorundadır. En çok öne çıkan; sözü edilen hasta bilgilerinin hastaya ait taşınabilir bir medya üzerinde yer alması ve sağlık uzmanının bu medyayı kullanarak işlemlerini yerine getirmesidir.

Yukarıda sözü geçen amaçlara hizmet edecek medyanın maliyetinin ucuz olması, kullanımının, taşınmasının ve yeni bilgilerle güncellenmesinin kolay olması ve darbelere karşı dayanıklı olması gerekmektedir. Bu tez çalışmasında olduğu gibi hastane otomasyon sistemlerinde yer alması savunulan ve bahsedilen özelliklere sahip medya *"akıllı kart"*lardır.

Akıllı kartlar içinde bilgi saklayan ve bu bilgiyi işleyen, taşınabilir bir entegre olarak tanımlanabilirler. Günümüzde, özellikle telekomünikasyon ve ulaşım sektörlerinde yaygın kullanımı mevcut olan bu kartlar aslında birer küçük bilgisayardır. Kendilerine ait bellekleri ve işlemcileri mevcuttur. Özellikle hız, güvenlik ve taşınabilirlik özellikleri son dönemde bu kartları popüler hale getirmiştir.

Bu tez çalışmasında; akıllı kartlara dayalı bir sağlık sistemi tasarlanmış ve gerçekleştirilmiştir. Sistem bir hastane bünyesinde; sağlık bilgisi akışında akıllı kartların kullanılmasını ve yine tez kapsamında hazırlanan dağıtık bir protokol üzerinden bilgi alışverişinin sağlanmasını içermektedir.

Sistemde hem hastaların hem de sağlık uzmanlarının kullanabileceği iki tip akıllı kart tasarlanmış ve sisteme entegrasyonu gerçekleştirilmiştir. Hasta akıllı kartları üzerinde hasta genel bilgileri dışında, hasta acil durum iletişim bilgileri, hasta sağlık bilgileri ve sisteme ait iletişim bilgileri yer alır. Doktor kartlarında ise doktorları

tanımlayıcı bilgiler yer almaktadır. Kartlar ayrıca sistem içerisinde kullanıcılarının yetkilendirilmelerini ve verilerin şifreli olarak protokol üzerinden iletilmesini de sağlamaktadır. Sağlık personeline ait akıllı kartlarda bu kişileri tanımlayan sayısal imzalar yer almakta, hasta kartlarında ise hasta bilgilerinin şifreli olarak sistem protokolü üzerinden iletilmesini sağlayacak kart sahibine özel şifreleme anahtarları bulunmaktadır.

Gerçekleştirilen sistemde doktor muayenehanelerinde birer akıllı kart terminalinin yer alması, bu terminallerin de dağıtık nesne protokolü üzerinden sistem sunucularına bağlanması tasarlanmıştır. Hasta akıllı kartı ile doktora başvurduğunda, doktor muayene sırasında ihtiyaç duyduğu bilgiye hasta akıllı kartından erişmekte, muayenesini tamamladıktan sonra da muayene ve varsa reçete bilgileri ile hasta akıllı kartını güncellemektedir. Muayene sonrası hasta sistem yönetim birimine başvurduğunda güncellenmiş bilgilerin hastane veritabanında güncellenmesi gerçekleştirilir.

Hazırlanan sistemde özellikle hasta akıllı kartına ait bilgilerin tutulduğu merkezi bir hastane veritabanı yapısı ortaya konmuştur. Hastane bölümlerine özel hasta bilgilerinin ise bu bölümde yer alan veritabanlarında tutulması tasarlanmıştır. Bu veritabanlarına güvenli ve yetkilendirilmiş erişim de yine akıllı kartlar aracılığı ile gerçekleştirilmiştir.

Tez hazırlanırken Ege Üniversitesi Tıp Fakültesi Nöröşirürji ABD ile ortak çalışmada bulunulmuş ve bu bölümde halihazırda çalışmakta olan bilgi sistemi incelenmiştir. Bölüm bazlı sistem bileşenleri bu bölüm ihtiyaçlarına cevap verebilecek şekilde hazırlanmıştır.

Tezin bundan sonraki organizasyonu şu şekildedir: İkinci bölümde tezin hazırlanmasında gerekli olan alt yapı bilgileri ve daha önce bu konuda gerçekleştirilmiş olan çalışmalar yer almaktadır. Akıllı kart başta olmak üzere kullanılan teknolojiler hakkında bilgi verilmiş; akıllı karta dayalı geliştirilmiş ve geliştirilmekte olan diğer projeler ortaya

konmuştur. Üçüncü bölümde hazırlanan sistemin genel tanıtımı ve yazılım mimarisi anlatılmıştır. Sistemin işleyişi, protokoller ve kullanıcı oturumları hakkında bilgi verildikten sonra nesne tabanlı sistem yazılım mimarisi, kullanılan tasarım deseni ve sistem bileşenleri ele alınmıştır. Dördüncü bölümde sistem akıllı kart bileşenleri anlatılmıştır. Kullanılan akıllı kartların teknik özellikleri, bu kartlar üzerinde çalışmak üzere hazırlanan yazılımların tasarımları açıklanmıştır. Akıllı kartlar ile iletişime geçecek istemci ara yazılımı da tüm detayları ile yine bu bölümde yer almaktadır. Beşinci bölümde hazırlanan sistemin sunucu bileşenleri hakkında bilgi verilmiştir. Sistem dağıtık nesne protokolü ve istemci sunucu ağ iletişiminde görev alan yazılımlar hakkındaki teknik bilgiler bu bölümde yer almaktadır. Altıncı bölümde sistem kullanıcıları ile etkileşimde bulunacak olan istemci yazılımları ve ara yüz bileşenleri hakkında bilgiler yer almaktadır. Yedinci bölüm ise sonuç bölümüdür. Tez çalışması sonucunda ortaya konan ürün ve ileriye yönelik yapılabilecek çalışmalar hakkında bilgi verilmiştir. Bu bölümü kaynaklar dizini ve ekler takip etmektedir.

2 ALT YAPI VE İLGİLİ ÇALIŞMALAR

Bu bölümde tez çalışması için gereken alt yapı bilgileri ve daha önce bu konuda gerçekleştirilmiş olan çalışmalar yer almaktadır. Alt yapı alt bölümünde akıllı kart, akıllı karta dayalı teknolojiler ve Java RMI dağıtık nesne protokolü hakkında bilgiler yer almaktadır. İkinci alt bölümde ise sağlık sektöründe akıllı kartların kullanılması ile ilgili diğer çalışmalar hakkında bilgi verilmiştir.

2.1 Alt Yapı

2.1.1 Akıllı kart

Akıllı kart, içinde bilgi saklayan ve bu bilgiyi işleyen, taşınabilir bir entegre olup. standart hafıza-yonga kartlarının gelişmiş bir versiyonudur. Standart hafıza-yonga kart sistemlerinde, veri işleme donanım ve yazılımı karttaki bilgiyi okumakta, hesaplamaları yapmakta ve veriyi tekrar karta yazmaktadır. Standart hafıza-yonga kartları çoğu zaman "kullan ve at" şeklinde hizmet verebilecek kadar ucuzdurlar (Örnek: Telefon kartları). Akıllı kart sistemlerinde ise kartın içinde hafızaya ilave olarak bir mikroişlemci bulunmakta ve gerekli hesaplamalar onun vasıtası ile yapılabilmektedir. Diğer bir deyişle *akıllı kartlar, kart üzerine monte edilmiş bilgisayarlardır*.

Akıllı kart ve benzeri sistemlerin ortaya çıkmasının nedenlerinden biri; bilgisayarlarda saklanan bilgilere yetkili kişilerin ulaşabilmesi için, ev kapı anahtarının sadece ev sahibinde olması gibi bir anahtara ihtiyaç olmasıdır. 1960-1970 yılları arasında gerekli olan sadece bir müşteri veya personelin numarasını taşıyan karttı. ***Barkod*** sistemleri bunun için yeterli oluyordu. Şahsın tanıtıcı numarası kartın üzerine barkod olarak yazıldıktan sonra kullanıcı bu kart ile ilgili uygulamalara kendisini tanıtıyordu. Fakat zamanla güvenlik ve aşınma problemleri meydana gelince barkoddan daha gelişmiş olan ve üzerinde şahsı sisteme tanıtan numara ile birlikte isim de saklayabilen ***Manyetik*** kartlar kullanılmaya

başlandı. Manyetik kartlar barkodlardan daha güvenliydi; barkod gibi kısa sürede deforme olmuyordu ve kullanıcının ismini de üzerinde taşıyabildiği için fazladan güvenlik sağlayabiliyordu. 1970'lerin sonunda akıllı kartlar bulundu ve patenti alındı. Fakat o zamanki teknoloji için manyetik kartlar yeterliydi ve akıllı kart kullanımı pahalı bir teknoloji gerektiriyordu. Geçen 20 sene içinde akıllı kart gereken ilgiyi görmedi. 1990'larda PC'lerin dünya genelinde yaygınlaşması, Internet ve buna bağlı ticaretin patlaması, yüksek teknolojinin maliyetinin 70'li yıllara göre yüzlerce kat düşmesi ve en önemlisi güvenliğin her alanda aranması sonucu akıllı kartlar tekrar gündeme geldi.

Manyetik kartlar aynı bir teyp bandı gibidir. Üzerinde sadece kısıtlı bilgi saklayabilir, bilgi işleme gücü yoktur, üzerindeki bilgiler kolaylıkla üçüncü şahıslar tarafından değiştirilebilir ve belli bir zaman içinde yenilenmeleri gerekir. Şu an kullanılan banka kartlarının hepsi manyetik bant şeridi taşımaktadır ve hepsinin de bir son kullanma tarihi vardır.

Akıllı Kartlar ise bir bilgi saklama alanı bulundurması ile birlikte yukarıda da belirtildiği gibi bir işlemci de taşırlar. Önceki kart jenerasyonlarına üstünlük sağlamasındaki en önemli etken bir mikroişlemci içermesinden kaynaklanan bilgi işleme gücüdür. İşlemler sırasında uzak veritabanlarına bağlanmasına gerek yoktur.

2.1.1.1 Akıllı kartların sağladığı faydalar

Akıllı kartlara olan ilgi sunmuş olduğu faydaların bir sonucudur. Yerleşik bilgi işleme gücü yanında *güvenlik, taşınabilirlik* ve *kolay kullanım* akıllı kartların temel avantajlarıdır.

İşlemci, bellek ve giriş/çıkış üniteleri tek bir entegre devre içinde ve plastik bir kart üzerine yerleştirilmiştir. Kart, zarar veren potansiyel dış kaynaklara bağlı olmadığından darbelere ve saldırılara karşı dayanıklıdır. Karttan bilgi edinilmesi için fiziksel olarak kartın bilgi istemcisinin elinde olması, akıllı kart yazılım ve donanımı hakkında

temel bilgi sahibi ve ek bilgi okuma aygıtlarına ihtiyaç vardır. Ayrıca kart üzerindeki kriptografik fonksiyonlar da kart güvenliğini biraz daha arttırmaktadır. Kart üzerinde veriler şifreli olarak saklanabilir ve kart ile bilgi istemcisi arasındaki veri iletişimi şifreli ve imzalı olarak gerçekleştirilebilir. Bunların yanına ek olarak; akıllı kartlara erişim için genellikle bir PIN'e (Personal Identification Number) ihtiyaç vardır. Bu PIN, yetkilendirilmemiş kimselerin kartı kullanmasını engellemektedir. Yerleşik işlem gücü burada da etkisini gösterir. Şöyle ki: Kart erişimi için girilen şifreyi (PIN) istemci tarafındaki bir işlem birimi değil de kartın kendisi kontrol eder. Şifre doğru ise kart oturumu açılır yoksa karta erişim engellenir. Bir oturumda arka arkaya yanlış şifre girişleri durumunda kart kendisini kilitleyecektir.

Akıllı kartın sağladığı diğer bir fayda ise taşınabilirliği olup bunun sonucunda içerdiği bilgilere kart sahibi yanında taşıdığı sürece gittiği her yerden ulaşılabilmektedir. Akıllı kartlar kullanım açısından oldukça rahat ve uygundur. İşleme başlamak için kartın bir kabul cihazına yerleştirilmesi ve işlem tamamlandığında cihazdan çıkarılması yeterli olmaktadır.

2.1.1.2 Akıllı kartların kullanım alanları

Akıllı kartlar daha çok güvenli olarak veri saklamada ve işlemlerin yetkilendirilmiş ve güvenli olarak gerçekleştirilmesinde kullanılmaktadır. Akıllı kartların en çok kullanıldığı alanlar ve bu sektörlerde kullanılan kartlara ait özellikler şöyle özetlenebilir:

- Telekomünikasyon Endüstrisi:
  - Cep telefonlarında yer alan SIM (Subscriber Identity Module) kartlar. Özellikle Türkiye'de akıllı kartlar ile gerçek anlamda ilk tanışma cep telefonlarının kullanılması ile gerçekleşmiştir.
  - Halkın kullanımına açık telefonların kullanımını sağlayan önceden ödemeli ve içerisinde telefon görüşme kontörü içeren kartlar

- Ödeme ve Bankacılık Sistemleri:
  - Güvenli para yatırma ve para çekme kartları
  - Elektronik cüzdan
- Ulaşım Sektörü: Kağıt bilet veya paso yerine toplu taşımada kullanılan kartlar
- Sağlık Sektörü: Hasta temel sağlık bilgilerini içeren ve hastane veritabanı sistemlerine güvenli ve yetkilendirilmiş erişimi sağlayan kartlar
- Internet: Akıllı kartların Internet sitelerine girişte kişileri tanımlama ve erişim kontrolünü sağlamada ve e-posta mesajlarının dijital imzalanmasında kullanımı mevcuttur.
- Kuruluşlara, bilgisayar ortamlarına güvenli giriş ve bu ortamlarda derecelendirilmiş yetkilendirmeyi sağlamada yine akıllı kartlar kullanılmaktadır.

2.1.1.3 Temel akıllı kart çeşitleri

Akıllı kartlar hafıza ve mikroişlemci kartları olmak üzere veya kontaklı ve kontaksız olmak üzere kategorize edilebilirler.

Hafıza kartları ve mikroişlemci kartları:

Akıllı kartlar ilk çıktığı zamanlarda bugünkü temel özellikleri olan bir mikroişlemciye sahip değillerdi. Hafıza kartları olarak adlandırılan bu kartlarda 1 Kbayt ile 4 Kbayt arası bilgi saklamak mümkündü. Ucuz maliyetinden dolayı özellikle kontörlü telefon kartı olarak kullanımı yaygındır. Mikroişlemci kartları ise adından da anlaşılacağı gibi bir işlemciye sahip olup veri işleme ve özellikle güvenlik alanında daha önce de bahsedilen gelişmiş yetenekleri vardır.

Kontaklı ve kontaksız kartlar:

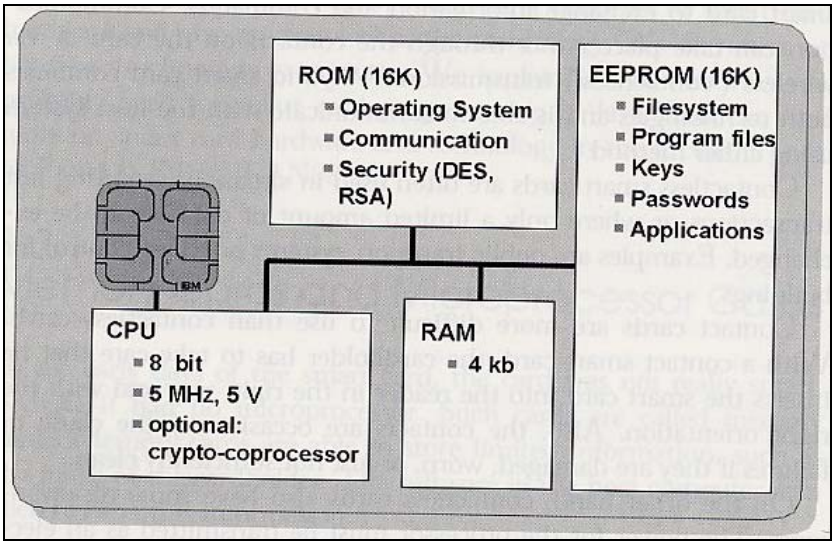
Kontaklı kartlar bir kart kabul cihazına yerleştirilerek işlemlerde kullanılırlar. Kart entegresi gereken elektriği bu kabul cihazından almakta ve seri iletişim ara yüzü ile istemci bileşenlerle veri alışverişinde bulunmaktadırlar. Kontaksız kartlar ise bir kart kabul cihazına yerleştirilmeyip, karta yerleştirilmiş bir anten aracılığı ile veri iletişimini gerçekleştirmektedir. Gerekli güç ise kart üzerinde yer alan bir pilden elde edilebileceği gibi antenden de elde edilebilir. Kontaklı kartlara göre bu tip kartların avantajı bir kabul cihazına yerleştirme ve bundan kaynaklanabilecek aşınmalardan kurtulmadır. Ancak maliyetlerinin pahalı olması ve kart kabul cihazı ile belli bir mesafeden iletişime geçme zorunluluğu ve buna bağlı olarak belli zaman aralığında sınırlı veri iletişimi bu kartların dezavantajı olarak görülmektedir.

2.1.1.4 Akıllı kart donanım mimarisi

Şekil 2.1'de de görüldüğü gibi bir akıllı kart entegresi; bir mikroişlemci, ROM (Read Only Memory), EEPROM (Electrical Erasable Programmable Read Only Memory) ve RAM (Random Access Memory – Rastgele Erişimli Bellek) bileşenlerini içermektedir.

Şu an mevcut birçok akıllı kart pahalı olmayan 8-bitlik mikroişlemci içermekte olup genelde Motorola 6805 veya Intel 8051 komut kümesine sahiptir. Saat hızları 5 MHz kadardır. Yüksek teknolojili kartlar 16 veya 32-bit'lik işlemciler içermektedir. Bugünün akıllı kartlarının işlem gücü yaklaşık olarak ilk kişisel bilgisayarların işlem gücüne eşittir.

Güvenlik uygulamalarında kullanılan bir çok kartta asıl işlemci yanında kriptografik işlemler gerçekleştiren yan işlemci bulunmaktadır. Maliyeti doğrudan etkileyen bu işlemciler modüler aritmetik ve büyük sayı hesaplamalarında görev alırlar.



Şekil 2.1 Bir akıllı kart entegresi ve bileşenleri (Hansmann et al., 2000)

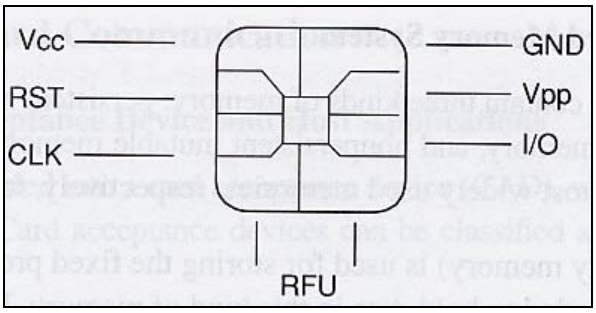
Kart üretilirken bellek mimarisinde yer alan ROM bölgesine, kart işletim sistemi, kalıcı uygulamalar ve kullanıcı bilgileri yazılmaktadır. Kart üretildikten sonra bu bölgeye bilgi yazılamaz. Bu alanda bilgilerin tutulması için elektriksel güce ihtiyaç yoktur.

EEPROM bölgesinde de tıpkı ROM'da olduğu gibi kalıcı bilgiler tutulmaktadır. ROM'dan farkı ise bu alana kart üretildikten sonra bile bilgiler ve uygulamalar yazılabilir. Bu bellek alanı kişisel bilgisayarlardaki sabit disklere benzetilebilir. Bilgiler bu alanda yaklaşık olarak 10 yıl süre ile saklanabilmektedir.EEPROM'dan bilgi okuma RAM'den bilgi okuma kadar hızlı olmasına rağmen bilgi yazılması 1000 kat daha yavaştır.

RAM bölgesi bilgi modifikasyonu ve depolamada geçici çalışma alanı olarak kullanılmaktadır. Bu bellekte bilgiler, kart bir güç kaynağına bağlı kaldığı sürece tutulmaktadır. Karttan elektrik kesildiğinde bu bilgiler kaybedilmektedir. Bu bellek alanı kişisel bilgisayarlardaki ana bellek ile aynı işleve sahiptir.

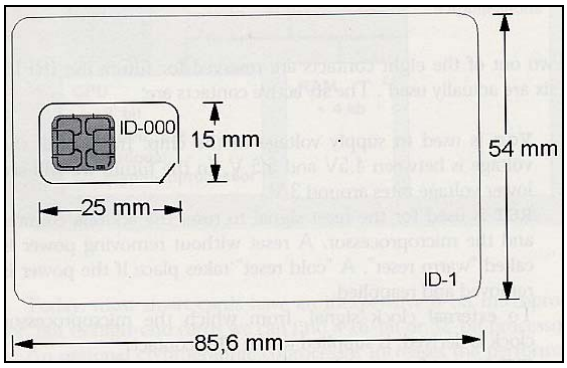
Bir akıllı kartın 8 adet kontak noktası vardır. Şekil 2.2'de görülen bu noktaların görevleri, pozisyonları ve boyutları ISO 7816-2 standardında belirtilmiştir.

- Vcc karta 3 ya da 5 voltluk güç sağlamaktadır.
- RST, mikroişlemciye yeniden başlatma (reset) sinyali göndermede kullanılır. Bu şekilde yeniden başlatma *"warm reset"* olarak adlandırılmaktadır. Kartın kart kabul cihazından çıkarılıp yeniden yerleştirilmesinde olduğu gibi karta giden voltajın kesilmesi ve yeniden verilmesi ise *"cold reset"* olarak adlandırılmaktadır.
- Akıllı kartlarda iç saat sinyali mikroişlemci tarafından oluşturulmayıp bu sinyal CLK noktasından elde edilen dış saat sinyalinden türetilmektedir.
- GND ise referans voltaj olarak kullanılmakta olup değeri 0 volt olarak kabul edilir.
- Vpp seçimlik olup sadece eski model kartlarda, EEPROM alanını programlamada kullanılmaktadır.
- I/O, akıllı kart ile dış dünya arasında bilgi alışverişinin yarı çift yönlü (half-duplex) olarak yani belli bir zaman aralığında sadece tek yönde iletim şeklinde gerçekleşmesini sağlar.
- RFU (Reserved For Future Use) noktaları ise daha sonra kullanılmak üzere ayrılmıştır.



Şekil 2.2 Akıllı kart kontak noktaları (Chen, 2000)

Akıllı kartlar için belirlenmiş olan iki kart boyutu formu Şekil 2.3'de de gösterilen ID-1 ve ID-000'dır. ID-1 formu bir kredi kartı ile aynı boyuta sahip olup birçok uygulamada akıllı artlar bu formda kullanılmaktadır. ID-000 ise daha çok cep telefonlarında kullanılan formdur.



Şekil 2.3 Akıllı kart boyutları: ID-000 ve ID-1 (Hansmann et al., 2000)

Aynı kartı hem akıllı kart olarak yeni uygulamalarda hem de manyetik kart olarak eski sistemlerde kullanma hedeflendiğinden akıllı kart boyutu manyetik kartlarla aynı tutulmuştur.

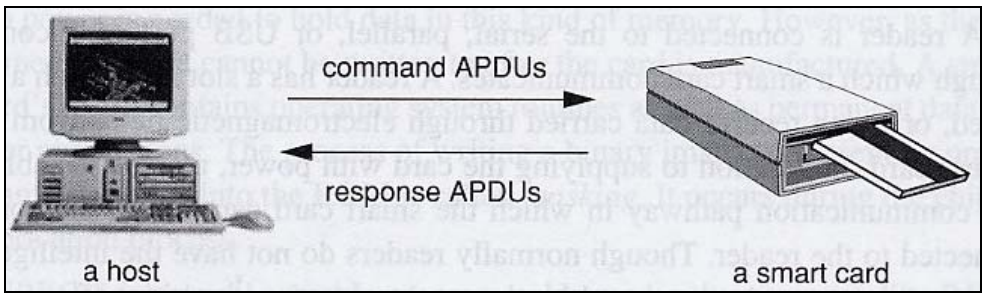
2.1.1.5 Akıllı kart iletişimi

Uygulamalarda akıllı kartların kullanılması için kartların *CAD (Card Acceptance Device)* adı verilen aygıtlara yerleştirilmesi gerekir. CAD aygıtları *okuyucu* ve *terminal* olmak üzere ikiye ayrılır. Okuyucular seri, paralel veya USB bağlantı noktasından (port) akıllı kartın iletişime geçeceği bilgisayarlara bağlıdırlar. Kartlar bunlar üzerindeki yuvalara (slot) yerleştirilerek kullanılırlar. Terminaller ise tek başlarına birer bilgisayar olup bünyelerinde okuyucuları barındırırlar. Banka ATM'leri bunlara örnek gösterilebilir. Kart okuyucusu olma özelliği yanında karttan alınan bilgileri işleme gücüne sahiptirler. Tezin bundan sonraki bölümlerinde kart ile iletişime geçen ev sahibi (host) uygulamaların üzerinde çalıştığı bilgisayarlar kart terminali olarak

belirtilecektir. Bu terminaller, kendilerine bir kart okuyucu ünitenin bağlı bulunduğu bilgisayarlar olarak düşünülmelidir.

Daha önce de belirtildiği gibi kart ile ev sahibi arasındaki iletişim kanalı yarı çift yönlüdür. Bilgisayar ağlarında nasıl veriler bir protokole (örneğin TCP/IP) dayanan paketler halinde transfer ediliyorsa aynı durum akıllı kart ile iletişimde bulunduğu bilgisayar arasında da geçerlidir. *APDU (Application Protocol Data Unit)* adı verilen veri paketleri ya emir ya da cevap mesajları içerirler.

Kullanılan model master / slave (efendi / köle) modeli olup akıllı kart köle, iletişimde bulunduğu terminal ise efendi durumundadır. Akıllı kart her zaman ana bilgisayardan emir APDU'ları beklemekte; gelen APDU içindeki komutları işlemekte ve ana bilgisayara cevap APDU ile yanıt vermektedir. Akıllı kart iletişim modeli Şekil 2.4'te gösterilmiştir.



Şekil 2.4 Akıllı kart iletişim modeli (Chen, 2000)

APDU protokolü, ISO 7816-4 standardında belirtildiği gibi, bir ev sahibi uygulaması ile bir akıllı kart arasında yer alan uygulama seviyesi protokolüdür. APDU mesajları yukarıda da belirtildiği gibi emir APDU ve cevap APDU adı verilen iki yapıda taşınmaktadır. Bu birimlerin yapısı Tablo 2.1 ve Tablo 2.2'de gösterilmiştir.

Zorunlu başlık				Seçimlik gövde		
CLA	INS	P1	P2	Lc	Veri alanı	Le

Tablo 2.1Emir APDU yapısı

Seçimlik gövde	Zorunlu kuyruk	
Veri alanı	SW1	SW2

Tablo 2.2 Cevap APDU yapısı

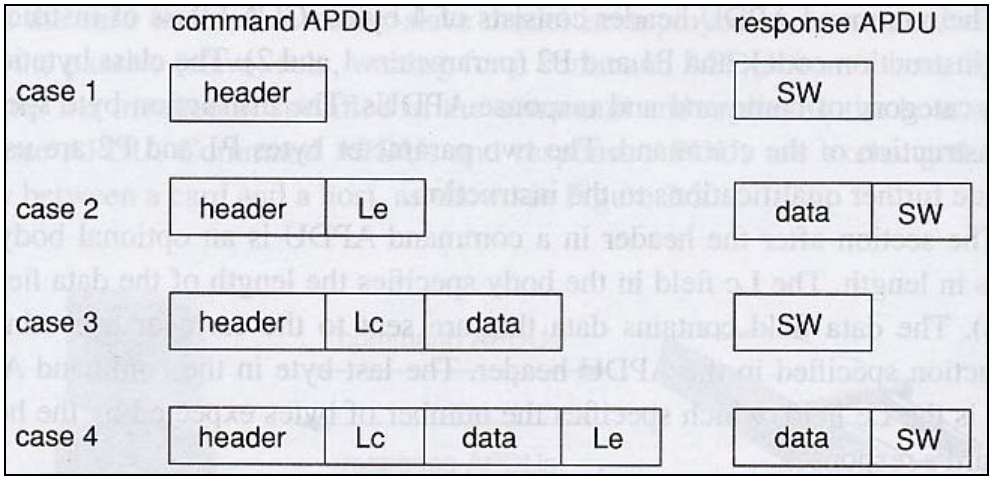
Emir APDU başlığı 4 bayttan oluşur: CLA (Class of instruction), emir APDU'sunun kategorisini tanımlar. INS (Instuction code), karta gönderilen komuttur. P1 (Parametre 1) ve P2 (Parametre 2) ise komutu destekleyici daha kapsamlı nitelikler sunmaktadır.

Seçimlik gövdenin boyutu ise değişkendir. Gövdedeki ilk bayt olan Lc, gönderilen verinin kaç bayt olduğunu bildirirken veri alanı gönderilen komutun işlenmesinde kullanılacak olan verileri içermektedir. Son bayt (Le) ise terminalin karttan kaç bayt veri beklediğini bildirir.

Terminalden karta gönderilen emir APDU'ya cevap olarak gönderilen cevap APDU da ise mutlaka gönderilen emir APDU'nun işlenmesinden sonra karttaki durumu gösteren bir bilgi yer alır. SW1 ve SW2 birlikte *"durum sözcüğü (status word)"* adı verilen yapıyı oluşturmakta ve durum bilgisini ev sahibi uygulamaya bildirmektedirler. Örneğin 0x9000 durum sözcüğü, kartta ilgili işlemin başarıyla yerine getirildiğini ev sahibi uygulamaya bildirir. Uzunluğu emir APDU'daki Le tarafından belirtilen bilgi kümesi ise seçimlik gövdede yer alır.

Tüm bunlara bağlı olarak iletişim modelinde 4 durum söz konusudur (Şekil 2.5):

- Durum 1: Karta veri transferi ya da karttan veri alımı yoktur. Emir APDU sadece başlığa sahip iken cevap APDU sadece zorunlu kuyruğa sahiptir.
- Durum 2: Karta veri transferi yokken karttan veri alımı mevcuttur. Emir APDU gövdesi 1 baytlık Le bilgisinden oluşurken, cevap APDU'da bu alanda belirtilen miktar kadar bilgi döndürülür.
- Durum 3: Karta veri transferi mevcutken karttan veri alımı yoktur. Emir APDU gövdesi Lc ve veri alanına sahip iken cevap APDU sadece durum bilgisi döndürür.
- Durum 4: Hem karta veri transferi hem de karttan veri alımı mevcuttur. Emir APDU'da Lc, veri alanı ve Le bilgileri yer alırken cevap APDU'da durum bilgisi dışında veri de yer alır.



Şekil 2.5 Emir ve cevap APDU durumları (Chen, 2000)

APDU paketleri ISO 7816-3'de belirtilen iletişim protokolü üzerinden gönderilmekte olup bu protokolün paket birimi ise *TPDU (Transfer Protocol Data Unit)* olarak adlandırılmaktadır. Kullanımda olan iki transfer protokolü T=0 (bayt orijinli) ve T=1 (bayt dizisi orijinli)'dir.

2.1.1.6 Akıllı kart işletim sistemleri

Masaüstü bilgisayarlarda görev alan işletim sistemlerinin aksine akıllı kartlar üzerindeki işletim sistemleri, daha çok kullanıcı uygulamalarının üzerinde işlem gördüğü bir komut koleksiyonundan ibarettir.

Birçok akıllı kart işletim sistemi ISO 7816-4 standardına dayanan dosya sistemini desteklemektedir. Dosya orijinli bu sistemlerde kullanıcı uygulamaları daha çok uygulamaya özel bilgi saklayan bir veri dosyasından ibarettir. Uygulama verilerine erişim komutları işletim sistemi tarafından uygulanmaktadır. Bu yüzden işletim sistemi ile uygulama arasındaki ayrım tam olarak tanımlanamamıştır.

ISO 7816-4 hiyerarşik bir dosya sistem yapısı sunar. Üç çeşit dosya vardır: *MF (Master File)*, tüm dosya sisteminin kökü olup diğer dosyaları içerisinde barındırır. Bir kartta sadece bir MF yer alır. *DF (Dedicated File)*, içerisinde birden fazla DF'yi ya da veri dosyasını barındıran, bir anlamda masaüstü bilgisayarlarındaki dizinlere karşılık gelen dosyalardır. *EF (Elementary File)* ise diğer dosyaları içermeyen en küçük veri dosyasıdır.

Dosya sistemi orijinli işletim sistemleri bir çok akıllı kart üzerinde kullanılmaktadır. Ancak daha iyi sistem-katman ayırımı sunan ve uygulama geliştirmeye daha uygun yeni kart işletim sistemleri gün geçtikçe daha popüler olmaktadır. Bir sonraki bölümde ele alınacak Java Card teknolojisi bu eğilimde yer alan en önemli teknolojidir. (Rankl and Effing, 2000; Chen, 2000; Hansmann et al., 2000)

2.1.2 Java Card teknolojisi

Java Card teknolojisi, akıllı kart uygulamalarının daha etkin, kolay ve hızlı bir biçimde tasarlanıp hayata geçirilmesini sağlamaktadır. Akıllı

kartların ve diğer bellek-sınırlı cihazların Java programlama dilinde yazılmış uygulamaları – ki bu uygulamalara "applet" adı verilir – çalıştırmalarına olanak sunar. Temelde Java Card teknolojisi, Java programlama dilinin birçok avantajını barındıran, güvenli, taşınabilir ve bünyesinde birden fazla uygulama (multiapplication) barındırabilen bir akıllı kart platformu tanımlamaktadır.

İlk sürümü 1996 yılında ortaya konan platformun Sun Microsystems tarafından 1999 yılında çıkarılan 2.1 sürümü tez kapsamında yer alan sistemde kullanılmıştır. Aşağıdaki bölümlerde de platformun 2.1 sürümüne ait bilgiler yer almaktadır.

2.1.2.1 Faydaları

Akıllı kart uygulama geliştiricilerine Java Card teknolojisinin sunduğu avantajlar şu şekilde özetlenebilir:

- *Kolay uygulama geliştirilmesi:* Java dilinin kullanılması ile yazılım geliştiriciler, çevirici dili (assembly language) ile mikroişlemci programlamaya gerek kalmadan hızlı kart uygulamaları hazırlayabilmektedirler. Java Card platformu, uygulama geliştiricileri akıllı kart sisteminin detaylarından ve karmaşıklığından soyutlamakta ve geliştiricilerin daha çok kendi uygulamalarına ait detaylara yönelmelerine izin vermektedir. Ayrıca nesne tabanlı olarak uygulamaların tasarımı, uygulamaların yeniden kullanılabilirliğini oldukça kolaylaştırmaktadır.
- *Güvenlik:* Java programlama dili içerisinde yer alan bütünleşik güvenlik özelliklerinden (örneğin kontrollü metot çağırma, bellek göstergelerinin (pointer) kullanılmasından kaynaklanan ve kaynağının bulunması oldukça zor olan hatalardan arınma, vb.) kart yazılımı geliştiricileri faydalanmaktadır. Bunun yanı sıra ileride de bahsedilecek olan "*applet kalkanı" (applet firewall)*, kart içerisinde yer alan *"applet"*lerin korunmasında ve

birbirleriyle kontrollü olarak kart içi iletişimlerinin sağlanmasında görev almaktadır.

- *Donanım bağımsızlığı:* Java Card *"applet"*ları kart içinde yer alan Java Card platformu üzerinde çalıştırılacaklarından tamamıyla kart donanımından bağımsızdırlar. Platform üzerinde çalışılabilecek halde bulunan tüm *"applet"*ler herhangi bir Java akıllı kartında yeniden derlenmeden çalışabilir.
- *Birden fazla uygulamanın kart üzerinde yer alması:* Bir Java akıllı kartı birden çok uygulamayı barındırabilir. Örneğin aynı kartta farklı uygulama geliştiriciler tarafından geliştirilmiş bir banka uygulaması ve bir tıp uygulaması yer alabilir. Bunlar birbirinden bağımsız olarak faaliyette bulunurlar. Java Card kalkan mekanizması sayesinde, izin verilmedikçe birbirlerine erişemezler.
- *Varolan akıllı kart standartlarına uyumluluk:* Java Card teknolojisi uluslar arası akıllı kart standardı olan ISO 7816'ya dayalı olduğundan bu standartla uyumlu tüm sistemleri ve uygulamaları desteklemektedir.

2.1.2.2 Java Card sistem mimarisi:

Java Card teknolojisinde, Java sistem yazılımının uygulamalara yeterince çalışacak alan bırakacak şekilde akıllı karta yerleştirilmesi hedeflendiğinden Java dilinin özelliklerinin belli bir alt kümesi desteklenmiş ve iki parçadan oluşan bir Java sanal makine mimarisi uygulanmıştır.

Java Card sanal makinesinin biri kart üzerinde biri de kart dışında çalışan iki parçası mevcuttur. Uygulama çalışma zamanında yer almayan sınıf yükleme, baytkodu (bytecode) doğrulama, bağlama ve optimizasyon gibi sistem kaynak kapasitesinin önemli olmadığı bir çok işlem kart dışında çalışan sanal makine üzerinde gerçekleştirilmektedir. Bu ayrık sanal makine mimarisinden dolayı platform akıllı kart ve masaüstü bilgisayar üzerinde dağıtık yapıda olup üç parçadan oluşur:

- JCVM (Java Card Virtual Machine), Akıllı kart uygulamalarına uygun Java programlama dili alt kümesi ve sanal makine tanımlamalarını içerir.
- JCRE (Java Card Runtime Enviroment), bellek yönetimi, *"applet"* yönetimi ve diğer çalışma zamanı özellikleri içeren kart çalıştırma işlemlerini tanımlar.
- Java Card API (Application Programming Interface), akıllı kart uygulamalarını programlamada kullanılacak olan çekirdek ve uzantı Java paketlerini ve sınıflarını tanımlar. Bu paketler ve sınıflar *JCF* (*Java Card Framework)*'yi oluştururlar.

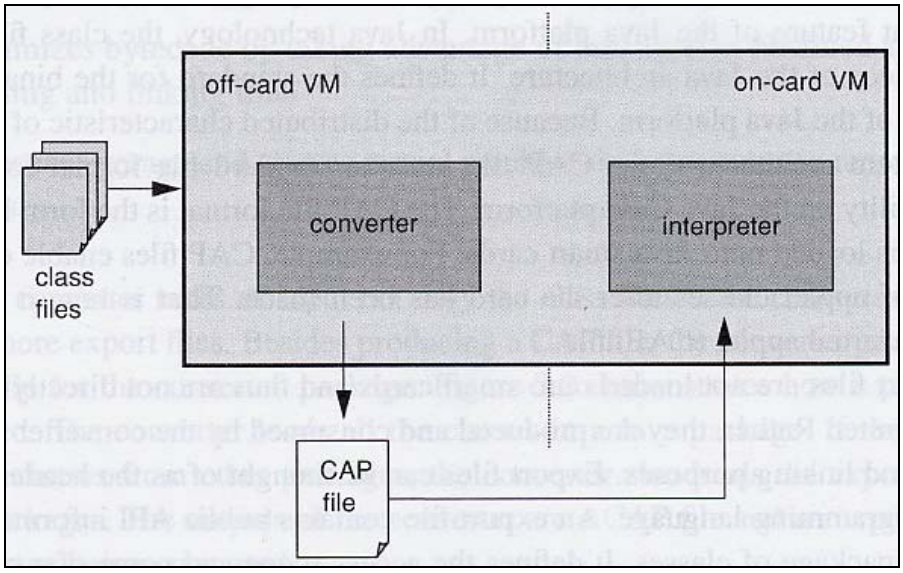
Java Card Dili Altkümesi:

Akıllı kart bellek kısıtlamalarından dolayı Java Card platformu, Java programlama dilinin belli bir alt kümesini içermektedir. Bu küme Java'nın nesneye dayalı program geliştirme özelliğini koruması yanında akıllı kartlar ve diğer bellek kısıtlı aygıtlara program hazırlanmasına uygun özellikler taşımaktadır.

Java Card çatısında desteklenmeyen Java programlama özellikleri: *long, double* ve *float* gibi büyük primitif veri tipleri (sadece *boolean, byte* ve *short* tipleri kullanılabilir), karakter kümeleri, çok boyutlu dizinler, dinamik sınıf yükleme, artık toplama (garbage collection), iş parçacıkları (thread), nesne serileştirme (serialization) ve nesne klonlamadır.

Java Card Sanal Makinesi:

Şekil 2.6'da gösterildiği gibi JCVM, *yorumlayıcı (interpreter)* adı verilen, kart üzeri parça ve *çevirici (converter)* adı verilen, kişisel bilgisayar ya da iş istasyonu üzerinde çalışan parçadan oluşmaktadır.



Şekil 2.6 Java Card Sanal Makinesi (Chen, 2000)

Çevirici kart üzerinde çalışacak olan uygulamaya ait Java paketlerini ve sınıflarını yükleyip ön işlemden geçirdikten sonra *CAP (Converted Applet)* adı verilen karta yüklenmeye hazır bir dosya haline getirir. Bu dosya akıllı karta yüklenir ve yorumlayıcı tarafından çalıştırılır. Yorumlayıcı aynı zamanda kart içi bellek yönetimi, nesne oluşumu ve çalışma zamanı güvenliğinden de sorumludur.

CAP dosyasının akıllı karta yüklenmesi işlevini *Java Card Kurucusu (Installer)* yerine getirir. Kart içerisinde yer alan bu yapı kart dışındaki yükleme programı ile ortak çalışır.

Java Card Çalışma Zamanı Ortamı:

JCRE, Java Card kart içerisinde çalışan sistem bileşenlerini içermektedir. Kart içi kaynak yönetimi, ağ iletişimi, *"applet"* çalıştırma ve kart üzeri güvenlikten sorumludur. Temelde *akılı kartın işletim sistemi* olarak görev almaktadır.

Şekil 2.7'de de görüldüğü gibi JCRE akıllı kartın donanımı ve yerli (native) sistemi üzerindedir. Java Card sanal makinesinin yorumlayıcı kısmını, Java Card uygulama çatısını (API'ler), endüstriye özel ekleri ve JCRE sistem sınıflarını içermektedir:

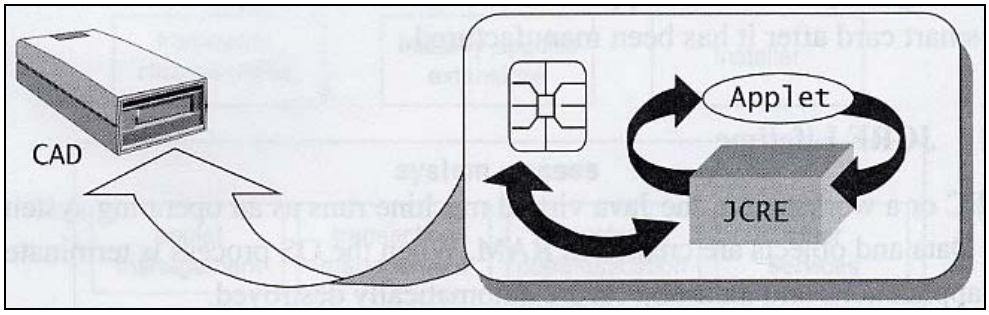
- JCVM'nin yorumlayıcı kısmı daha önce de belirtildiği gibi baytkodu işletimi ve bellek ataması gibi işlemlerden sorumludur.
- Yerli metotlar alt seviye iletişim protokollerinin kullanılması, kriptografik destek gibi işlemlerde JCVM'yi ve sonraki katman sistem sınıflarını destekler.
- Sistem sınıfları bir işletim sisteminin çekirdeğine benzemektedirler. İşlemlerin ve ev sahibi uygulama ile kart *"applet"*leri arasındaki iletişimin yönetilmesinden sorumludurlar.
- Java Card uygulama çatısı daha sonra anlatılacağı üzere kart üzeri uygulama programlama için ara yüz sunmaktadır.
- Özel endüstri paketleri, sistem modeli ve güvenlik ile ilgili ek servisler sunmaktadır.
- Kurucu daha önce de belirtildiği gibi kart üretildikten sonra karta yazılımların güvenilir bir şekilde yüklenmesini sağlar.

Java Card *"applet"*leri, Java Card platformunda çalışan kullanıcı uygulamaları olup Java Card çatısı kullanılarak yazılmışlardır. *"applet"*ler kart üretildikten sonra Java akıllı kartına yüklenebilmektedirler.

JCRE, akıllı kart üzerine yüklendiğinden itibaren çalışmaya başlar ve hiçbir zaman sonlanmaz. Akıllı kart üretim sonucu ilk çalışmaya başladığında JCRE de çalışmaya başlar ve kart ömründe JCRE başlangıcı sadece bir kez gerçekleşir. Bu ilklendirmede (initialization) JCRE; sanal makinesini başlatmakta ve servislerini sürdürmede kullanacağı nesneleri oluşturmaktadır. *"applet"*ler de yüklendikçe JCRE tarafından örnekleri (instance) kart içerisinde oluşturulmakta, oluşan örnekler de veri saklamada kullanacağı nesneleri hazırlamaktadır. Burada unutulmaması gereken kart üzeri uygulama programlarının da birer nesne olduğudur.

Kart üzerinde güç kesildiğinde sadece sanal makine askıya alınmaktadır. JCRE'nin ve kart üzerinde oluşturulan kalıcı nesnelerin tüm durumları ve içerikleri EEPROM üzerinde saklanmaktadır. Karta yeniden enerji verildiğinde JCRE kalıcı bellekten ilgili verileri alarak sanal makineyi yeniden başlatır.

CAD Oturumu: Akıllı kartın, kart kabul cihazına (CAD) yerleştirilip işleme sokulmasından cihazdan çıkartılmasına kadar geçen periyot bir "CAD Oturumu" olarak adlandırılmaktadır. CAD oturumu sırasında JCRE tipik bir akıllı kart gibi davranıp APDU'ları almakta ve ev sahibi uygulama ile iletişime geçmektedir. Ortam çoklu-*"applet"* yapısında olduğundan ev sahibi uygulamaya bilgi gönderecek olan *"applet"* ilk komutta JCRE tarafından seçilir ve gelen komut *"applet"*e yönlendirilir. (Şekil 2.7)



Şekil 2.7 CAD oturumunda JCRE'nin rolü ve APDU I/O iletişimi (Chen, 2000)

Java Card Çalıştırma Ortamı Özellikleri: Yukarıda anlatılanlara ek olarak JCRE'nin desteklediği üç ek özellik şunlardır:

Kalıcı ve geçici nesneler: Varsayılan olarak, Java Card nesneleri kalıcı olarak yaratılıp EEPROM üzerinde saklanmaktadır. Ancak güvenlik ve performans nedenleri ile RAM üzerinde çalışan geçici nesneler de üretilebilir. Bu nesneler sadece bir CAD oturumunda yaşamakta olup oturum sonlandırıldığında değerleri kaybolur.

Atomik işlemler: Java Card sanal makinesi bir nesnenin herhangi bir özelliğinin (attribute) yazılmasının atomik olarak gerçekleştirilmesini garanti eder. Aynı zamanda bir işlemler dizisi de JCRE kullanılarak istendiğinde atomik hale getirilir. Bu durumda ya tüm işlemler yerine getirilir ya da - tüm işlem kümesi sırasında bir hata ile karşılaşıldığında - hiç biri işletilmez.

"Applet" kalkanı ve paylaşma mekanizması: "Applet" kalkanı her bir *"applet"*in izole edilmiş bir ortamda çalışmasını ve kart üzerindeki *"applet"*lerin birbirlerini etkilememesini sağlamaktadır. İki *"applet"* arasında iletişim ve veri paylaşımı gerektiğinde ise sanal makine, güvenli paylaşım mekanizması ile buna izin verir.

Java Card Çatısı:

Java Card Çatısı (JCF), Java akıllı kartı üzerinde çalışacak uygulamaların ISO 7816 standardına uygun olarak hazırlanmasında kullanılacak olan Java paketlerinin ve sınıflarının içerisinde bulunduğu bir çatıdır.

Java Card API topluluğu bünyesinde yer alan üç çekirdek ve bir ek paket şunlardır:

- `java.lang`: Temel Java programlama dili desteğini sağlar.
- `javacard.framework`: Bir Java Card *"applet"*inin çekirdek fonksiyonları için gerekli olan sınıfları ve ara yüzleri içerir.
- `javacard.security`: Java Card platformunda desteklenen kriptografik fonksiyonların uygulama geliştirmede kullanılmasını sağlar.
- `javacardx.crypto`: Veri şifreleme ve çözümlemede kullanılan *Cipher* ayrık sınıfı gibi ek kriptografik sınıf ve ara yüzleri içerir.

2.1.2.3 Java Card *"applet"*leri:

Bir Java Card *"applet"*i daha önce de belirtildiği gibi Java Card Çalıştırma Ortamı'nda çalışan bir Java programıdır.

Java Card *"applet"*lerini, masaüstü bilgisayarlarında ağ tarayıcıları üzerinde çalışan Java uygulamaları ile karıştırmamak gerekir. Bir Java Card *"applet"*i, bir Java akıllı kartı üzerinde çalışmak üzere Java Card Çatısı kullanılarak hazırlanmış olan uygulamadır. Bir Java nesnesi olarak hazırlanan kart *"applet"*i, `javacard.framework.Applet` sınıfının uzantısıdır (*extend* ilişkisi).

Paket ve "applet" İsimlendirme:

Java Card platformunda her *"applet"* nesnesi tekil bir *AID (Application Identifier)* ile tanımlanmakta ve seçilmektedir. Benzer şekilde kart üzerinde yer alan paketlere de böyle bir AID atanmakta ve paketlerin kart üzerinde birbirlerine bağlanması (linking) bu AID ile gerçekleşmektedir.

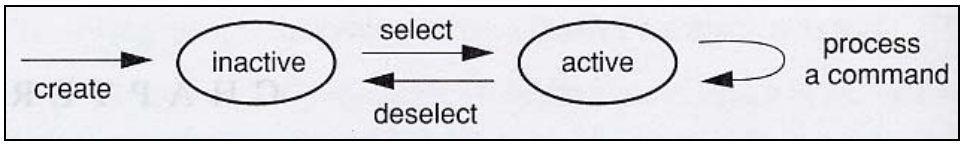
ISO 7816 tarafından tanımlanan AID bir bayt dizisi olup iki kısımdan oluşmaktadır: 5 baytlık ilk bölüm *RID (Resource Identifier)* adı verilen ve uygulamayı hazırlayan kuruluşu belirten on altılı (hex) koddur. Sonraki kısım ise değişken uzunluklu (0-11 bayt arası) *PIX (Proprietary Identifier Extension)* adı verilen ve uygulamayı hazırlayan kuruluşun kendisinin belirlediği bir koddur. Buna göre kart içinde bir uygulama, uzunluğu 5 ila 16 bayt arasında değişen AID'si ile tanımlanır.

"Applet" yükleme ve çalıştırma:

CAP formatında karta yüklenen ve diğer paketler ile bağlantıları hazırlanan bir Java Card *"applet"*inin kart üzerindeki hayatı, *"applet"*in JCRE tarafından bir örneğinin yaratılması ve kaydedilmesi (register) ile

başlar. Yükleme adı verilen bu işlem sonunda uygulama, çağrılara ve kullanılmaya hazırdır.

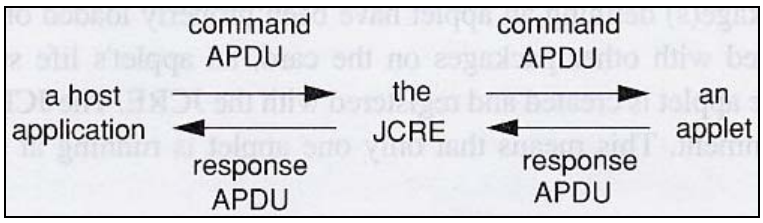
JCRE'nin tek iş parçacıklı (single-threaded) bir ortam olmasından dolayı belli bir zamanda sadece bir tek uygulama çalışmaktadır. Uygulama, yüklendikten sonra aktif durumda değildir; terminal uygulaması tarafından seçildiğinde aktif duruma geçer. Kart üzerinde yeni bir uygulama seçimi ya da kartın okuyucudan çıkarılmasına kadar uygulama ile ev sahibi uygulama arasında APDU iletişimi devam etmektedir. (Şekil 2.8)



Şekil 2.8 "Applet" işletim durumları (Chen, 2000)

"Applet" İletişimi:

Bir *"applet"* ile ev sahibi uygulama arasındaki iletişim APDU paketleri ile sağlanmaktadır. Bir ev sahibi uygulama, kart üzerindeki bir uygulamanın seçilip çalıştırılmasını sağlamak için ilgili uygulamanın AID'sini ve SELECT komutunu içeren APDU'yu karta gönderir. JCRE kendi tablosuna bakarak belirtilen AID'ye sahip uygulamayı çalışmak üzere aktif duruma geçirir. Terminalden gelen bu SELECT APDU'su ve diğer tüm komut APDU'ları seçili olan uygulamaya iletilirler. İletişim yeni bir akıllı kart uygulaması seçilinceye ya da kart oturumu sonlandırılıncaya kadar devam eder. (Şekil 2.9) (Chen, 2000; Hansmann et al., 2000)



Şekil 2.9 "Applet" iletişimi (Chen, 2000)

2.1.3 OpenCard Çatısı

OpenCard Framework (OCF), Java programlama dili kullanılarak hazırlanmış bir akıllı kart özel yazılımıdır (middleware). IBM önderliğinde dünyanın diğer önde gelen akıllı kart üretici kuruluşlarının oluşturduğu *OpenCard Konsorsiyumu* tarafından geliştirilmiştir.

Mimaride OCF, bir akıllı kart ev sahibi uygulaması ile kart okuyucu (CAD) aygıtı arasında yer almaktadır. Çok esnek bir altyapısının olması ve Java dilinde hazırlanmasından kaynaklanan platform bağımsızlığı, akıllı kart uygulama geliştiricilerinin yüksek seviyeli programlama ara yüzlerini kullanabilmelerine ve farklı kart üreticilerine ait kart okuyucular ile uğraşmaya gerek kalmadan uygulama hazırlamalarına imkan sağlar.

Tez projesinde akıllı kart terminalleri üzerinde çalışan uygulamalar ile bu terminallere bağlı kart okuyucular (dolayısıyla akıllı kartlar) arasında iletişim bu çatıdan faydalanılarak gerçekleştirilmiştir. Kullanılan OCF sürümü 1.2 olup bu sürümle ilgili özet bilgiler aşağıda yer almaktadır.

2.1.3.1 OCF mimarisi

OCF, farklı tipteki akıllı kartlar ile iletişime geçmek ve uygulamalarda kullanabilmek amacıyla iki katmanlı bir yapıdan

oluşmaktadır. Bu katmanlar *CardTerminal* katmanı ve *CardService* katmanıdır. Şekil 2.10'da görülen bu katmanların nesneye dayalı olarak tasarımında *ayrık fabrika (abstract factory)* ve *tek örnek (singleton)* tasarım desenleri kullanılmıştır. Kart üreticisine has detayları içeren nesneler ilgili üreticiye ait fabrika nesneleri tarafından üretilmekte; hangi fabrika nesnesinin kullanılacağı ise tek örnek yapıdaki ilgili `registry` nesnesi tarafından belirlenmektedir. Bu ve benzeri diğer sistem tasarım desenleri tez kapsamında hazırlanan sistemin mimarisinde de yer almakta olup Gamma et al. 'de (1994) tanımlanmışlardır.

CardTerminal Katmanı:

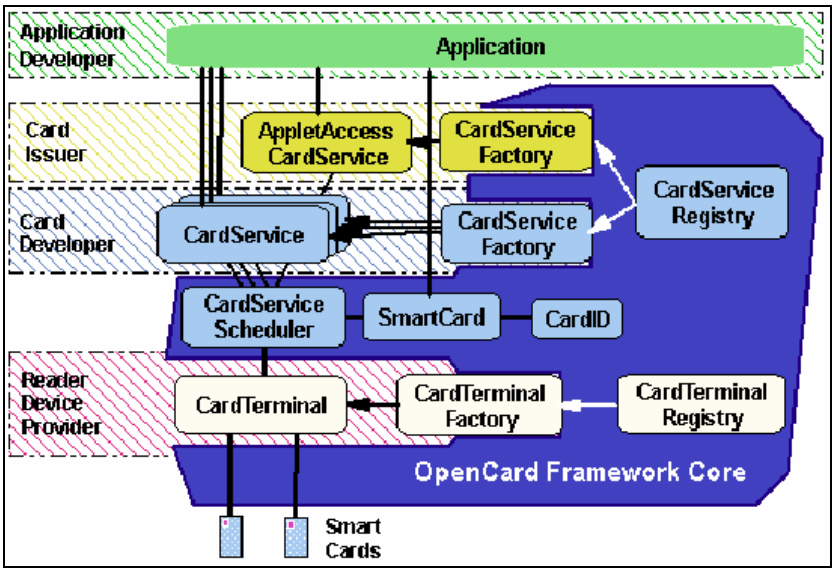
Bu katman uygulama geliştiricilerinin kart okuyucu yazıcı ünitelerine ve bu üniteler üzerindeki yuvalara erişmesini sağlamaktadır. OCF'de; fiziksel her bir CAD, üzerindeki yuvalar ile beraber `CardTerminal` sınıfından türetilen bir nesne olarak temsil edilmektedir. `CardTerminalFactory` oluşturduğu `CardTerminal` nesnelerinin kaydını tutmaktadır.

`CardTerminal` sınıfı; bir akıllı kartın CAD'e yerleştirilmesi veya çıkarılması gibi olayları üretmekte ve bu olaylar (CTEvents) `CardTerminalRegistry` aracılığı ile `EventGenerator` nesnesine aktarılmaktadır. `EventGenerator`'a kaydolan her bir olay dinleyici, oluşan bu kart terminali olayından haberdar edilmektedir (Şekil 2.11). Tezde yer alan sistemde de CAD ile iletişime geçen uygulama ara yüzü bileşenleri `EventGenerator`'a kayıtlı birer olay dinleyici olarak tasarlanmışlardır.

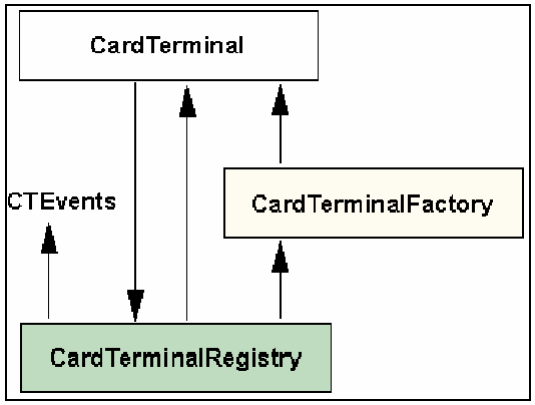
CardService Katmanı:

OCF, uygulama geliştiricilere akıllı kart fonksiyonlarını kart servisleri aracılığı ile kullanma imkanı sunmaktadır. Bu fonksiyonları yerine getirmede CardService katmanında tanımlanan ve `CardService`

adı verilen soyut (abstract) sınıflardan yaralanılmaktadır. Her bir sınıf, spesifik akıllı kart fonksiyonu sunan API tanımlar.



Şekil 2.10 OCF mimarisi bileşenleri (OpenCard Consortium, 1999)



Şekil 2.11 CardTerminal katmanı (OpenCard Consortium, 1999)

Tez projesinde akıllı kart üzerinde *"applet"* seçimi, kriptografik anahtar değişimi ve APDU gönderimi gibi, detayları daha sonra verilecek servisler kullanılmıştır. (OpenCard Consortium, 1999; Hansmann, 2000)

2.1.4 Java RMI

Altyapı içeriği olarak buraya kadar hazırlanan bölümlerde, tez projesinin akıllı kart bileşenleri ile ilgili olan teknolojiler hakkında bilgi verilmiştir. Bu bölümde ise hazırlanan sağlık sisteminin dağıtık mimari özelliğinin temelini oluşturan *Java RMI (Remote Method Invocation)* protokolü hakkında bilgiler yer almaktadır.

2.1.4.1 Uzak nesne kavramı

Nesneye dayalı analiz ve tasarım metodolojilerinin popülerleşmesi ve buna dayalı olarak ortaya çıkan nesne tabanlı bilgisayar yazılımlarının farklı makineler üzerinde haberleşmesi ihtiyacı uzak nesnelerin ortaya çıkmasına neden olmuştur. Hedeflenen; farklı makineler üzerinde çalışan nesnelerin birbirleri ile *doğrudan* mesajlaşmasını sağlamaktır.

Yapısal sistemlerde dağıtık süreçler arasında kullanılan RPC (Remote Procedure Call) mekanizmasına benzer şekilde standart bilgisayar ağı protokolleri (örneğin TCP/IP) üzerinden nesnelerin birbirleri ile haberleşmesi düşünülmüş ve buna dayalı mimariler gerçekleştirilmiştir. Bunlardan biri olan CORBA (Common Object Request Broker Architecture) adı verilen, OMG (Object Management Group) tarafından ortak veri iletişimi ve servis tanımlamalarının verildiği bir standarttır. Nesne tabanlı farklı programlama dilleri ile hazırlanmış olan nesnelerin birbirleri ile iletişimine imkan verir. Örneğin Java kullanılarak hazırlanmış bir istemci nesnenin başka bir makinede yer alan ve C++ kullanılarak hazırlanmış bir sunucu nesneden hizmet alması mümkündür. Ancak performansının yavaş olması, karmaşık gerçekleştirimlere ihtiyaç duyması ve birlikte işlerlik (interoperability) problemlerinin olması CORBA'nın dezavantajlarıdır. Uzak nense haberleşmesinde kullanılan Java RMI ise sadece Java nesnelerinin kullanımı göz önüne alındığında anlaşılması daha kolay ve kullanımı daha uygun bir yapıdadır. Tez projesinde n-katmanlı mimari içerisinde hem istemci hem de sunucu bileşenlerin Java platformunda

hazırlanmasından dolayı Java RMI, üst seviye protokolü olarak kullanılmıştır.

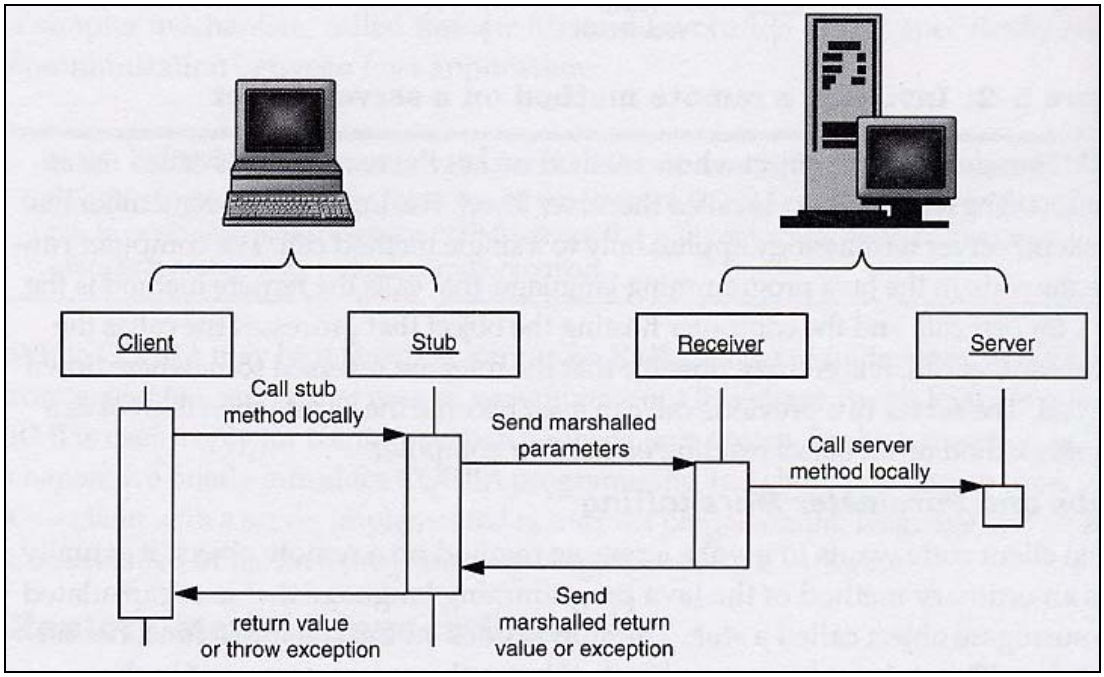
2.1.4.2 Java RMI mimarisi

Java RMI, Java teknolojisine bağlı dağıtık uygulamaların hazırlanmasına; uzak Java nesnelerine ait metotların ağ üzerindeki diğer Java sanal makinelerince uyandırılıp kullanılmasına olanak sağlar. Uzak nesneye ait metodun çağırılabilmesi için çağıran programın ilgili nesnenin ara yüzüne bir referansının olması gerekir.

RMI, parametrelerin ağ üzerinde belirlenen formatta gönderilmesi için kodlanmasında (parameter marshalling) ve bunların geri elde edilmesinde (parameter unmarshalling), nesne serileştirmeyi (object serialization) kullanmaktadır.

İstemci bir uygulama uzak bir nesnenin metodunu kullanmak istediğinde *"stub"* adı verilen vekil bir nesnenin kapsadığı metodu sıradan bir Java nesnesinin metodunu kullanmak istermiş gibi çağırmaktadır. İstemci makinede yer alan *"stub"*; çağırımda kullanılacak olan parametreleri bir bayt bloğu olarak hazırlamakta ve kodlamaktadır. *Parameter Marshalling* adı verilen bu kodlama işlemi, verilerin bir sanal makineden diğer bir sanal makineye belli bir format altında transferini sağlar.

"Stub" nesnenin hazırladığı blokta, çağırılacak metodun tanımı ve uzak nesnenin belirteci (identifier) de yer alır. Sunucu tarafında görev yapan *alıcı nesne* taşıma katmanından kendisine ulaşan bloğu açar ve çağrılan nesneyi belirleyerek bu nesnenin ilgili metodunu çağırır. İşlem sonucunu benzer şekilde kodlayarak oluşan paketi istemcideki *"stub"* nesnesine gönderir. Ağ üzerinde transfer, varsayılan olarak TCP/IP üzerinden gerçekleşmesine karşın esnek yapıdan dolayı UDP'nin de kullanılması imkanı vardır. (Şekil 2.12)



Şekil 2.12 RMI mimarisinde nesnelerin birbirleri ile olan iletişimi ve parametre kodlama (Hortsmann & Cornell, 2000)

Bu sürecin karmaşık olduğu ortadadır. Ancak mimaride bu prosedürler tamamıyla otomatik işlemekte; böylelikle uygulama geliştiricilerin bu yapılardan çok protokol üzerinde çalışan sistemleri hazırlama ile uğraşmalarına imkan tanınmaktadır.

RMI İsimlendirme Servisi:

İstemcilerin uzaktaki nesnelerin metotlarını RMI ile çağrılabilmeleri için, söz konusu uzak nesnelere ilişkin referansları elde etmesi gerekmektedir. RMI modelinde bu amaca yönelik olarak bir isim sunucu (name server) kullanılmaktadır. Uzak nesneler `java.rmi.Naming` isimli bir sınıfta bulunan `bind()` metodunu kullanarak kendilerini İsim Kayıtları (Naming Registry) servisine kaydettirmelidirler. İstemciler de `java.rmi.Naming` isimli sınıfın `lookup()` isimli metodunu kullanarak

uzak nesnelerin hangileri olduğuna bakıp istenen uzak nesneye bir referans elde edebilmektedir.

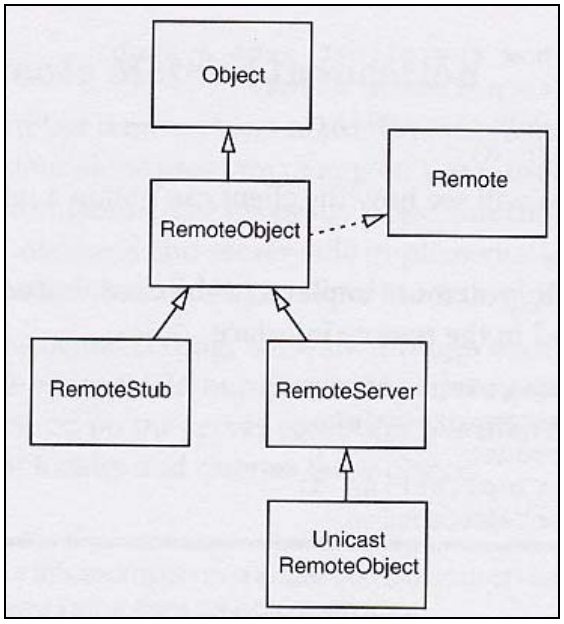
2.1.4.3 RMI sisteminin hazırlanması

RMI tabanlı bir sistemin hazırlanmasında öncelikle uzak nesneye ait ara yüzün tasarlanıp hayata geçirilmesi daha sonra bu ara yüzü uygulayan sınıfın hazırlanması gerekir. Bu sınıf *rmic* adı verilen araca (tool) parametre olarak verildiğinde gerekli *"stub"* kodu otomatik olarak hazırlanır. (RPC'ye; *rpcgen*'e protokol tanımlama dosyasının verilmesi ile ilgili *"stub"* ve iskelet (skeleton) kodlarının oluşması açısından benzerlik göstermektedir.)

Şekil 2.13'de verilen nesne kalıtım diyagramında görüldüğü gibi sunucu sınıfının, `java.rmi.server` paketinde yer alan `RemoteServer` sınıfının bir uzantısı olması gerekmektedir. Ancak soyut bir sınıf olan `RemoteServer`'in, sadece sunucu nesneleri ile uzak *"stub"*lar arasındaki iletişim mekanizmasını tanımlamasından dolayı uzak nesne çağırımlarında sunucu olacak nesneler, RMI paketi ile gelen ve `RemoteServer` sınıfının bir uzantısı olan `UnicastRemoteObject` sınıfının bir uzantısı olarak tasarlanır.

Uzak nesneleri türeten RMI sunucusu, isimlendirme kayıtçısına bu nesneleri bağlar. İstemci uygulama ise bu kayıtlanmış nesnelerin ara yüzünü kullanarak uzak metot çağırımını gerçekleştirir. (Horstmann & Cornell, 2000; Danny et al., 1999)

Tez projesinde, doktor muayenehanelerinde yer alan akıllı kart ev sahibi uygulamaları belli işlemlerde veritabanı erişimine ihtiyaç duymaktadır. Bu veritabanı istemcileri doğrudan veritabanlarına bağlanmak yerine bu hizmeti, veritabanı bağlantılarını kendileri için gerçekleştiren ve ilgili sorguları yerine getiren RMI nesnelerinden almaktadırlar. Bu yapıya neden ihtiyaç duyulduğu ve mimari ilerleyen bölümlerde yer almaktadır.



Şekil 2.13 RMI uzak nesne kalıtım diyagramı

2.2 İlgili Çalışmalar

Sağlık sistemlerinin hazırlanmasında hastalara kaliteli ve hızlı bir hizmetin sunulması, sistem geneli sağlık verilerinin güvenliğinin sağlanması, kayıtların kolay olarak sisteme aktarılması ve geri getirimi gibi özellikler ön plana çıkmaktadır. Bu özellikleri en iyi şekilde sunabilecek çözümlerde akıllı kartların yer alması doğaldır. Çünkü akıllı kartların tasarımlarından kaynaklanan özellikleri bu kartların sağlık sektöründe kullanılmasına imkan vermektedir. Gelişmiş kapasiteleri ve işlem güçleri ihtiyaç duyulan bilgilerin mobil ve hızlı bir şekilde elde edilmesini sağlamaktadır. İçerdikleri güvenlik özellikleri ise sağlık kayıtlarının güvenli olarak saklanmasını ve sadece yetkilendirilmiş kişilerin erişimini sağlar.

Sağlık sektöründe akıllı kartların kullanılması ile ilgili olarak, başta Fransa ve Almanya olmak üzere birçok Avrupa ülkesinde ve ABD' de, tamamlanan veya geliştirilme aşamasında olan bir çok çalışma

bulunmaktadır. Bu bölümde bu çalışmalar ile ilgili genel bilgiler verilmiştir.

2.2.1 Sesam Vitale

Sağlık sektöründe tamamlanmış en kapsamlı projelerden biri Fransa'da kullanılmakta olan *Sesam Vitale* projesidir. Akıllı kart tabanlı bu sistemin amacı, yazılı1 milyar tedavi formunun yerine elektronik formları kullanmak; hasta, tıp uzmanı ve sağlık sigorta organizasyonları arasındaki ilişkileri yüksek güvenlikli bir ortamda kolaylaştırmak; hastaya yüksek kalitede sağlık hizmeti sunmak; tıbbi verilerin daha güvenilir bir ortamda tutulması ile bu bilgilerin daha kolay analizi ve sağlık giderlerinin kontrolünü sağlamaktır.

GIE SESAM-VITALE konsorsiyumu Fransa'da faaliyet gösteren, sektörde önemli konuma sahip sigorta kuruluşları tarafından 1993'de kurulmuştur. Bu proje aynı zamanda; Fransa, Almanya, İtalya ve Kanada'nın Quebec eyaletinin üyesi olduğu uluslararası NETLINK projesinin de bir parçasıdır.

Bu proje kapsamında ülke genelinde 40 milyondan fazla akıllı kart dağıtılmıştır. Doktorların %79'u veri iletimini bu sistem içerisinde yer alan hat üzerinden gerçekleştirmektedir.

Sesam Vitale sistemi, elektronik verilerin sağlık uzmanları ile sosyal sigorta organizasyonları arasında, mikroişlemci içeren Vitale kartları ile güvenli olarak alışverişinin gerçekleştirilmesine dayanmaktadır. İlk jenerasyon olan Vitale 1 kartları yönetimsel bilgi içeren ve 3-4 Kilobayt EEPROM'a sahip akıllı kartlardır. Haziran 1999 itibari ile 37 milyon adet Vitale 1 kartı hazırlanmıştır. İkinci jenerasyon Vitale 2 kartlarından 2003 senesi içerisinde 60 milyon tanesinin dağıtımı planlanmıştır. Bu kişisel kart hem yönetimsel (Vitale 1'dekilerinin üzerine ek sigorta verileri) hem de tıbbi bilgiler içerecektir. Kartlar 8 Kilobaytlık EEPROM'lara sahip olacaktır. (Pagetti el al., 2001)

2.2.2 Alman sağlık sigorta kartı ve DENTcard

Almanya da, Fransa gibi akıllı kartların sağlık sektöründe kullanılmasında öncülük etmektedir. Almanya Sağlık Sigorta Kartı Sistemi'nde, sağlık sigorta kuruluşlarına üye herkesin bir akıllı kartı bulunmaktadır. Bu kart ile hastalar kendilerini doktorlara tanıtmaktadır. Kartın arka kısmında ise imzaları yer alır. Önceki sistemde hastalara ait basılı sağlık sigorta biletleri yerine de bu kartlar geçmiştir. Sistem bünyesinde 80 milyon kart dağıtılmıştır.

Hastanın temel bilgileri elektronik ortamda hatasız olarak okunup işlenmekte ve birçok standart dijital forma basılmaktadır. Verilerin elektronik formlarda hızlı bir şekilde taranıp işlenmesi hem zamandan hem de kağıt giderlerinden büyük tasarruf sağlanmasına neden olmaktadır. Ayrıca eski sistemde yer alan sağlık sigorta biletlerinin posta ile gönderilmemesi de ek posta ve basım maliyetlerinden kurtulmayı sağlamıştır. Kartlar üzerinde tıbbi bilgiler bulunmamakta sadece sistem yönetim bilgileri yer almaktadır.

Almanya'da diş hastalarının kullanılması için tasarlanan ilk akıllı kart sistemi DENTcard'dır. Bu sistem, hastaların doktorların ve hatta laborantların diş tedavisi alanında üstlendikleri işlevleri bütünleştirmektedir. Sistem, diş tedavisinde ihtiyaç duyulan teşhis, terapi ve geri çağırma isteklerini karşılama amacıyla modüler bir yapıda tasarlanmıştır. DENTcard, Built Informationstechnologie AG, DENTcard AG ve ORGA Kartensysteme GmbH kuruluşları ile yapılan ortak çalışmalar sonucunda geliştirilmiştir. Ortaklar 1996'da bir test market kurmuş ve 1998'e kadar desteklemişlerdir. Verilerin, halihazırda kullanılmakta olan diş hekimi yazılımlarından otomatik transferi, "Open Data" ve VDDS ara yüzlerinin entegrasyonu ile 1999'da planlanmıştır. (Pagetti el al., 2001)

2.2.3 CIS

CIS (la Carte d'Identite Sociale Belge), Belçika'da 12 milyon insanın kullandığı ve sağlık uzmanlarının kullandığı, bilgisayar ağına bağlı 35000 terminal üzerinde çalışan bir sistemdir.

CIS, biri sağlık uzmanları biri de vatandaşlar için olmak üzere iki akıllı kart sistemi üzerinde çalışmaktadır. Hasta kartı 1 Kilobayt bellekli ve PIN korumalı olup hastaya ait bilgiler şifreli olarak kart üzerinde tutulmaktadır. Bu bilgilere erişim ve bilgileri okuma ise ancak bir sağlık uzmanı kartı ile olabilmektedir. İkinci tip kartlar da 1 Kb belleğe sahiptir ve özel sağlık sigorta şemasına uygun bilgiler kart içinde yer alır. (Pagetti el al., 2001)

2.2.4 ISHTAR

1 Şubat 1996'da başlatılan ve 31 Ekim 1998'de sona eren projede katılımcı olarak İngiltere, Hollanda, İrlanda, Fransa, Belçika, Portekiz, Yunanistan, İsviçre, Finlandiya, İtalya ve Almanya yer almıştır. Projenin amacı hukuk, tıp ve teknoloji uzmanlarının bir araya geldiği bir grup oluşturmak ve veri koruma yönergelerini sağlamaktır. (Pagetti el al., 2001)

2.2.5 DIABCARD

1998'de üçüncü aşaması bitirilen projede şeker hastalarına yönelik bir akıllı kart geliştirilmiştir. Kartın bir parçası olduğu tıp bilgi sisteminin pilot uygulaması, 3 ay içerisinde inşa edilmiş ve değerlendirilmiştir. DIABCARD farklı ihtiyaçlar için esnek ve uygundur. Mimarisi halihazırda yer alan bilgi sistemleri ve bilgisayar ağı ortamlarında uygulanabilir. Ayrıca teknolojik ilerlemelerle paralel olarak daha fazla özellik kazanması mümkündür. İlk prototip Barselona'da test edilmiştir: Prototip "Hospital de la Santa Creu i Sant Pau" hastanesinin veri kayıt

sisteminde kullanılmıştır. İlk gerçekleştirim 1 aylık periyotta denenirken ikinci gerçekleştirim 3 aylık bir sürede denenmiş ve endokrinoloji, obstetriks ve nefroloji ünitelerini içermiştir. Uygulamada 108 hasta yer almıştır. Bunların bir kısmı DIABCARD kullanan deneysel grup içerisinde yer alırken diğerleri geleneksel kağıt dosya kullanan kontrol grubundadır. Hasta her doktora gidişinde hastaya ait bilgiler güncellenmektedir. Doktorlar bu durumdan pek memnun olmamış ve işlevsel problemlerle karşılaştıklarını belirtmişlerdir. Öte yandan hastalar projeden oldukça memnun kalmışlardır.

Diğer prototip ise Perugia'da test edilmiştir: Bu deneysel aşama yersiz beklentilerin ve servis kalitesi hakkındaki erken reaksiyonların önlenmesi amacıyla gizli tutulmuştur. Hedeflenen, DIABCARD ile Millenium adı verilen yazılım arasında bağlantı kurmak olmuştur. Millenium sistemini kullanan 5 doktor çalışmalarda yer almış ve bu doktorların hastalarına ait kayıtlar hazırlanmıştır. İkinci aşamada DIABCARD'ların oluşturulması ve hastalara dağıtılması yer almıştır. Hastalar her doktora gidişlerinde kart üzerindeki bilgilerini Milennium sistemine transfer etmişlerdir. Sonuç olarak hem doktorların hem de hastaların sisteme reaksiyonları olumlu olmuştur.

Üçüncü prototip ise Atina'da test edilmiştir: Barselona'da gerçekleştirilen modelin aynısı kullanılmış ancak kart yapısı Yunan diline ve çalışma metotlarına uygun olarak değiştirilmiştir. 15 tıp uzmanı ve 100 hasta sistemde yer almıştır. Denemeler sırasında ve sonucunda doktorlar ve hastalar, sistem sonuçları, sistemin günlük rutinlere etkisi ve kabul edilebilirliği hakkındaki anketi doldurmuşlardır. Test "Kızılhaç Hastanesi"ne de taşınmış ve 3 ayda sonlanmıştır. Projeden hem doktorlar hem de hastalar memnun kalmışlardır. (Pagetti el al., 2001)

2.2.6 CARDLINK 2

Mart 1996'da başlayan ve Mart 1999'da sona eren projede asıl amaç acil durumlarda kullanılmak üzere bir hasta kartı tasarlamak ve gerçekleştirimini sağlamaktır. Proje 9 Avrupa ülkesinden 10 bölgeyi

kapsamaktadır. Kart bir güvenlik sistemi içerisinde hasta bilgilerini içermekte ve genel pratisyenler, uzmanlar, eczacılar ve yönetici memurlar tarafından okunabilmekte ve güncellenebilmektedir. Kart ayrıca hastane veritabanlarına girmede kullanılacak imleçleri de içermektedir.Veri mimarisi Eurocards projesinde yer alan kart veri önerilerine uygundur. Bahsedilen 10 Avrupa bölgesinde sistemin gerçekleştirimi, yönetimsel ve acil bilgilerle ile etkileşimli çalışma prensibini esas almıştır. Proje, daha önceki "Cardlink 1" uygulamasının sonuçlarının genişletilmesini ve Eurocards projeleri üzerindeki araştırmalar için bir test alanı sağlamayı hedeflemiştir. Projede 100000 adet kart kullanılmış ve servis sağlayıcıların ve kullanıcıların detaylı değerlendirmeleri ortaya konmuştur. Projede, İrlanda (Dublin), Almanya (Koblenz), Hollanda (Delft), İspanya(Valencia ve Madrid), Yunanistan (Atina Uluslararası Teknik Üniversitesi), Portekiz (Lizbon), Fransa(St. Nazaire), İtalya(Roma) ve Finlandiya yer almıştır. (Pagetti el al., 2001)

2.2.7 TRUSTHEALTH

Nisan 1998'de başlayan ve Nisan 2000' de sona eren projede amaçlanan, güvenlik servisleri ve ara yüzleri hakkında çeşitli tanımlamalar ve servis altyapıları sağlamaktır. Projede Belçika, Fransa, Almanya, Norveç, İsveç ve İngiltere yer almıştır. Bu 6 ülkede de güvenlik servisleri işler halde olup bu servisler, laboratuar sonuçları için tıbbi mesaj iletişimi, ilaç reçetesi, dijital imza ve kullanıcı doğrulanması gibi uygulamalarda sürdürülmektedir. Sağlık alanındaki ileriye yönelik üç beklenti projenin servisler ve ara yüzler üzerindeki işleyişini etkilemiştir: Kişisel bilgisayarlar en fazla kullanılan terminaller olup gelecekte de varolacaklardır; sağlık uzmanları yer değiştirebilir konumdadır; veri kaydetmede anahtar ve sertifika kontrollerinin akıllı kartlar aracılığıyla yapılmasının en uygun çözüm olacağı açıktır. Proje üç doküman ortaya koymuştur:

- Güvenlik servisleri ve ara yüzlerinin seçimi. Güvenlik servisleri için işlevsel ihtiyaçlar
- Güvenlik servisleri ve ara yüzlerinin gerçekleştirimi için yönergeler

- Servislerin işlevsel tanımlamaları. Servislerin işlevsel tanımlamaları ad, genel anahtar ve profesyonel sertifika, kart basımı ve yön servislerini içermektedir. Her kullanıcının bir tanımlama numarası vardır. Bu numaralar genel bir kayıt otoritesi tarafından kaydedilir. Daha sonra her kullanıcı, bir anahtar üretme mekanizmasından 3 tip anahtar alır: Dijital imza, şifreleme ve tanımlama ve doğrulama anahtarları. Bundan başka her kullanıcı her anahtar ile bağlantılı genel anahtar sertifikaları ve elektronik uzmanlık sertifikası alır. (Pagetti el al., 2001)

2.2.8 G-8 Sağlık bilgi kartı projesi

Avrupa Komisyonu'nun G-8 üyeleri, uluslar arası bir ortaklığın ürünü olacak bir projenin belirlenmesi için 1995 Şubat'ında Brüksel'de bir araya geldiler. Belirlenen projenin resmi adı "Sağlık Alanında Veri Kartlarının Uluslararası Kullanımı" idi. Proje:

- Acil bir durumda elzem tıbbi bilgileri sağlayacak olan uluslararası bir acil durum akıllı kartının
- Uluslar arası yönetimsel bir veri kümesinin
- Tıbbi verilere ve bilgisayar ağı servislerine erişimde güvenli tanımlamayı yerine getirecek bir sağlık uzmanı kartının

tasarımını içermektedir. (Pagetti el al., 2001)

2.2.9 TRANSCARDS

Avrupa Komisyonu destekli, Fransa-Belçika ortak projesi olan TRANSCARDS Mayıs 2000'den beri uygulamada olan bir sistemdir. Sistem, 100000 Fransız ve 50000 Belçikalı hastanın "Thierache" bölgesindeki 8 hastanedeki sağlık ve ilk yardım hizmetlerine erişimini sağlamaktadır. Hedeflenen; bu sınır bölgesinde sağlık hizmeti erişiminde

yer alan yönetimsel prosedürlerin basitleştirilmesidir. Sistem kapsamında sağlık hizmetlerine erişimde kullanılan yönetimsel kağıt-formlar yerine *Vitale* (Fransız) ve *CIS* (Belçika) sosyal güvenlik kartları kullanılmaktadır.

Bir Fransız hastanın bir Belçika hastanesinde tedavi görme senaryosu bu sistemde aşağıdaki gibidir:

1. Fransa'daki hastanın doktoru Thierache bölgesinde bir Belçika hastanesinde de tedavi görebileceğini hastaya bildirir.
2. Hasta, sağlık sigorta kartını ilgili bir terminalden Transcards sisteminden faydalanacak bir şekilde günler (Geçerliliği bir yıldır).
3. Hasta bu kartı ile bir Belçika hastanesine başvurur.
4. Hastane, hastanın kimliğini kontrol eder ve hastanın Vitale kartını okur.
5. Eğer kart Transcards bilgisi içermiyorsa hasta gittiği bu hastane içindeki terminallerden de kartını günleyebilir.
6. Tedavide kullanılan E112T formu karttaki bilgilere dayalı olarak ekrana getirilir ve hasta tedavisi gerçekleştirilir.

Benzer senaryo Belçikalı bir hastanın bir Fransız hastanesinde tedavi görmesinde de geçerlidir.

Haziran 2001'den beri proje ortakları deneysel programa devam etmekte ve sistemin kalıcı olması için gereken değerlendirmeleri yerine getirmektedirler. (Transcards)

2.2.10 NETLINK

1998 Haziran'ında başlayan ve Haziran 2000'de sonlanan Netlink, Avrupa Birliği'nin IV. Framework Ar-Ge projesidir. Proje katılımcıları Fransa, Almanya, İtalya ve Kanada'nın Quebec eyaletidir. Projenin amacı, ülke geneli sağlık bilgi sistemleri arasında, hastalar, tıp uzmanları, hükümetler ve sağlık sigorta şirketleri yararına olacak bir etkileşim sağlamaktır. Bu ortak çalışmanın sağlanması için bilgisayar ağları ve akıllı kart temelli sağlık sistemlerinin işlevsel özellikleri ve teknik yapıları ile ilgili tavsiyeler bu projede ortaya konmuştur. Ortaya konan dokümanlar: Ortak çalışma için Netlink ihtiyaçları (versiyon 2.1) ve onaylanmış sistem tarifnamesidir (versiyon 2.1). Proje çözümleri, akıllı kart teknolojileri ve PKI' yı (Public Key Infrastructure) içermiştir. İtalya projesi kapsamında 130000 hasta ve 1000 sağlık uzmanı kartı dağıtılmıştır. Bu sistem içerisinde bankaların uzak işlem yönetim sistemleri aracılığı ile hasta sağlık kartları kullanılarak borç ödeme mekanizması geliştirilmiştir. Quebec (Kanada)' de geliştirilen sistemde medikal işlemlerin yürütüldüğü bilgisayar ağı veritabanı sistemi oluşturulmuştur. Almanya'da 1993 ve 1994 yılları arasında 80 milyon akıllı kart vatandaşlara dağıtılmıştı. Bu proje bünyesinde Almanya, PKI-anahtar erişimi, tanımlama ve dijital imza kartları üzerine çalışmalarda bulunmuştur. Fransa da ise 1998 ve 1999 yıllarında 42 milyon *Vitale* adlı sigorta kartı dağıtılmış böylelikle sistemde elektronik geri ödeme süresi 2 güne indirilmiş ve kötü kullanımlar minimum seviyeye indirilmiştir. (Netlink; Pagetti et al., 2001)

2.2.11 NETCARDS

NETCARDS projesinin amacı sık yer değiştiren Avrupa vatandaşları için elektronik servisler sunan, Avrupa bünyesinde bir ağ kurmaktır. Spesifik olarak, tıp uygulama prosedürlerinin hem daha basitleştirilmiş hem de yüksek kalite servisler sunar hale getirilmesi hedeflenmektedir. Turistler, öğrenciler, geçici işçiler ve diğer sık yer değiştiren vatandaşlar, Avrupa bünyesindeki hastanelerde taşıyacakları akıllı kartlar aracılığıyla hizmet alacaklar; tedavi prosedürü, bu kişisel

kartlardan bilgi okunması ve gerektiğinde güvenli sunuculara bağlanarak bilgi getirimi şeklinde basite indirgenecektir.

NETLINK çalışmasındaki tanımlamalara uygun olarak geliştirilmekte olan NETCARDS bünyesinde teknik bileşenler:

- Akıllı kartlar
- Çoklu kart uygulamaları ve kart okuyucular
- SSL (Secure Socket Layer), RSA (Rivest Shamir Adleman), veri şifreleme ve güvenli Internet işlemleri için sertifikalar gibi güvenlik seçenekleri bulunan Internet tarayıcıları içeren iş istasyonları
- Uzak doğrulama (remote authentication) için Intranet/Extranet sunucuları
- Lokal sağlık sigorta veritabanları
- Internet/Extranet/Intranet telekomünikasyonlarıdır

Projeden beklenen temel faydalar:

- Avrupa ülkeleri dahilinde seyahat eden kişilere geliştirilmiş sağlık servisi erişimi sunulması
- Avrupa ülkeleri arasındaki maliyet harcamaları kontrolünün optimum hale getirilmesi ve kağıt-formların işlem maliyetlerinin düşürülmesi
- Geniş anlamda Avrupa bünyesinde akıllı kartların ve güvenli IT ağ uygulamalarının beslenmesidir

A ve B olmak üzere iki aşamada yapılandırılan projede 2003 yılı sonunda A Aşaması'nın (Pazar Geçerliliği), 2005 yılı sonunda da B Aşaması'nın (Başlangıç Seviyesi Kurulum) tamamlanması planlanmaktadır. 2005-2010 yılları arasında da sistemin Avrupa geneli tam kurulumu hedeflenmektedir. (Netcards Project; Netcards)

2.2.12 Amerika Birleşik Devletleri'ndeki çalışmalar

1993'de ulusal sağlık sistemi kavramı ortaya konduğunda, bilgi taşımada ucuz maliyetleri nedeniyle manyetik kartlar seçilmişti. Ancak akıllı kartların fiyatının geçen süre zarfında düşmesi hem de özellikle güvenlik konusunda manyetik kartlarda herhangi bir ilerlemenin olmaması bu ülkede de sağlık alanında akıllı kartları popüler hale getirdi. Özellikle isteklerin geçerliliği, isteklerin karşılanması maliyetinin azaltılması, hasta kişisel gizliliğinin sağlanması ve hasta kayıtlarının doğru ve zamanında saklanması gibi akıllı kartlar ile sağlanacak unsurlar ABD'de akıllı kartların bu sektörde kullanımını arttırdı. (Pagetti et al., 2001; Anderson et al., 1997)

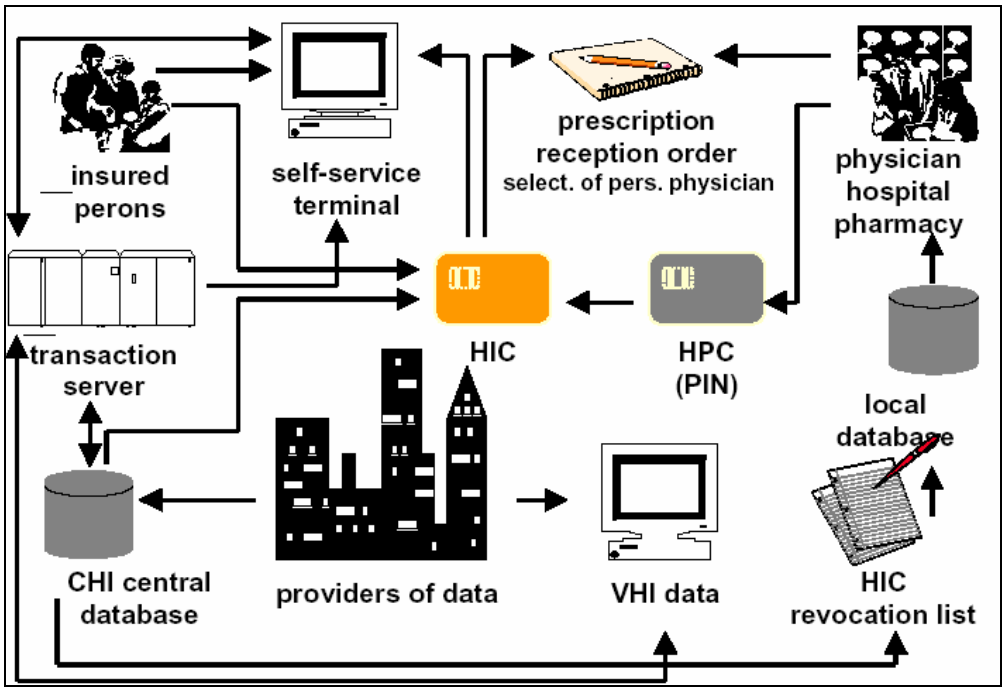
Halihazırda Texas eyaleti El Paso şehrinde kullanılmakta olan "Health Smart Card" sisteminde hastalara ait temel sağlık bilgileri (acil durum, medikal ve hastane bilgileri, alerjiler, teşhisler, ilaç tedavileri, aşılar ve sigorta bilgileri) satın aldıkları akıllı kartlar üzerinde yer almakta; sisteme dahil olan kliniklerde de bu kartları okuyucu ünitelerden bilgiler doktor ekranına getirilmektedir. Kart istemci bilgisayarları arasında herhangi bir ağ mevcut değildir. Kart üzerinde bilgi değişikliğine ihtiyaç duyulduğunda şehrin belli bölgelerinde bulunan terminallerde belli bir ücret karşılığı kart güncellemesi yerine getirilmektedir. (Health Smart Card)

2.2.13 Sloven akıllı kart ve IP tabanlı sağlık bilgi sistemi

Akıllı kartlara dayalı en kapsamlı ve tamamlanmış sağlık sistemi yakın zamanda Slovenya'da faaliyete geçen HIC (Health Insurance Card) sistemidir. Projenin asıl amacı sigorta ilişkili prosedürleri düzenlemek olmasına rağmen, sistem tüm sağlık sektörü için bir alt yapı sunacak şekilde tasarlanmıştır. Eylül 2000'de ilk aşaması kullanıma geçen sistemi iyileştirme çalışmaları devam etmekte ve iki yeni uygulama daha bu sistem kapsamında geliştirilmektedir. Aşağıda bu sistemin geçmişi, genel mimarisi, güvenlik özellikleri ve gelecekte gerçekleştirimi düşünülen ek özellikleri hakkında genel bilgiler yer almaktadır.

2.2.13.1 HIC projesinin geçmişi

1995 yılında HIIS (Health Insurance Institute of Slovenia – Slovenya Sağlık Sigorta Enstitüsü) Internet ve akıllı kart teknolojilerinin yer alacağı HIC projesini başlattı. Genel amaçlar; yönetimsel görevlerin en aza indirilmesi, sigortalı kişilere sunulacak servislerin kalitesinin arttırılması (prosedürlerin en aza indirilmesi, ilaç reçetesi yönetimi, medikal pasaport, vb.) ve kişisel veri güvenliğinin arttırılması idi. Öncelikli olarak detaylı kart tanıtım planı ortaya kondu: Tüm HIC sisteminin tasarımı (Şekil 2.14), HPC (Health Professional Card)' nin tasarımı, bilgisayar ağı ve self-servis terminallerinin tasarımı ortaya kondu.



Şekil 2.14 HIC sistemi: İçerdiği sistem üyeleri ve ilgili prosedürler (Novak et al., 2001)

İlk aşamada, kişi tanımada kullanılan ve sigorta durum bilgilerini içeren kağıt formlar, bu bilgileri içeren sağlık kartları ile değiştirilmiştir.

Sağlık sigorta bilgileri; hasta tanımlama, zorunlu ve gönüllü sağlık sigorta durumu ve hastaya özel doktor bilgilerini içermektedir. HIC sahibi kişilerin sigorta bilgilerini güncelleyebilecekleri self-servis terminaller hazır hale getirilmiştir. Bu terminaller aynı zamanda görevde olan doktor, en yakın eczane, vb bilgileri ileten bilgi kioskları olarak da hizmet etmekte olup HIIS'de yer alan merkez sunucu ile bağlantılıdırlar. Tüm sistem veritabanı bu sunucu üzerinde yer alır. Hatlar, standart protokollere dayalı, uygun genel anahtar ve simetrik anahtar kriptografisi ile korunmaktadır.

Doktor tarafında; bir hastaya ait HIC kartı bir HPC kartı ile aynı kart okuma ünitesine konur. Karşılıklı kimlik denetimi sonucunda uygun veriler erişime hazır hale gelir. Doktor bilgisayarları herhangi bir ağa bağlı olmayıp tek başlarına çalışmaktadırlar. Gemplus teknolojisine dayalı her iki kart da ISO7816-1..8 kriterlerine uygun olup HIC kartları 16 Kilobayt, HPC kartları ise 8 Kilobayt EEPROM'a sahiptir. Şu anda sistemde 1,946,000 HIC, 20,000 HPC, 5400 kart okuyucu ve 218 bölgede 270 self- servis terminali yer almaktadır. Sistem ilk aşamasının gerçekleştirimi daha önce de belirtildiği gibi Eylül 2000 itibari ile sona ermiş olup eczacıları ve elektronik reçeteyi içerecek ikinci aşamanın önümüzdeki 3 yıl içinde tamamlanması planlanmaktadır.

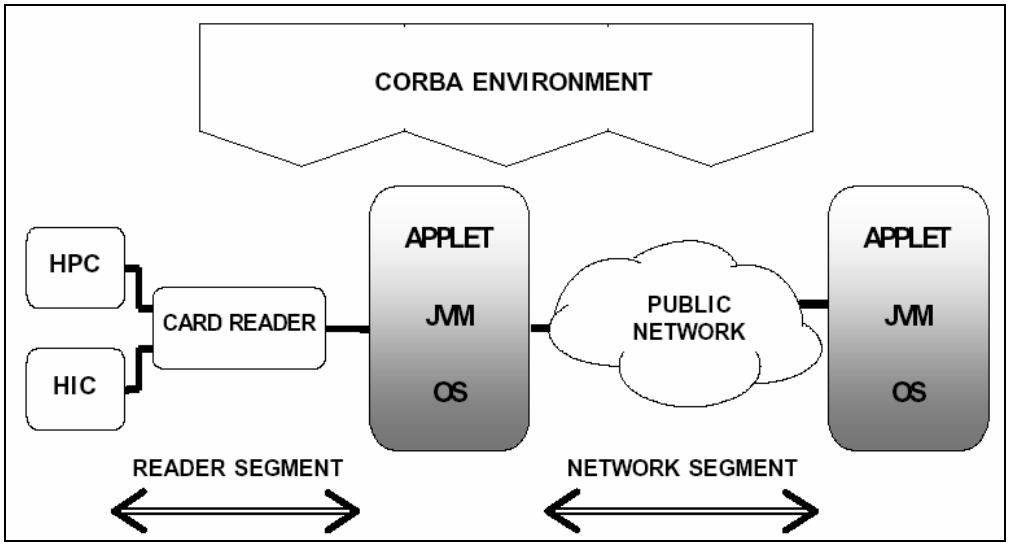
2.2.13.2 Genel mimari ve güvenlik özellikleri

HIC Projesi'nde, daha önce incelenen NETLINK çalışmasında ortaya konan sistem mimarisi ve kriterler benimsenmiştir. Ancak NETLINK'de; bilgi akışında S/MIME ve diğer bazı elektronik doküman formlarının kullanımını tavsiye edilmesine rağmen HIC projesinde DBMS (Database Management System) orijinli bir kullanım yer almaktadır. Sebep olarak medikal alanda oldukça az EDI (Electronic Document Interchange) tanımlamasına uygun kullanımın olması ve iş görevlerinin yerine getirilmesinde tamamlanmış prosedür eksikliği gösterilmektedir.

EDI' nin sistemde yer alması için veri kümelerinin çok iyi tanımlanması ve stabil olması gerekmektedir. Şu anda sağlık alanında bu şekilde tanımlı bir veri kümesi ve buna bağlı iş süreçleri yer almamaktadır. Ayrıca akıllı kartlar ve bunların birlikte işlerliği alanında hazırlanan temel dokümanlar ancak 4-5 yıllık olduğundan doküman saklama konusunda geleneksel veritabanı yaklaşımının seçilmesi benimsenmiştir. Ancak sistemin daha sonraki geliştirilme aşamalarında web teknolojilerinin ve içerik tanımlamada oldukça popüler olmaya başlayan XML (Extended Markup Language)'in entegrasyonunun da yer alması planlanmaktadır.

HIC projesinin en önemli özelliklerinden biri akıllı kartlara dayalı simetrik kriptografi kullanımının sistem mimarisinde yer almasıdır. Dijital imzalama işlemlerinde hasta kartları değil de sağlık uzmanı kartları kullanılmaktadır. Böylelikle ileriye yönelik sistem iyileştirmelerinde tüm sistemde yer alan kartların sadece %1'inin güncellenmesi söz konusudur. Temel hedef akıllı kartlar ile ağ üzerindeki istasyonlar arasındaki güvenliğin uçtan uca güvenlik kanallarının oluşturulması ile sağlanmasıdır. Oldukça gelişmiş kriptografik servislere sahip akıllı kartlar HIC projesinde yer almasına rağmen kriptografik protokollerin hatta genel anlamda iletişim protokollerinin etkin olarak işlenmesi bir sorun olarak görülmektedir. Bu amaçla CORBA uyumlu bir ortam sistem yapısında yer alır. Akıllı kartlar belli işlemlerin yerine getirilmesinde sistemde yer alan aracılar (agent) ile iletişime geçer. Bu aracılarla iletişime ancak HPC'ler geçebilir ve sadece HPC'ler asimetrik algoritmaları kullanırlar. Erişim uygulamalarına dayalı olarak işlem sonrası gerekli durumlarda HPC'nin HIC'deki ilgili veriyi günlemesi düşünülmektedir.

Protokol zinciri Şekli 2.15'te de görüldüğü gibi iki kısımdan oluşur. Okuyucu kısımda iletişim için "java.io" ve "java.net" paketleri kullanmak mümkün olmadığı için kullanılan karta özgü "javacardx.framework" de yer alan metotlar kullanılır. Alt seviyede iletişim her akıllı kart sisteminde olduğu gibi APDU paketleri ile yerine getirilmektedir. Bu açıdan zincirin bu kısmı en kritik kısım özelliğini taşımaktadır. İkinci kısım ise ağ kısmı olup sözü geçen "java.net" ve "java.io" paketlerini dolaysıyla soketleri kullanmak ve WWW (World Wide Web) sunucu ve istemcilerini bütünleşik güvenlik servisleri (Örnek: SSL) ile hazırlamak mümkündür.



Şekil 2.15 Uçtan uca protokol senaryosu ve HPC altyapısı (Novak et al., 2001)

Sistemde uygulanan kriptografik prosedürlerin yanı sıra HPC'de kimlik denetimini sağlamak amacıyla PIN koruması yer almaktadır.

2.2.13.3 Sistemin geliştirilmesi

SHIC sisteminin ileriye yönelik düşünülen entegrasyonunda; DBMS tabanlı işlemlerin gerçek EDI tabanlı işlemlere dönüştürülmesi ve OCF'in sistem mimarisinde kullanılması yer almaktadır.

EDI tabanlı bir sistem için öncelikle uluslar arası bir koordinasyonun sağlanması ve medikal alanda işlerin yönetilmesi için stabil veri kümeleri, uygun elektronik dokümanlar ve prosedürlerin ortaya konması gerekmektedir.

Kullanılması durumunda OCF; kart işletim sistemi, kart dağıtıcısı ve kart terminal satıcısı bağımlılığı problemlerini çözecektir. OCF'in Java ortamında gerçekleştirilmesinden dolayı akıllı kart iletişiminde bunu kullanan SHIC yazılımları üzerlerinde çalıştırıldıkları doktor

bilgisayarları, self-servis terminalleri ve diğer SHIC bileşenleri üzerindeki işletim sistemlerinden bağımsız çalışabileceklerdir.

Halihazırda SHIC sistemi iki alt sistemden oluşmaktadır. İlki self-servis terminallerinin yer aldığı ve hastaların sağlık sigorta kartları üzerinde bilgi güncellemesi yaptıkları ağ sistemi; ikincisi ise doktor muayenehanelerinde yer alan ve herhangi bir ağa bağlı olmayan terminallerin bulunduğu alt sistemdir. Doktor muayenehanesindeki terminallerin gerçek zamanlı uzak iletişim gerçekleştirememesi, sistemde şu anda görülen bir eksiklik olup ileride bu terminallerle ve Slovenya'daki sağlık sistemi bilgi omurgası ile de güvenli bir ağ bağlantısına gereksinim duyulacağı açıktır. (Novak et al., 1999, 2001)

3 AKSS GENEL TANITIMI VE MİMARİSİ

Tez çalışması kapsamında tasarlanan ve uygulaması gerçekleştirilen sisteme *Akıllı Kart Sağlık Sistemi (AKSS)* adı verilmiştir. Bu sistem ile bir hastane bünyesinde:

- Sağlık kayıtlarının sisteme kolay aktarılması
- Sağlık personelinin, ihtiyacı olan bilgiye hızlı erişimi
- Sürekli tekrarlanan form doldurma işlemlerinin en aza indirilmesi
- Hasta bilgilerine güvenli ve yetkilendirilmiş erişimin sağlanması
    - Kart üzerindeki bilgilere güvenli kart oturumunun açılması dışında ancak özel bir PIN ile erişim ve PIN kontrolünün kartın kendisi tarafından yapılıyor olması
    - Hangi sağlık personelinin hangi bilgiye erişeceğinin belirlenmiş olması
- Tutulacak bilgi miktarına göre:
    - Hasta üzerinde ek testlerin yapılması ihtiyacının ortadan kaldırılması
    - Yanlış ilaç kullanımının önlenmesi

hedeflenmiştir.

Bu bölümde ilk olarak sistemin genel tanıtımı ve işleyişi ele alınmıştır. Daha sonra sistemin mimarisi ve yazılım tasarımı hakkında bilgiler yer almaktadır.

3.1 AKSS Genel Tanıtımı

AKSS' de hem hastaların hem de doktorların birer akıllı kartı vardır. Doktorlar sistemde işlem yapabilmek için kendilerine ait kartları kullanmak zorundadırlar. Hasta kartlarında ise uzak veritabanlarına bağlantı kurulmadan erişilebilecek hasta bilgileri yer alır. Bu bilgilerin yanı sıra tasarlanan hasta kartları üzerinde sistemin işleyişi ile ilgili diğer bilgiler yer alır.

Hastanenin merkezi veritabanında - ki bu veritabanı dağıtık bir yapıda olabilir – sistem kullanıcılarının hem kart üzerindeki hem de kart üzerinde tutulamayan diğer bilgileri yer alır.

Sistem, kliniklerin kendilerine ait lokal veritabanları ile de bağlantılı olacak şekilde tasarlanmıştır. Burada düşünülen; hasta akıllı kartları kullanılarak hastanın çeşitli kliniklerde tutulan bilgilerine *güvenli* olarak erişimi sağlamaktır. Sistemin gerçekleştirilmesinde lokal klinik veritabanı olarak Ege Üniversitesi Tıp Fakültesi Nöroşirürji Ana Bilim Dalı'nda halihazırda kullanılmakta olan sistemde yer alan veritabanı yapısı ve örnek veriler kullanılmıştır.

Aşağıdaki alt başlıklarda hazırlanan sistemin içerdiği güvenlik ve yetkilendirme özellikleri, sistemde kullanılan hasta ve doktor akıllı kartlarının içerikleri, doktor ve hastaların sistemi nasıl kullanacağı ve ilgili sistem oturumları açıklanmıştır. Sistemin tasarımı ve teknik özellikleri ise bundan sonraki bölümlerde ele alınacaktır.

3.1.1 Sistemin içerdiği güvenlik ve yetkilendirme özellikleri:

Akıllı kartların okunmasından verilerin uzak sunuculara iletimine ve veritabanlarına yazılmasına kadar sistemin her bileşeninde güvenlik ön plana çıkmaktadır.

Muayenehanelerde yer alan kart terminalleri ile kart kabul cihazları (CAD) arasında öncelikle bir güvenlik kanalı oluşturulmakta ve veriler doğrudan terminal üzerinde çalışan uygulamaya transfer edilmektedir. Kart okuyucuya bir doktor veya hasta akıllı kartı yerleştirildiğinde, merkez uygulama ile kart arasında öncelikle anahtar alışverişi ile karşılıklı doğrulama sağlanmaktadır. Daha sonra kart sahibinden karta erişim şifresi (PIN) alınmakta şifre *kart tarafından* kontrol edilmekte ve doğru olması durumunda kart oturumu açılmaktadır.

Veritabanı istemci bilgisayarlarının hem hastane merkez hem de kliniğe özel veritabanlarına erişiminde özellikleri daha sonra açıklanacak olan özel ve güvenli bir dağıtık sistem protokolü yer almaktadır. Bu protokolün kullandığı kanal üzerinde de veriler şifreli olarak gönderilmektedir.

Sistemde doktorların yetkilendirmelerini sağlamada ve kart terminalleri üzerinden sistem sunucularına erişimlerinde dijital imzalar kullanılmaktadır.

3.1.2 Hasta kartı üzerinde yer alan bilgiler

Sistemde hastalara ait kartlar üzerinde yer alan bilgiler aşağıdaki alt başlıklarda belirtilmiştir:

3.1.2.1 Hasta genel bilgileri

Hasta ID: Tüm sistem içerisinde her hastanın tek (unique) bir tanımlama numarası olacaktır. Hasta ID olarak T.C. kimlik numaralarının kullanılması öngörülmüştür.

PIN: Hasta kartına erişim için gerekli 4 haneli kart giriş şifresi

Bu bilgilerin dışında hastanın adı, soyadı, doğum tarihi, kan grubu, cinsiyeti, adresi, ev, iş ve cep telefonu numaraları da hasta genel bilgileri başlığı altında kart üzerinde yer alır.

3.1.2.2 Acil durum iletişim bilgileri

Acil durumda iletişime geçilecek olan hasta yakınına ait ad, soyad, yakınlık derecesi, ev, iş ve cep telefonu bilgileri kart üzerinde yer almaktadır.

Bu bilgiler ve hasta genel bilgileri kart şifresi (PIN) tarafından korunmamaktadır. Yani şifre girilmesi gerekmeden bu bilgilere erişilebilir. Özellikle acil durumlarda hastanın bilincinin yerinde olmaması ve dolaysıyla PIN'in elde edilememesi söz konusu olabilir. Böyle bir durumda karttan hasta genel bilgileri ve acil durum bilgilerinin okunmasında izin verilmektedir. Hasta genel bilgileri ve acil durum kontak bilgileri dışındaki tüm bilgiler şifre korumalıdır.

3.1.2.3 Sigorta bilgisi

Hastanın bağlı bulunduğu resmi sigorta kurumu (Bağ-Kur, Emekli Sandığı, SSK) ve bu kurumdaki numarası kart üzerinde yer almaktadır. Hastanın özel sigortası olması durumunda da bu sigortaya ait bilgiler (özel sigorta şirketinin adı ve poliçe numarası) de kart üzerinde yer alır.

3.1.2.4 Hasta sağlık bilgileri

Hasta sağlık bilgileri olarak kartta yer alan bilgiler:

- Hastanın geçirmiş olduğu önemli ve/veya kronik hastalıklarının adları ve bulgu tarihleri
- Sürekli kullandığı ilaçlar ve kullanım miktarları

- Hastanın alerjileri ve bu alerjilerin bulgu tarihleri
- Hastanın yaptırmış olduğu aşılar ve bu aşıları yaptırma tarihleri
- Hastanın geçirmiş olduğu ameliyatlara ait bilgiler (ameliyatın adı, gerçekleştirildiği tarih, klinik ve özet açıklama)
- Yukarıdaki kategorilere girmeyen diğer bilgiler

Kart kapasitesinden dolayı her kategoride belli sayıda bilgi yer almaktadır. Örneğin hasta kartı üzerinde en fazla 8 adet alerji bilgisi yer almaktadır. Eğer hastanın 8'den fazla alerji bulgusu varsa bulgu tarihlerine göre en güncel 8 tanesi kartta yer almaktadır. Tüm alerji bilgileri ise hastanın, hastane merkezi veritabanındaki kayıtlarında yer alır. Diğer kategorilerde de benzer durum geçerlidir.

Hasta sağlık bilgileri doktorlar için sadece okunabilir durumdadırlar. Bu bilgilerde bir değişikliğin yapılması ve dolaysıyla kart üzerinde güncelleme ancak hastane yetkili birimi tarafından gerçekleştirilmektedir.

3.1.2.5 Hastanın son muayene ve son reçetesine ait bilgiler

Kart üzerinde hasta son muayene ve reçete bilgileri de yer almaktadır. Son muayene bilgileri olarak:

- Muayenenin gerçekleştirildiği tarih
- Muayenenin gerçekleştirildiği klinik
- Muayeneyi gerçekleştiren doktora ait bilgiler (Doktorun sistemdeki ID'si, adı ve soyadı)
- Muayenenin özeti yer alır

Kartta yer alan son reçete bilgileri ise:

- Reçetenin yazıldığı tarih
- Reçetenin yazıldığı klinik
- Reçeteyi yazan doktora ait bilgiler (Doktorun sistemdeki ID'si, adı ve soyadı)
- Reçetenin içeriği (İlaç listesi)
- Reçetenin hastane tarafından onaylı olduğuna ya da olmadığına dair bilgi

Sistemde düşünülen; doktorların muayene işlemini bitirdikten sonra hastaya ait kartta muayene bilgisi ve varsa yeni reçeteye ait bilgilerini yazmasıdır. Böylelikle kağıt doküman kullanımı yerine bilgilerin kartta elektronik olarak taşınması düşünülmektedir. Hasta kendisine ait kartla hastane sistem yönetimine başvurduğunda doktorun yazmış olduğu muayene ve reçete bilgileri merkezi veritabanına aktarılmakta ve yazılan reçeteye onay verilmektedir.

Hasta son reçetesinin kartta yer alması; kartın eczanelerde de kullanımına imkan verir. Kart okuyucuya sahip bir eczanede, hasta reçetesi elektronik ortamda işlem görebilir.

3.1.2.6 Sistem bilgileri

Hazırlanan sağlık sisteminin işleyişinde kullanılan ve hasta akıllı kartında yer alan diğer bilgileri ise şunlardır:

- Kartın hastaya veriliş ve son güncelleme tarihleri
- Hastanın kayıtlarının bulunduğu hastane merkez veritabanı sunucusuna ait bilgiler
- Hasta DES (Digital Encryption Standard) anahtarı

Kart üzerinde, hastanın kayıtlarının tutulduğu hasta veritabanı sunucusunun adı ve ilgili ağ protokolündeki adresinin bulunması, hasta veritabanının dağıtık yapıda olabilmesine imkan vermektedir. Şöyle ki; hastaya ait kartı kullanan terminal, gereken durumlarda hasta bilgilerini getireceği veritabanına ait bağlantı bilgilerini karttan alabilmekte, bu da sisteme dinamik bir yapı kazandırmaktadır.

Hasta DES anahtarı, hasta verilerinin ağ üzerinde şifreli olarak iletimini sağlamaktadır. Hasta oturumunda bu anahtarın nasıl kullanıldığı ilerleyen bölümlerde yer almaktadır.

3.1.3 Doktor kartı üzerinde yer alan bilgiler

Sistemde doktorlara ait kartlardaki bilgi miktarı hasta kartlarındaki kadar fazla olmayıp saklanan veriler doktor genel bilgilerinden ve sistem bilgilerinden ibarettir.

3.1.3.1 Doktor genel bilgileri

Doktor ID: Tüm sistem içerisinde her doktorun tek bir tanımlama numarası olacaktır. ID olarak doktorların tıp diploma numaralarının kullanılması düşünülmüştür.

PIN: Doktor kartına erişim için gerekli 4 haneli kart giriş şifresi

Bu bilgilerin dışında doktorun adı, soyadı, çalıştığı klinik veya bölümün adı, adresi, ev, iş ve cep telefonu numaraları da kart üzerinde yer alır.

3.1.3.2 Sistem bilgileri

Sisteminin işleyişinde kullanılan ve doktor akıllı kartında yer alan bilgiler:

- Kartın doktora veriliş ve son güncelleme tarihleri
- Doktor kayıtlarının bulunduğu hastane merkez veritabanı sunucusuna ait bilgiler
- Doktor dijital imzası

Kart üzerinde, doktor kayıtlarının tutulduğu hastane veritabanı sunucusu ile ilgili bağlantı bilgilerinin bulunması tıpkı hasta kartlarında olduğu gibi hastanedeki sisteme kayıtlı olan doktorlara ait bilgilerin tek bir sunucuda değil de birden fazla sunucuda saklanmasına izin verir. Ayrıca sistem içerisinde yer alan mesajlaşma altyapısında da bu bilgilerin kartta tutulması rol oynamaktadır.

Hasta bilgilerinin klinik veritabanlarına kaydında yetkilendirmeyi sağlayacak olan doktora ait dijital imzadır. DSA (Digital Signature Algorithm) adı verilen yapıya ait özel anahtar dijital imza olarak kartta saklanmaktadır. Lokal veritabanlarına doktor tarafından gönderilecek olan bilgiler doktor kartındaki bu anahtar ile imzalanarak gönderilmektedir. Karşı tarafta da bu bilgiler karttaki özel anahtarın karşılığı olan genel anahtar ile doğrulanarak sisteme kaydedilir.

3.1.4 Doktor oturumu

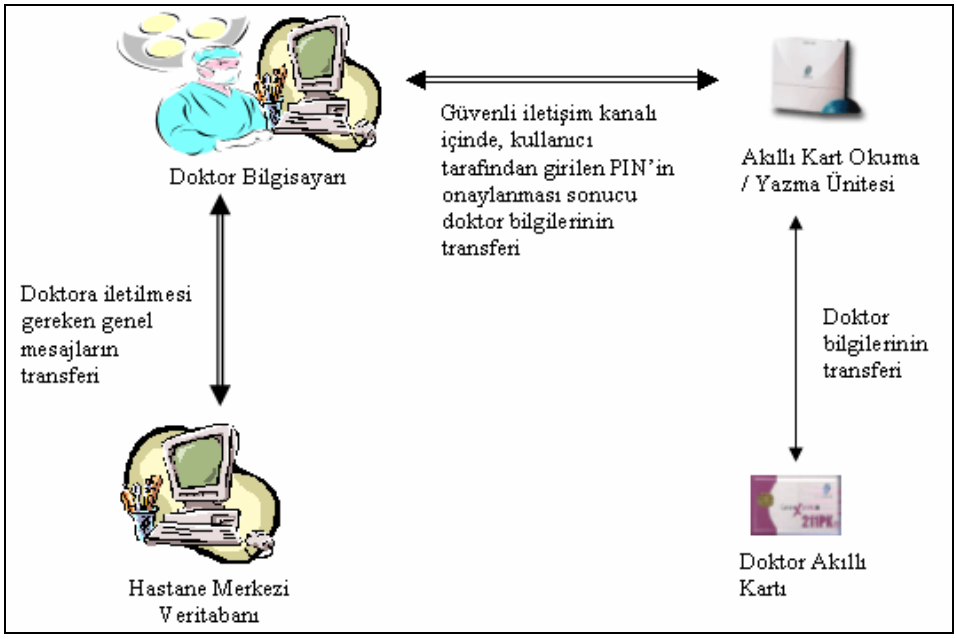
Akıllı Kart Sağlık Sistemi'nin muayenehanelerde kullanılan kart ve veritabanı istemcisi terminalleri üzerinde kullanılabilmesi için öncelikle sistem üzerinde bir doktor oturumunun açılması gerekmektedir. Öngörülen, her doktorun muayenehanesinde bir terminalin bulunmasıdır.

Terminal üzerinde çalışacak olan uygulama ilk olarak bir doktorun oturum açmasını beklemektedir. Bunun için doktorun kart kabul cihazına kartını yerleştirmesi gerekmektedir. Kart yerleştirilince ev sahibi uygulama ile akıllı kart birbirlerini tanıyarak güvenli kart kanalını oluştururlar. Bu işlemler sonucunda kart sahibinden karta giriş şifresi (PIN) istenir. Şifrenin kart tarafından onaylanması sonucu doktorun kayıtlarının bulunduğu uzak hastane veritabanı ile iletişime geçilerek doktora ait mesajlar ekrana getirilir. Ayrıca oturum için gerekli olan doktor bilgilerinin (ID, ad, soyad, dijital imza, vb.) karttan uygulamaya transfer edilerek doktor oturumunun açılması tamamlanmış olur. Bu aşamadan sonra terminal, hasta kartlarını kabul etmeye hazırdır. Doktor, kartını kart okuyucudan çıkararak hasta kabullerine başlayabilir.

Buradaki bir diğer yaklaşım doktor oturumunun, doktor kartının kart okuyucusunda yer aldığı sürece açık olması olabilirdi ki bu durum zaten hasta kartları ve oturumları için geçerlidir. Ancak sistemin hazırlanmasında kullanılan kart okuyucu ünitesi tek yuvaya sahip olduğundan aynı anda iki kartın işlem görmesi mümkün olmamıştır. Kartların karşılıklı birbirini doğrulaması yerine doktor oturumu için gerekli bilgilerin geçici olarak karttan uygulamaya transferi gerçekleştirilmiştir ve istemci uygulama bir doktor oturumu açık değilken kesinlikle bir hasta kartını güvenlik nedeniyle kabul etmemektedir.

Hasta kabullerinden sonra doktor, oturumunu ilgili ara yüzden kapatabilir. Böyle bir durumda yeni oturumun açılması için yeniden doktor kartına ihtiyaç duyulmaktadır. Benzer şekilde ev sahibi uygulama tamamen kapatılırsa doktor oturumu ve eğer açıksa hasta oturumu da otomatik olarak kapatılır.

Şekil 3.1’de doktor oturumunda görev alan sistem bileşenleri gösterilmiştir:



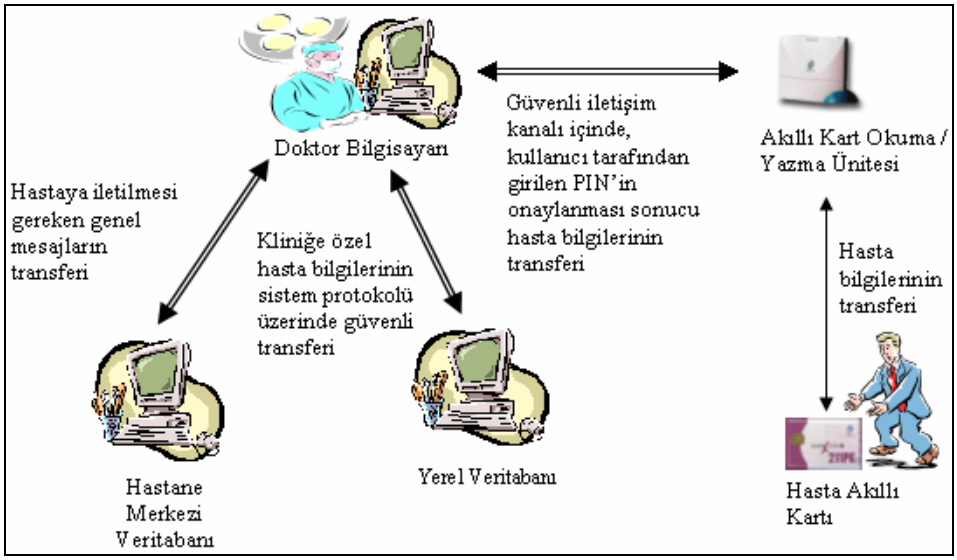
Şekil 3.1 Doktor oturumu

3.1.5 Hasta oturumu

Sistemin istemci bileşenlerinin yer aldığı muayenehane terminallerinde bir hasta oturumunun açılabilmesi için ön şart bir doktor oturumunun terminal üzerinde açık olmasıdır.

Uygulama işlem için bir hasta kartı bekler durumda iken hasta kartı yerleştirildiğinde doktor oturumunda olduğu gibi hasta akıllı kartı ile ev sahibi uygulama anahtar alışverişinde bulunarak birbirlerini tanırlar ve güvenli kart kanalını oluştururlar. Bu aşamadan sonra kart sahibinden şifre (PIN) girişi istenir. Girilen şifre hasta kartı tarafından onaylandıktan sonra öncelikle hasta bilgilerinin yer aldığı uzak veritabanı sunucusundan hastaya iletilmesi gereken mesajlar alınarak kullanıcıya iletilir. Daha sonra hasta oturumunun açılması için gerekli olan hasta genel bilgileri, hasta şifreleme anahtarı gibi bilgiler karttan alınır.

Doktor, kart üzerindeki hasta genel bilgilerine erişebileceği gibi lokal klinik veritabanında yer alan bu hastaya ait kliniğe özel bilgilere de ulaşabilir ve yetkili ise güncelleyebilir. (Şekil 3.2)



Şekil 3.2 Hasta oturumu

Doktor muayene işlemlerini tamamladıktan sonra muayene bilgilerini ve eğer varsa reçete bilgilerini hasta kartına kaydeder. Hasta oturumunun kapatılması ise ara yüzden ilgili isteğin belirtilmesi ile ya da uygulamanın tamamen kapatılması ile mümkündür. Ara yüzden hasta kartı oturumu kapatıldığında hastaya kartı iade edilir ve uygulama bir sonraki hasta kabulü için hazır konumuna geçer. Burada dikkat edilmesi gereken hasta oturumu süresince kart, kart okuyucusuna yerleştirilmiş ve aktif durumdadır. Kart, hasta oturumu açıkken herhangi bir zamanda okuyucudan çıkarılırsa hasta oturumu ve buna bağlı ara yüz diyalogları güvenlik nedeniyle otomatik olarak kapanır.

3.1.5.1 Mevcut veritabanları ile iletişim

Sistem hazırlanırken göz önüne alınan bir diğer konu da hasta akıllı kartları kullanılarak hastane bünyesinde yer alan kliniklerin/bölümlerin veritabanlarına erişimdir. Her klinikte kart erişimi ile ilgili istemci modülleri ortak olup sadece hastanın kliniklere özel bilgilerinin getirildiği ara yüzün her klinik için ayrı olması düşünülmüştür. Sistemin gerçekleştirilmesinde örnek bölüm olarak daha önce de belirtildiği gibi Ege Üniversitesi Tıp Fakültesi Nöroşirürji A.B.D. alınmıştır. Hastaların bu bölüme ait kayıtları ayrı bir ara yüzde, bu bölümde görev alan doktorların ekranına getirilmektedir.

Temelde doktor kendi kliniğinde ilgili hastaya ait kayıtları güncellemek istediğinde, bu iş için yetkisinin olduğunu belirtmek zorundadır. Doktor, özel protokol üzerinden ilgili hastanın bilgilerini istediğinde, sunucu tarafında hasta bilgileri şifrelenerek doktorun bilgisayarına iletilir. Alıcı tarafta gelen bilgiler ancak hasta kartında yer alan şifreleme anahtarı ile çözülebilir. Eğer doktor hastasına ait bu bilgileri veritabanında güncellemek isterse bu bilgiler önce hastanın kartından alınan şifreleme anahtarı ile şifrelenir. Daha sonra bu şifreli bilgiler doktorun kartından alınan ve doktor oturumunda mevcut olan doktorun dijital imzası ile imzalanırlar. Şifreli ve imzalanmış veriler sisteme ait özel protokol üzerinden alıcı konumundaki sunucuya iletilir. Sunucu tarafında öncelikle doktorun dijital imzası doğrulanır. Doğrulama başarılı olursa bu sefer gelen şifreli hasta bilgileri deşifre edilir. Tüm bu işlemler başarılı olursa güncellenmiş bilgiler veritabanına kaydedilir. İlgili işlemi gerçekleştiren doktor ve işlem zamanı gibi bilgiler sunucu kayıtlarında (log dosyalarında) yer alır.

3.1.6 Sistem yönetim birimi

Sistem yönetim birimi, temelde hastane merkezi veritabanının yönetilmesinden sorumludur. Bunun dışında; hasta ve doktorların sisteme kaydı, yeni kullanıcılar için akıllı kart hazırlanması, kartların iptali ve karttan sistem veritabanına veya sistem veritabanından karta bilgi

güncellemesi gibi kart işlemlerini yürüten hastane birimi olarak da görev yapması tasarlanmıştır.

Merkez veritabanına yeni hasta veya doktor kaydedilirken bu birim tarafından ilgili kullanıcıya ait akıllı kart da hazırlanarak kart sahibine teslim edilir. Kart üzerinde bilgi güncelleme de yine bu birim tarafından yapılmaktadır. Örneğin hasta kartında yer alan sağlık bilgilerinde bir değişiklik gerektiğinde (yeni bir alerji bulgusunun bulunması veya varolan bir alerji kaydının bulgu tarihinin değiştirilmesi gibi) değişikliği talep eden kliniğin onayı ile hastanın hem sistem veritabanındaki hem de akıllı kartı üzerindeki sağlık bilgilerin güncellenmesi bu birim tarafından gerçekleştirilir.

Hasta, muayene işlemleri sonucunda yine yönetim birimine akıllı kartı ile başvurur. Burada, muayene sonrası doktor tarafından karta yazılan muayene ve varsa reçete bilgilerinin sistem veritabanına kaydı gerçekleştirilir ve yazılan reçete için hastane tarafından verilen onay bilgisi karta yazılır.

Bu birim aynı zamanda sistem kullanıcılarına iletilmek istenen mesajların / duyuruların da merkezi veritabanında, alıcıya ait alanlara kaydını gerçekleştirir. Örneğin bir hastaya gönderilmek istenen mesaj bu hastanın sistem veritabanındaki mesaj alanına kaydedilir. Hastanın akıllı kartı herhangi bir kart terminalinden merkezi sistem ile iletişime geçtiğinde bu mesaj terminale dolaysı ile alıcısına iletilmiş olur.

3.2 Sistem Mimarisi

Bu bölümde hazırlanan sistemin yazılım mimarisi ve tasarımı ile ilgili bilgiler yer almaktadır. İlk alt bölümde sistemin genel yapısı ortaya konmuş ve 3 katmanlı mimaride görev alan sistem bileşenleri tanıtılmıştır. Sonraki alt bölümlerde ise bu bileşenler açıklanmaktadır.

3.2.1 Sistemin genel yapısı

Sistem, 3 katmanlı yazılım mimarisi modeli baz alınarak tasarlanmıştır. Modelde istemci, sunucu ve veritabanı katmanları yer almaktadır.

Tipik bir istemci/sunucu uygulamasında istemciler sunucu üzerinde yer alan veritabanına bağlanarak bu veritabanı üzerinde sorgular gerçekleştirmektedirler. Ancak böyle bir yapıda istemci makine üzerinde yer alan yazılım bileşenleri tamamı ile sunucuya bağımlıdır. Veritabanına erişim ve sorgu işlemleri ile ilgili kodlar istemci tarafında bulunmak zorundadır. Oysa istemciler için bu işi yerine getiren ve iş akışını kontrol eden bir yapının olması bileşenlerin mümkün olduğunca birbirinden bağımsız olmasına ve yeniden kullanılabilirliğine izin verir. Böyle bir yapı üç katmanlı sistemin uygulanması ile mümkündür.

Nesneye dayalı bir yazılım olarak ortaya konan sisteme ait bileşenlerin bütünleşik olarak çalışmasında *MVC (Model, View & Controller)* yazılım mimarisi tasarım deseninden faydalanılmıştır. (Gamma et al., 1994) ve (Buschmann et al., 1996)'da da belirtildiği gibi MVC deseninde görünüm bileşenleri – ki bunlar arasında sistem aktörlerinin kullandığı formlar ve diğer ara yüzler yer alır – sistem veri modeli ile bir denetleyici mekanizma aracılığı ile haberleşir. Buradaki en önemli avantaj sistem içerisinde yer alan istemci uygulamalarda sadece ilgili ara yüzler yer alır ve bu uygulamalar kullandıkları veri modelinin saklandığı ilişkisel veri tabanından tamamı ile bağımsızdır.

Sistemde MVC deseninin kullanılması ile muayenehanelerde yer alması düşünülen ve veritabanları için istemci konumunda yer alan uygulamalar sistemin kullandığı veritabanı yapılarından bağımsız olup sadece sistem özel protokolü üzerinden denetleyici konumundaki uzak nesneler ile haberleşmektedirler. Burada söz konusu olan özel protokol, TCP/IP üzerinde Java RMI teknolojisinin bir uygulamasıdır.
İstemciler RMI kanalı üzerinden uzak nesnelerin ara yüzlerini kullanmakta ve veritabanı işlemlerini gerçekleştirmektedirler. Örneğin

klinik veritabanına bir bilgi güncellemesi gerektiğinde istemci uygulama bilgileri kullanıcı ara yüzünden alarak bu işlem ile görevli uzak nesnenin metodunu çağırıp bu metoda parametre olarak aktarır. İstemcinin bu iş için yetkili olup olmadığı ve veritabanında bilgi güncelleme bu uzak nesnenin görevidir.

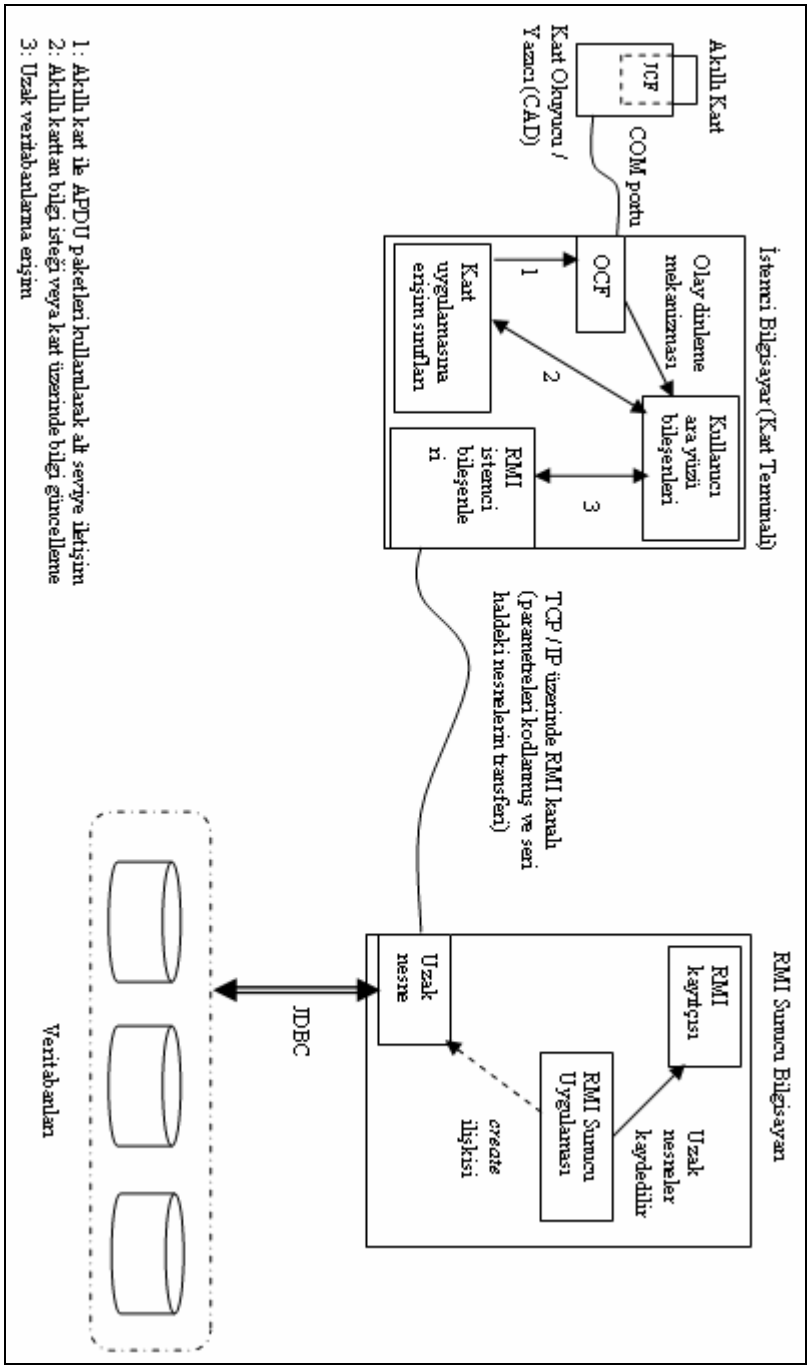
MVC deseni ve RMI protokolünün kullanılması aynı zamanda sistem yetkilendirmesinde ve veri güvenliğinde gerekli olan dijital imzaların ve şifreleme anahtarlarının kullanılmasına da imkan tanımaktadır. Hedeflenen; özellikle klinik veritabanlarına yetkilendirilmiş erişim ve hasta bilgilerinin güvenliğidir. Bunun için protokol üzerinde veriler şifrelenmiş ve imzalı olarak transfer edilmektedir. Denetleyici konumdaki uzak nesnenin metoduna istemci ara yüzleri kullanılarak geçirilen parametreler, nesne serileştirme ve kodlama yöntemlerine ek olarak bir de şifreli ve imzalı olarak iletilmektedir. Eğer böyle bir protokol ve MVC gibi bir desen kullanılmayıp istemciler direkt veritabanına erişim de bulunsalardı hem istemci yazılım bu veritabanlarına bağlı kalacak hem de sistem bu güvenlik ve yetkilendirme özelliklerini sunamayacaktı.

Sistem yazılımı yaklaşık 30000 satır koddan oluşmuştur. Tasarım desenine uygun olarak yazılım 7 paket içerisinde uygun sistem bileşenlerine dağıtılmıştır.

Şekil 3.3'te tüm sistem mimarisi ve yer alan bileşenler gösterilmiştir. Kart terminali, RMI sunucuları ve veritabanı sunucuları ile ilgili bilgiler izleyen alt başlıklarda verilmektedir.

3.2.2 Kart terminalleri

Akıllı Kart Sağlık Sistemi kapsamında, muayenehanelerde doktorlara ait bilgisayarların yer alması ve bu bilgisayarların birer akıllı kart terminali olarak görev yapması tasarlanmıştır.



Şekil 3.3 Hazırlanan sağlık sisteminin mimarisi

Her kart terminaline bir akıllı kart okuyucu / yazıcı ünitesi bağlıdır. Uygulamada bağlantı noktası olarak, kullanılan CAD'in özelliği gereği COM bağlantı noktası kullanılmıştır. Sağlık sistemi yazılımları dışında bilgisayar üzerinde bir Java sanal makinesinin ve OCF kütüphanesinin yer alması yeterlidir.

Bilgisayar üzerinde akıllı kart ortamının başlatılması ve bağlı bulunan kart terminalinin iletişim için hazır hale getirilmesinde OCF görev almaktadır. Hazırlanan kart iletişim paketi de bu çatıyı kullanmaktadır.

Hem doktor hem de hasta kartları ile iletişimi sağlayan kart istemci paketinde; kart ortamını hazırlayan, akıllı kart uygulamasını (*"applet"*) seçen ve bu program parçacığı ile APDU paketleri aracılığıyla iletişimi sağlayan sınıflar yer almaktadır. Bu paket bir uygulama programlama ara yüzü (API) şeklinde tasarlanmış olup kart istemcilerini özellikle APDU iletişiminden soyutlamayı esas almıştır. Doktor veya hasta kartına özel okuma yazma işlemlerini bu paket bünyesindeki sınıflar yerine getirmektedir. Böylelikle kullanıcı ara yüzünden karta erişim işlemleri yerine getirilirken arka planda bu paketin sunduğu hazır nesneler kullanılmakta; bu da özellikle APDU bayt vektörlerinin hazırlanması gibi zahmetli işlemlerden ara yüz geliştiricilerinin soyutlanmasını sağlamaktadır.

Akıllı kart terminalleri aynı zamanda sistem RMI protokolünde uzak metot çağıran istemciler olarak görev yapmaktadırlar. Daha önce de belirtildiği gibi istemci bilgisayar üzerinde uzak veritabanı bağlantılarını gerçekleştirmek yerine bu bağlantıları gerçekleştiren ve istenen sorguları yerine getiren uzak nesneler kullanılmaktadır. Böylelikle hem görünüm (view) bileşenlerinin sistem veri modelinden ayrıştırılması sağlanmış hem de veri şifreleme ve kullanıcı yetkilendirme özelliklerini içeren dağıtık bir sistem protokolünün gerçekleştirilmesi imkanı doğmuştur.

Sistem istemci bilgisayarlarında, kullanıcılar ile etkileşimde görsel kullanıcı ara yüzü bileşenleri – ki bunlar JFC'den (Java Foundation

Classes) türetilen Java nesneleridir – görev almaktadır. MVC mimarisi içerisinde görünüm katmanını bu sınıfların topluluğu oluşturmaktadır.

Ara yüz paketi içerisindeki form nesneleri; akıllı karta dayalı sistem olaylarını OCF içerisinde yer alan `opencard.core.event.CTListener` ara yüzünü uygulayarak dinlemektedirler. Böylelikle uygulama, akıllı kartın okuyucuya yerleştirilmesi veya çıkarılması gibi olaylardan haberdar olarak bunlara dayalı kullanıcı oturumlarının açılması veya kapatılması işlemlerini yerine getirmektedir.

Olaya dayalı (event driven) model yerine kart istemci yazılımında uygulamaya dayalı (application driven) bir model de kullanılabilirdi. Uygulamaya dayalı model akıllı kart nesnelerine senkron erişimi içermekte olup daha çok tek iş parçacığı üzerinde çalışan (single threaded) uygulamalar için elverişlidir. Örneğin konsoldan çalışan bir uygulama için bu model uygun olabilir ki bu model tez çalışmasında ara ürün olarak hazırlanan yazılımlarda denenmiş ve etkileri gözlenmiştir. Ancak görsel kullanıcı ara yüzü bileşenlerini içeren yazılımlarda, (Opencard Consortium, 1999)'da da belirtildiği gibi olaya dayalı bir modelin kullanılması, diğer modele göre kullanımı daha karmaşık olmasına rağmen daha doğrudur. Hazırlanan sağlık sistemi ara yüzleri de birbirleri ile haberleşen çok sayıda grafiksel ara yüz nesnesinden oluştuğu için tasarımda olaya dayalı model temel alınmıştır.

Ara yüz bileşenleri akıllı karttan okuma ve karta yazma işlemlerini ise yukarıda bahsedilen kart iletişim paketi bünyesinde yer alan sınıflar aracılığı ile gerçekleştirirler. Bunun yanı sıra uzak nesne metotlarının çağırılması ve kriptografik işlemler de yine bu bileşenler tarafından yerine getirilir.

3.2.3 RMI sunucuları

Akıllı Kart Sağlık Sistemi'nde, sistem veritabanlarına erişim ve bu veritabanları üzerinde sorguların belirlenen güvenlik ve yetkilendirme politikaları altında gerçekleştirilmesi, kart terminali olan bilgisayarların, metotlarını çağırdıkları uzak nesneler aracılığı ile olmaktadır. Bu nesnelerin üzerinde bulunduğu bilgisayarlar ise sistem RMI sunucularıdır.

Sunucularda herhangi bir akıllı kart bileşeni yer almamaktadır. TCP/IP protokolü yüklü ve üzerinde Java sanal makinesi çalışan bu bilgisayarlar üzerinde sağlık sistemine ait sunucu yazılım paketi yer alır. Her bir hastane bölümünde ya da kliniğinde bir RMI sunucusunun yer alması tasarlanmıştır. Bu sunucuda, o klinikte / bölümde yer alan kart terminallerinin iletişime geçeceği RMI uzak nesneleri yer alır.

Şekil 3.3'te de görüldüğü gibi sunucu yazılımı çalışmaya başladıktan sonra uzak nesneleri oluşturmakta ve bu nesneleri RMI kayıtçısına kaydetmektedir. İstemci terminaller öncelikle bu sunucu üzerindeki kayıtçıya bakarak kullanacakları nesnelerin ara yüzlerini elde ederler. Bu aşamadan sonra terminaller uzak nesnelerin ara yüzlerini kullanarak işlemlerini gerçekleştirirler.

Uzak nesneler şifreli veri alışverişinden, yetki kontrolünden ve JDBC (Java Database Connectivity) teknolojisini kullanarak sistem veritabanlarına bağlantıdan sorumludurlar. Örneğin istemci bilgisayar veritabanına bilgi kaydetmek istediğinde, bu bilgileri şifreleyip imzaladıktan sonra veritabanına kayıtla görevli olan uzak nesnenin metoduna parametre olarak gönderir. Bilgiler TCP/IP üzerindeki RMI kanalından kodlanmış olarak RMI sunucusu üzerindeki *"stub"*a aktarılır. *"Stub"* da kodlanmış bilgileri açarak görevli nesneye iletir. Nesne istemcinin kayıt için yetkili olup olmadığını kontrol ettikten ve şifreli verileri açtıktan sonra gereken veritabanı sorgularını hazırlar. JDBC aracılığı ile de ilgili veritabanıyla bağlantı kurup sorguları gerçekleştirir. Sorgu sonucu yine aynı kanal üzerinden istemciye iletilir.

Görüldüğü gibi muayenehanelerde yer alan terminaller tüm bu kontrol ve veritabanı bağlantı işlemlerinden soyutlanmıştır. Kontrol mekanizması uzak nesneler, dolaysıyla RMI sunucusu üzerinde işlemektedir. Sunucu üzerinde görev alan nesneler sistem MVC mimarisinin denetleyici (controller) katmanını oluşturmaktadırlar.

Uygulamada Ege Üniversitesi Beyin Cerrahi A.B.D. için bir RMI sunucu yazılım paketi hazırlanmış olup bu paket ile ilgili detaylı bilgi, sunucu bileşenlerinin anlatıldığı tez bölümünde yer almaktadır.

3.2.4 Sistem veritabanları

Sistem içerisinde hem hastanenin merkezi bir veritabanın olması – ki bu veritabanı dağıtık bir yapıda olabilir – hem de hastane bölümlerinin (kliniklerinin) kendilerine özel veritabanlarının olması düşünülmektedir. Sisteme kayıtlı hasta ve doktorlara ait genel bilgilerin hastane merkez veritabanında, kliniklere özel hasta bilgilerinin ise ilgili klinik bünyesinde yer alan veritabanlarında yer alması tasarlanmıştır.

Uygulamada, ilişkisel bir hastane merkez veritabanı ve beyin cerrahi bölümüne ait veritabanı hazırlanmıştır. Beyin cerrahi bölümüne ait veritabanı halihazırda bu bölümde kullanılmakta olan veritabanı ile aynı yapıda olup yeniden tasarlanmamıştır.

Hazırlanan veritabanları üzerinde sistemin veri modelini oluşturan nesnelerin özellikleri saklanmaktadır. Model tamamı ile verinin kullanılmasından ve görüntülenmesinden bağımsızdır. Bu nedenle; sistemin MVC mimarisi içerisinde model katmanını bu veritabanları oluşturmaktadır. Söz konusu ilişkisel veritabanları ve bu veritabanlarında yer alan kayıtların sistemde hangi yapıdaki nesneler ile temsil edildikleri (nesnelerin veritabanlarına nasıl *map* edildiği) veritabanı tasarımı bölümünde detaylandırılmıştır.

4 SİSTEM AKILLI KART BİLEŞENLERİ

Bu bölümde ilk olarak sistemde kullanılan akıllı kartlar, kartların teknik özellikleri ve bu kartların içerisinde yer alan yazılımlar hakkında bilgi verilmektedir. İkinci bölümde ise istemci kart terminallerinde bu akıllı kartlar ile iletişime geçecek olan ara yazılım paketi hakkında bilgi verilmektedir.

4.1 Sistemde Yer Alan Akıllı Kartlar

Akıllı Kart Sağlık Sistemi'nde daha önce de belirtildiği gibi hastalara ve doktorlara ait olmak üzere iki tip akıllı kart yer almaktadır. Bu bölümde, söz konusu iki kart ile ilgili teknik bilgiler ve içerdikleri kart yazılımlarının ayrıntıları ortaya konmuştur.

4.1.1 Sistem akıllı kartlarının teknik özellikleri

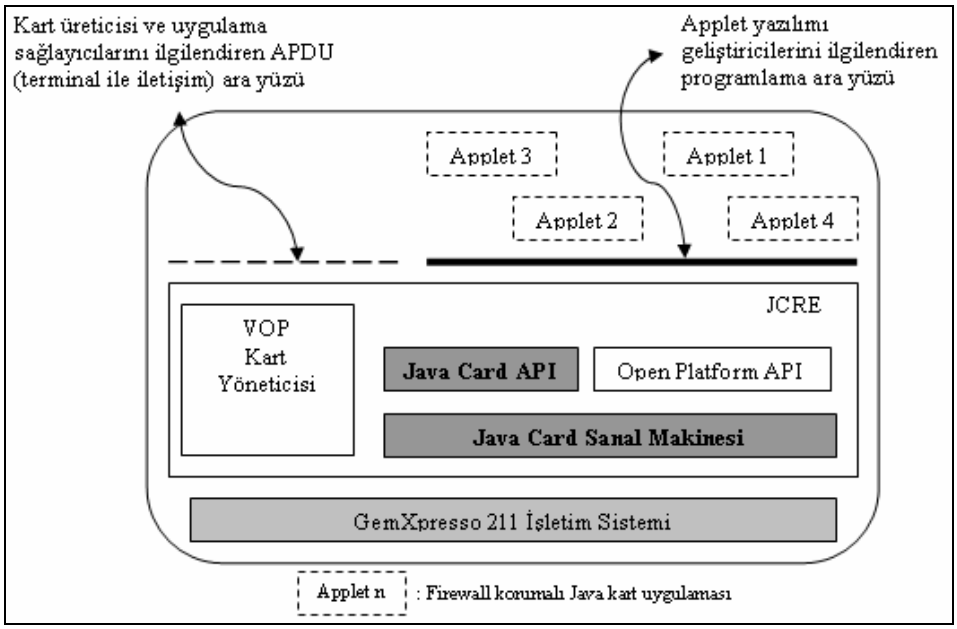
Sistemin gerçekleştirilmesinde kullanılan akıllı kartlar, Gemplus firmasının *GemXpresso 211/PK* model, çoklu "applet" destekli (multiapplet) Java kartlarıdır.

Şekil 4.1'de mimarisi gösterilen GemXpresso 211/PK'nın bileşenleri izleyen alt başlıklarda açıklanmaktadır.

4.1.1.1 Kart işletim sistemi

GemXpresso 211 PK işletim sistemi tek iş parçacıklı bir işletim sistemi olup kart kaynaklarını kontrol etmektedir. I/O alt sistemi ISO 7816-3 ve ISO 7816-4 standartları ile uyumludur.

İşletim sistemi 8-bitlik mikroişlemci üzerinde çalışmaktadır. Kart kaynakları 32 K ROM, 32 K EEPROM ve 2 K RAM'den ibarettir. Ancak bu kaynakların önemli bir kısmı kart işletim sistemi ve diğer sistem yazılımları tarafından kullanıldığından uygulama geliştiricilere yazılımları için 16.5 K EEPROM ve 0.5 K RAM kalmaktadır.



Şekil 4.1 GemXpresso 211 PK Kart Mimarisi

4.1.1.2 JCRE

GemXpresso 211/PK Java kartı Sun Microsystems'in Java Card 2.1 spesifikasyonları ile tam uyumluluk göstermektedir. Kart içerisinde görev alan JCRE; Java Card Sanal Makinesini, Java Card API'yi ve *"applet"* kaynak paylaşımı ve kalkan servislerini içermektedir. Java Card teknolojisi ile ilgili ayrıntılı bilgi tez alt yapı çalışmaları bölümünde verilmiştir.

4.1.1.3 Open Platform

Visa'nın Open Platform'u (OP), bünyesinde birden fazla uygulama içeren akıllı kartların yönetimi için ortaya konmuş genel bir çatıdır. JCRE'nin bir uzantısı olan bu platforma ait bileşenler:

- Kart ve kart üzerindeki uygulamaların yaşam döngüsünün kontrolünde kullanılan komut kümesi
- Kart terminali ile kart arasında güvenli iletişim kanalının kurulması ve anahtar alışverişini içeren kart güvenlik komutları kümesi
- OP özelliklerini uygulama geliştiricilerin kullanmasını sağlayan API'dir.

Tez kapsamında, akıllı kartlar üzerinde çalışmak üzere hazırlanan yazılımların sadece GemXpresso 211/PK tip akıllı kartlarda değil de tüm Java kartlarda çalışması düşünülmüştür. Bu nedenle Java Card 2.1 çatısı baz alınmıştır. Uzantısı olan VOP (Visa Open Platform)'dan ise sağlık sistemi akıllı kartlarının terminaller ile iletişiminde güvenlik kanalının kurulmasında yararlanılmıştır.

4.1.2 Hasta akıllı kartı

4.1.2.1 Kart üzerinde saklanan bilgiler

Hasta kartı üzerinde hastaya ait genel bilgiler, acil durum kontak bilgileri, hasta sigorta bilgileri, hasta sağlık bilgileri, hastanın son muayenesi ve son reçetesi ile ilgili bilgiler ve sağlık sisteminin çalışmasında kullanılan sistem bilgileri yer alır. Tutulan toplam bilgi miktarı *4058* bayttır.

Hasta genel bilgileri:

Kart üzerinde toplam 209 baytta saklanan hasta genel bilgileri Tablo 4.1'de gösterilmiştir.

Bilgi	**Uzunluk (Bayt)**	**Açıklama**
Hasta ID	11	Hastayı tanımlayan sistem geneli numara
PIN	4	Kart kullanım şifresi
Ad	20	
Soyad	20	
Doğum Tarihi	10	GG/AA/YYYY formatında
Kan Grubu	10	
Cinsiyet	1	Erkek için "1", kadın için "2"
Adres	100	
Ev Telefonu	11	###-####### formatında (alan kodu – numara)
İş Telefonu	11	###-####### formatında (alan kodu – numara)
Cep Telefonu	11	###-####### formatında (alan kodu – numara)

Tablo 4.1 Kart üzerinde yer alan hasta genel bilgileri

Hasta acil durum kontakt bilgileri:

Kart üzerinde toplam 93 baytta saklanan hasta acil durum kontak bilgileri Tablo 4.2'de gösterilmiştir.

Bilgi (Hasta Yakınının)	**Uzunluk (Bayt)**	**Açıklama**
Adı	20	
Soyadı	20	
Yakınlık Derecesi	20	"Eşi", "Annesi", "Kardeşi", vb. karakter dizisi
Ev Telefonu	11	###-####### formatında (alan kodu – numara)
İş Telefonu	11	###-####### formatında (alan kodu – numara)
Cep Telefonu	11	###-####### formatında (alan kodu – numara)

Tablo 4.2 Kart üzerinde yer alan hasta acil durum kontak bilgileri

Hasta sigorta bilgileri:

Kart üzerinde toplam 100 baytta saklanan hasta sigorta bilgileri Tablo 4.3'te gösterilmiştir.

Bilgi	**Uzunluk (Bayt)**	**Açıklama**
Resmi Kurum Adı	30	"Emekli Sandığı", "Bağ-Kur" veya "SSK"
Resmi Kurumdaki Sigorta No	20	İlgili kurumdan hastaya verilen numara
Özel Sigorta Kuruluşu Adı	30	
Özel Sigorta Poliçe No	20	

Tablo 4.3 Kart üzerinde yer alan hasta sigorta bilgileri

Hasta sağlık bilgileri:

Kart üzerinde hasta sağlık bilgisi olarak hastanın alerjileri, yaptırdığı aşılar, geçirmiş olduğu ve/veya kronik hastalıkları, sürekli kullandığı ilaçlar ve geçirmiş olduğu ameliyatlar ile ilgili bilgiler yer alır. Tüm bu bilgiler için kart üzerinde toplam 1830 baytlık bir bellek alanı gerekmektedir:

- Hastaya ait en fazla 8 adet alerji kaydı kart üzerinde saklanmaktadır. Her bir alerji kaydı 20 baytlık bulgu adından ve 10 baytlık GG/AA/YYYY formatındaki bulgu tarihinden oluşmaktadır.
- Hastaya ait en fazla 8 adet aşı kaydı kart üzerinde saklanmaktadır. Her bir aşı kaydı 20 baytlık aşı adından ve 10 baytlık GG/AA/YYYY formatındaki aşı yapılış tarihinden oluşmaktadır.
- Hastaya ait en fazla 8 adet hastalık kaydı kart üzerinde saklanmaktadır. Her bir hastalık kaydı 20 baytlık hastalık adından ve 10 baytlık GG/AA/YYYY formatındaki bulgu tarihinden oluşmaktadır.
- Hastaya ait en fazla 8 adet sürekli kullandığı ilaç kaydı kart üzerinde saklanmaktadır. Her bir ilaç kaydı 20 baytlık ilaç adından ve 10 baytlık kullanım miktarı / dozaj bilgisinden oluşmaktadır.
- Hastanın geçirmiş olduğu en fazla 3 adet ameliyat da yine kart üzerinde tutulan sağlık bilgileri arasındadır. Her bir ameliyat için 20 bayttan oluşan ameliyat adı bilgisi, 10 bayttan oluşan GG/AA/YYYY formatındaki ameliyat tarihi bilgisi, 50 bayttan oluşan ameliyatın gerçekleştirildiği klinik/bölüm adı bilgisi ve 125 baytlık ameliyata ait açıklama bilgisi tutulmaktadır.
- Yukarıdaki kategorilere girmeyen diğer hasta sağlık bilgileri ise 255 baytlık bir metin alanında tutulmaktadır.

Hasta son muayene ve son reçete bilgileri:

Hastanın son muayenesine ait bilgiler 365 baytlık bir alanda saklanmakta olup bu bilgiler Tablo 4.4'te gösterilmiştir. Hasta son reçete bilgileri ise 366 baytlık bir alanda saklanmaktadır (Tablo 4.5).

Bilgi	**Uzunluk (Bayt)**	**Açıklama**
Tarih	10	GG/AA/YYYY formatında
Klinik / Bölüm	50	Muayenenin gerçekleştirildiği yer
Doktor ID	10	Muayene eden doktorun sistem geneli numarası
Doktor Adı	20	Muayene eden doktorun adı
Doktor Soyadı	20	Muayene eden doktorun soyadı
Açıklama	255	Muayene ile ilgili diğer özet bilgiler

Tablo 4.4 Kart üzerinde yer alan hasta son muayene bilgileri

Bilgi	**Uzunluk (Bayt)**	**Açıklama**
Tarih	10	GG/AA/YYYY formatında
Klinik / Bölüm	50	Reçetenin yazıldığı yer
Doktor ID	10	Reçeteyi yazan doktorun sistem geneli numarası
Doktor Adı	20	Reçeteyi yazan doktorun adı
Doktor Soyadı	20	Reçeteyi yazan doktorun soyadı
İçerik	255	Yazılan ilaçlar ve miktarları
Onay Durumu	1	Reçetenin hastane tarafından onaylı olup olmadığına dair bilgi

Tablo 4.5 Kart üzerinde yer alan hasta son reçete bilgileri

Kart sistem bilgileri:

Akıllı kart sağlık sisteminin çalışmasında kullanılan sistem bilgileri kart üzerinde toplam 1095 baytlık bir alanda tutulmakta olup bu bilgiler Tablo 4.6'da verilmiştir.

Bilgi	**Uzunluk (Bayt)**	**Açıklama**
Kart Veriliş Tarihi	10	GG/AA/YYYY formatında
Kart Son Güncelleme Tarihi	10	GG/AA/YYYY formatında
Merkez Veritabanının Adı	30	Hastaya ait bilgilerin tutulduğu sunucunun adı
Merkez Veritabanı RMI Adresi	21	Hastaya ait bilgilerin tutulduğu sunucunun ağ adresi. #[##].#[##].#[##].#[##][:#####] formatında
Hasta DES Anahtarı	1024	Hasta bilgilerinin şifrelenmesinde kullanılacak olan hastaya ait DES anahtarı

Tablo 4.6 Kart üzerinde yer alan sistem bilgileri

4.1.2.2 Hasta kartı yazılımı

Hasta kartı içerisinde çalışan sistem yazılımı *tr.edu.ege.ube.akss.hastakarti* olarak adlandırılan Java paketinden oluşmaktadır. Paket içerisinde istemci terminal uygulamaları ile iletişimde bulunan Java kart *"applet"*i ve hasta bilgilerinin saklanmasında kullanılan nesne sınıfları yer almaktadır. Teknoloji olarak Java kartın kullanılması ve uygulamada Java Card Çatısı'ndan (JCF) yararlanılması kart üzerinde çalışacak yazılımın tamamı ile nesneye dayalı olarak tasarımı imkanını vermiştir. Şekil 4.2'de pakette yer alan sınıfların ve bu sınıfların birbirleri ile olan ilişkilerinin yer aldığı nesne modeli gösterilmiştir.

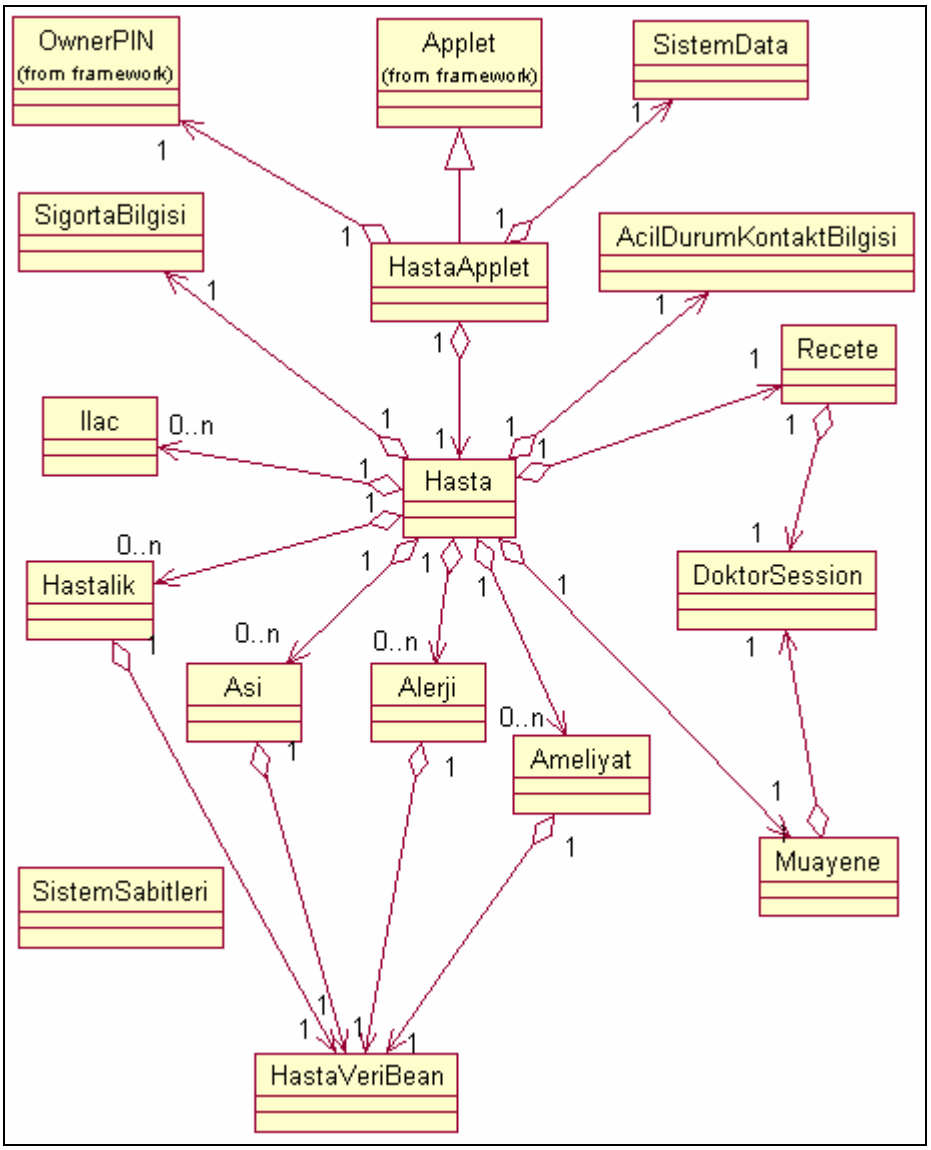
Nesne modelinden de anlaşıldığı gibi istemci ile iletişimde bulunacak olan Java kart *"applet"*i `javacard.framework.Applet` sınıfının bir uzantısı olan `HastaApplet`'dir. Genel tasarımda, hastaya ait bilgilerin `Hasta` adı verilen nesne tarafından, sistem bilgilerinin ise `SistemData` nesnesi tarafından saklanması düşünülmüştür. Buna göre `HastaApplet` nesnesi *birebir toplama (1 to 1 aggregation)* ilişkisi (association) ile kendi üzerinde bir `Hasta` ve bir `SistemData` nesnesini özellik olarak saklamaktadır (encapsulation). Bu iki nesne dışında, uygulama erişim şifresi de `HastaApplet` nesnesi içerisinde `javacard.framework.OwnerPIN` nesnesi olarak benzer şekilde saklanmaktadır.

İstemci uygulamadan gelen APDU'lar `HastaApplet`'in `process()` metodunda işlenmektedir. Örneğin hastanın sigorta bilgilerinin istenmesine dair APDU paketini *"applet"* aldığında bu paketin INS koduna bakarak isteği belirler ve bu isteği yerine getirecek olan özel (private) `getSigortaBilgisi()` metodunu çağırır. Bu metot içerisinde istek yerine getirilerek sonuç APDU nesnesi hazırlanıp istemci uygulamaya aktarılır. *"Applet"* içerisinde, tutulan her bilgi ile ilgili, bilgileri yazan (*set*) ve getiren (*get*) metotlar yer almaktadır. Şekil 4.3'te `HastaApplet` sınıfı tüm özellikleri, metotları ve ilişkileri ile beraber gösterilmiştir.

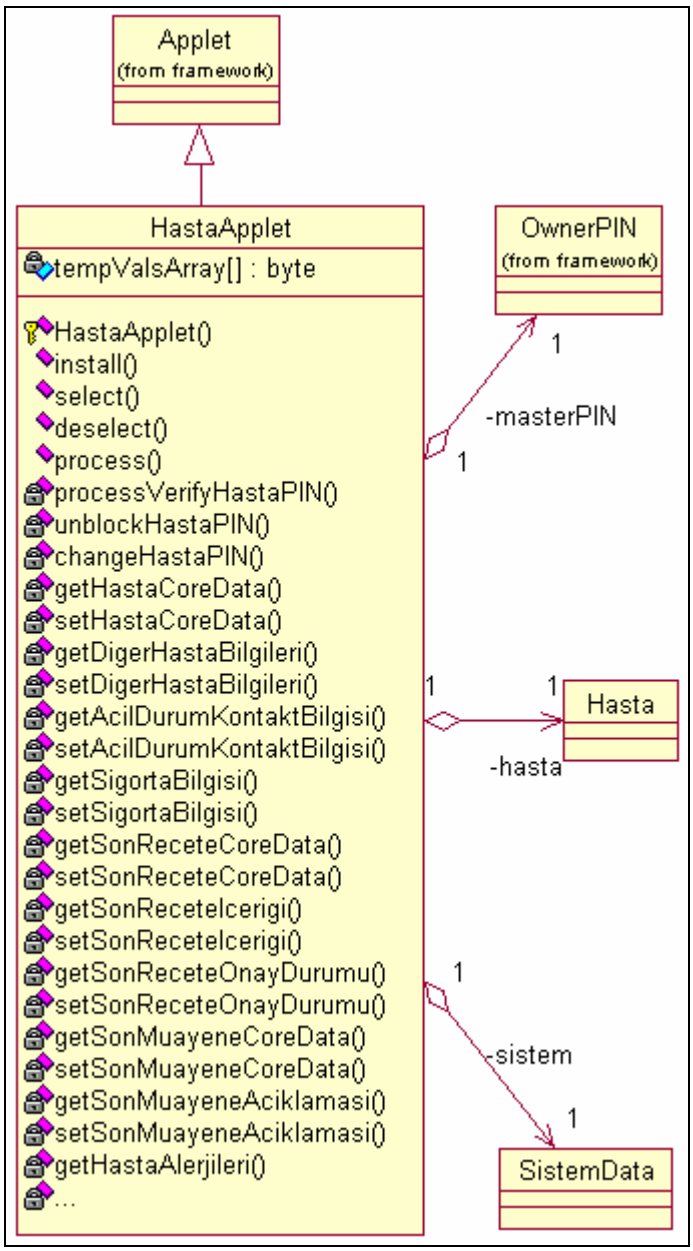
Hastaya ait tüm bilgiler sistemde `Hasta` sınıfından türetilen bir nesne içerisinde yer alır (Şekil 4.4). Hasta genel bilgileri bu nesnenin doğrudan birer özelliği iken acil durum kontak bilgisi, sigorta bilgisi ve sağlık bilgileri ayrı nesneler içerisinde saklanmakta ve `Hasta` nesnesi içerisinde bu nesneler özellik olarak barındırılmaktadır. Şekil 4.2'deki nesne modelinden de görüldüğü gibi `Hasta` nesnesi içerisinde birebir toplama ilişkisi ile `SigortaBilgisi`, `AcilDurumKontaktBilgisi`, `Recete` ve `Muayene` nesneleri saklanmaktadır.

Hastanın alerji, aşı, ameliyat, hastalık ve sürekli kullandığı ilaçlar ile ilgili bilgiler sırası ile `Alerji`, `Asi`, `Ameliyat`, `Hastalik` ve `Ilac` nesneleri içerisinde saklanmaktadır. `Hasta` nesnesi; bilgi türüne göre, içerisinde bu nesnelerin yer aldığı dizileri (array) özellik olarak

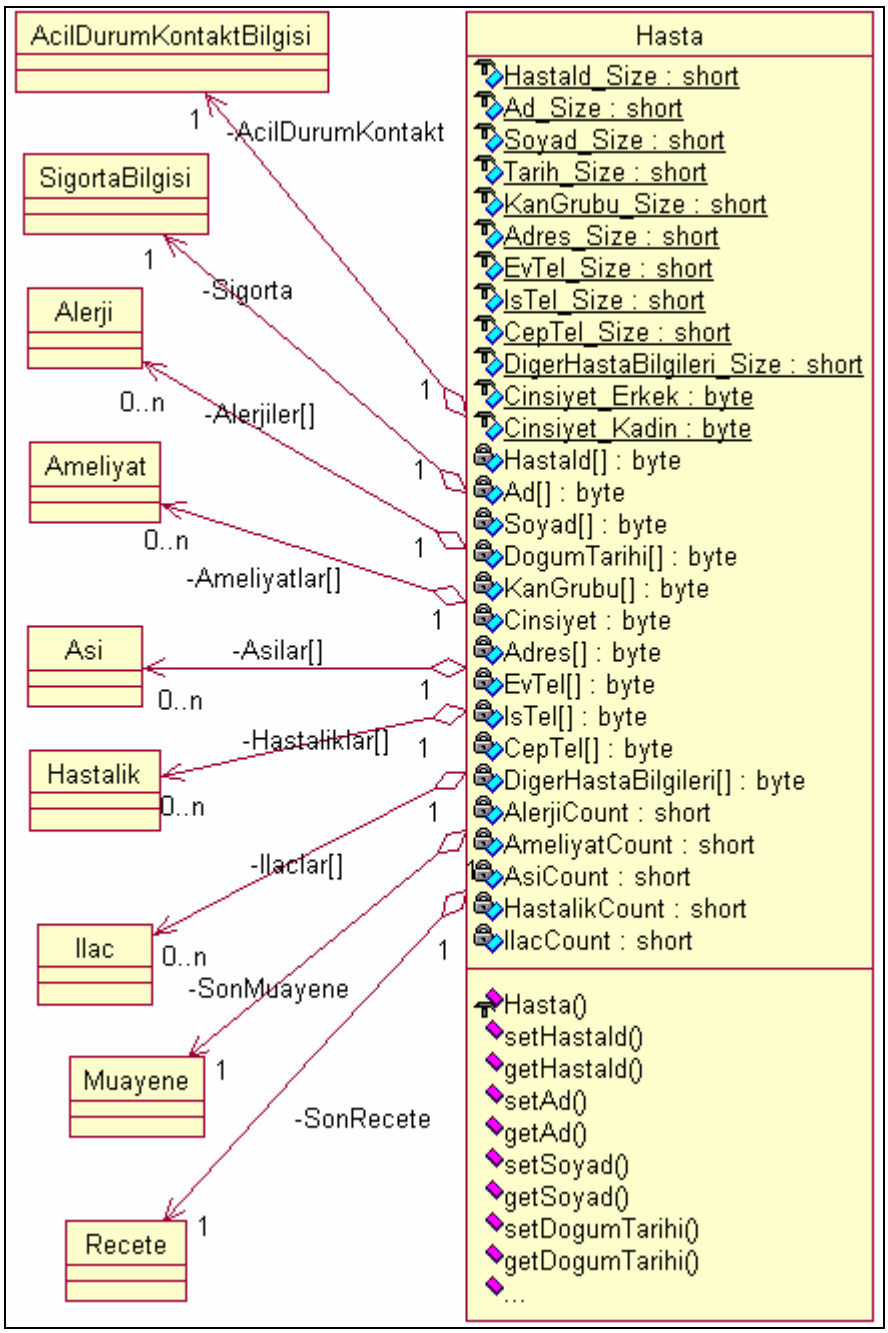
saklamaktadır. Örneğin hastanın geçirmiş olduğu bir ameliyat ile ilgili tüm bilgiler `Ameliyat` sınıfından türetilen bir nesnede saklanmaktadır. `Ameliyat` nesneleri ise `Hasta` nesnesi içerisindeki `Ameliyatlar` dizininde yer almaktadır. Benzer durum `Hasta` nesnesi ile 0..n toplama ilişkisinde bulunan diğer nesneler için de geçerlidir.



Şekil 4.2 Hasta kartı yazılımı nesne modeli

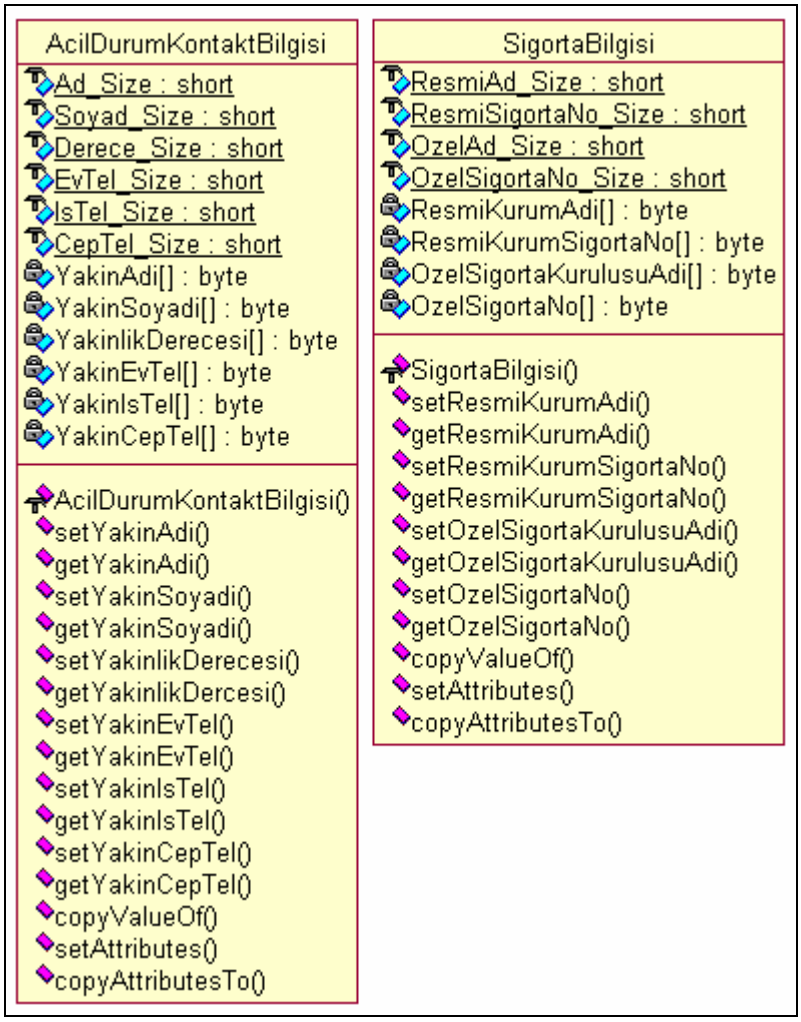


Şekil 4.3 HastaApplet sınıfı



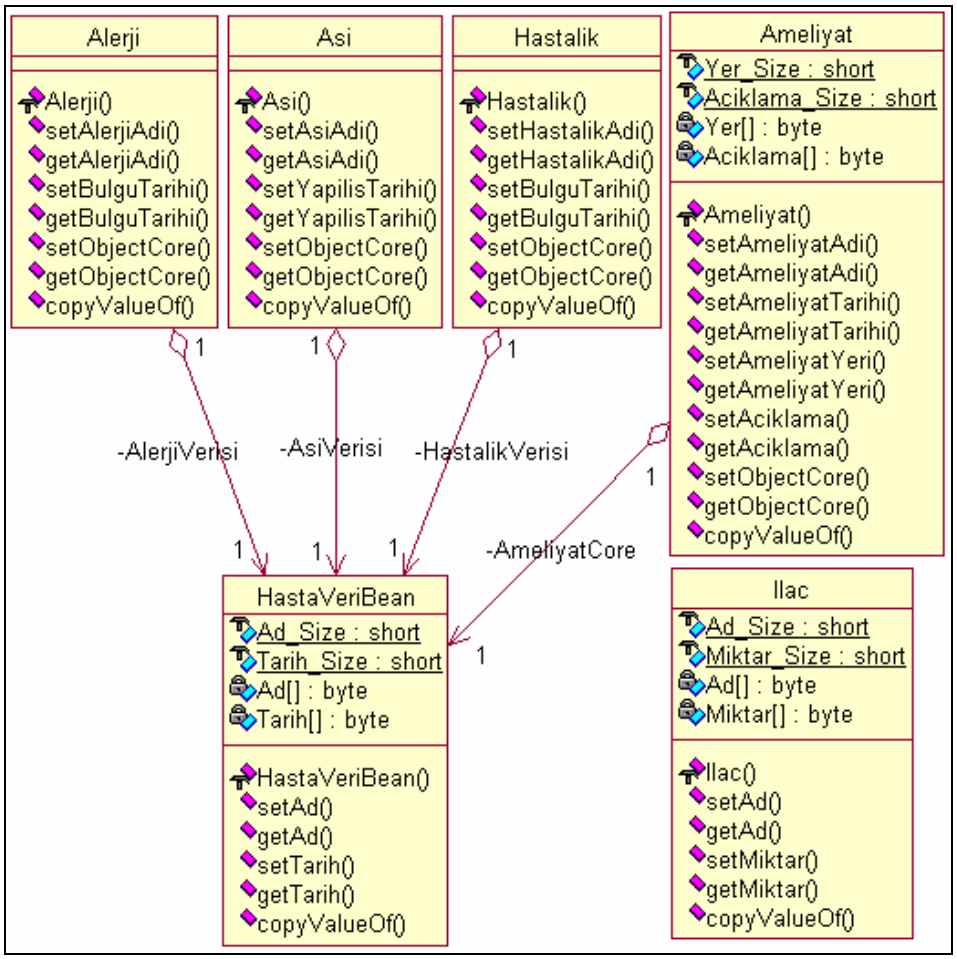
Şekil 4.4 Hasta sınıfı

Şekil 4.5'te özellik ve metotları verilen `AcilDurumKontakt` ve `SigortaBilgisi` sınıflarından türetilen nesneler adlarından da anlaşılacağı gibi acil durumlarda iletişime geçilecek hasta yakını ile ilgili bilgileri ve hastanın resmi ve özel sigortasına ait bilgileri saklamaktadırlar.



Şekil 4.5 AcilDurumKontaktBilgisi ve SigortaBilgisi sınıfları

Herhangi bir sağlık bilgisi için ortak olan *ad* ve *tarih* alanlarının `HastaVeriBean` adı verilen sınıftan türetilen nesnelerde yer alması tasarlanmıştır. `Alerji`, `Asi`, `Hastalik` ve `Ameliyat` nesneleri de bu `HastaVeriBean` nesnelerinden birer örneği içermektedirler. Şekil 4.6'da hastaya ait sağlık bilgilerini içeren nesne sınıfları gösterilmiştir.



Şekil 4.6 Hasta sağlık bilgilerini içeren sınıflar

Hastaya ait son muayene ve reçeteye ait bilgiler sırası ile `Muayene` ve `Recete` nesnelerinde saklanmaktadır. Her iki nesne için de ortak olan

doktor oturumu bilgilerini `DoktorSession` sınıfından türetilen nesneler içerir (Şekil 4.7).

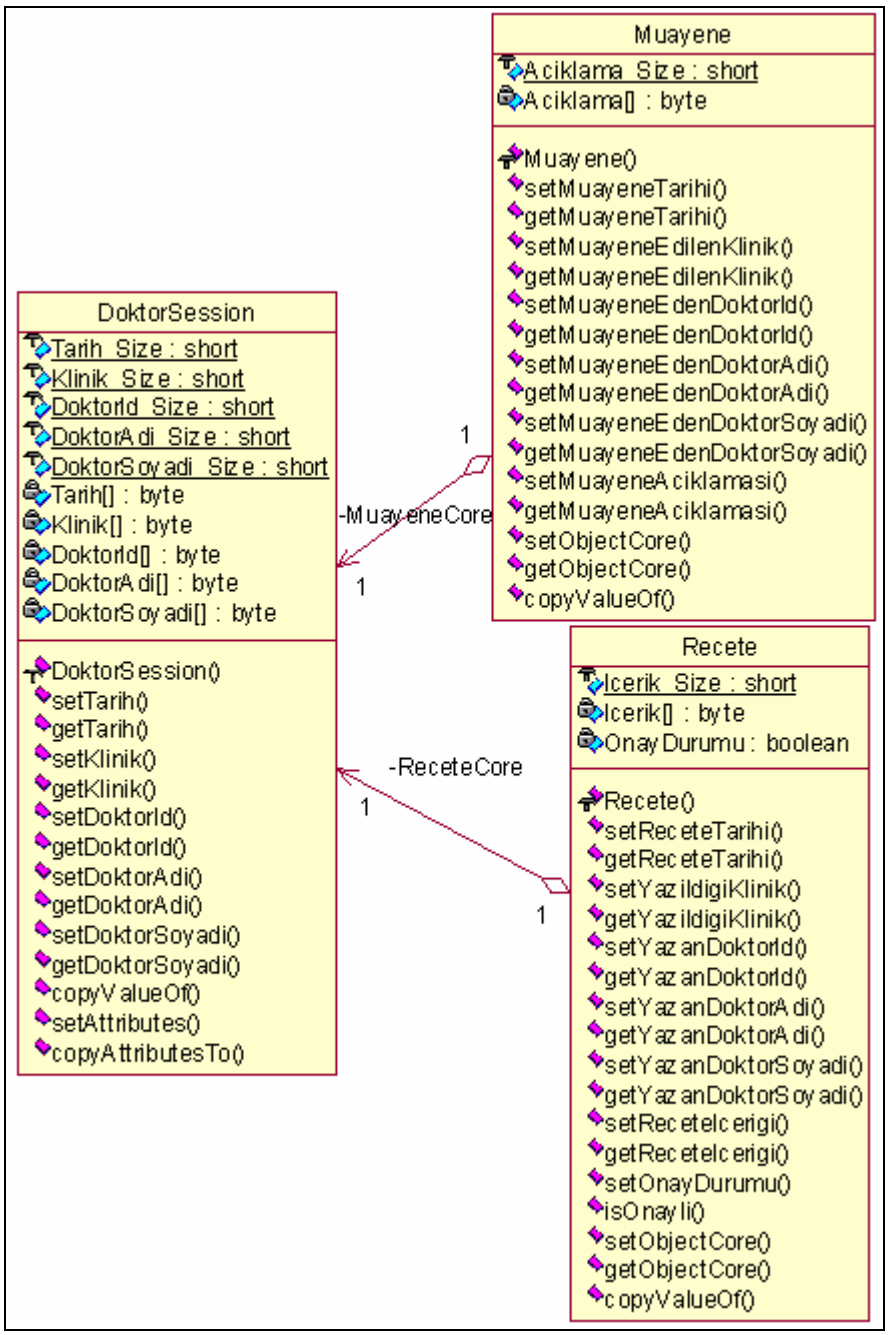
Sağlık sisteminin işleyişi ile ilgili bilgiler ise paket içerisinde yer alan SistemData sınıfından türetilen nesne de saklanmaktadır (Şekil 4.8). Bu nesne de tıpkı Hasta ve OwnerPIN nesneleri gibi HastaApplet nesnesinin bir özelliğidir.

Kart yazılımının terminal uygulamaları ile iletişimi:

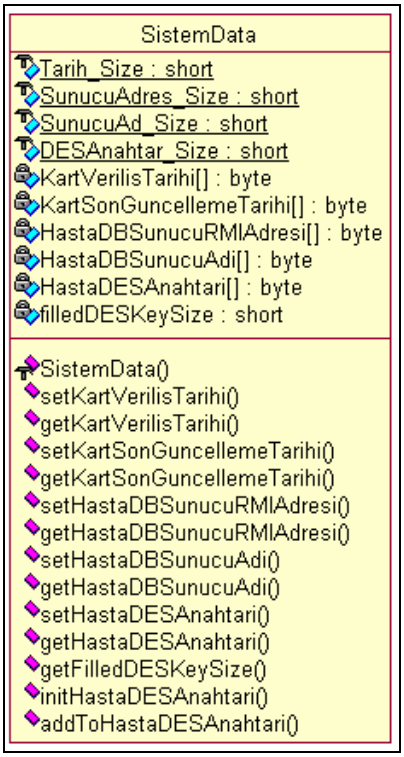
Hasta kartı yazılım paketi dosyaları CAP formatında karta transfer edildikten ve yüklendikten sonra kart içerisinde `HastaApplet` nesnesi ve içerdiği diğer tüm nesneler kartın EEPROM'unda kalıcı (persistent) nesneler olarak yaşamaya başlarlar.

Yazılım paketinin AID'si "85 66 69 65 75 83 83 00 00 00 00 00 00 00 00 01", içerdiği Java kart *"applet"*inin (`HastaApplet`) AID'si de "85 66 69 65 75 83 83 00 00 00 00 00 00 00 00 02" olarak belirlenmiştir. Hasta kartı kart okuyucuya yerleştirildikten sonra istemci terminal uygulaması JCRE'ye bu *"applet"* AID'sini bildirerek `HastaApplet` nesnesinin seçili hale gelmesini sağlar. Bu aşamadan sonra terminal uygulaması ile kart üzerindeki `HastaApplet` nesnesi arasında APDU paketleri aracılığı ile bilgi alışverişi başlatılır. Tüm Komut APDU'ların CLA kodları 0x90h olarak belirlenmiştir. Bu APDU'lara ait INS kodları ise Tablo 4.7'de verilmiştir.

Tablo 4.7'de verilen INS kodları ve diğer kart içi sabit değerler paket içerisindeki `SistemSabitleri` sınıfında özellik olarak yer almaktadır.



Şekil 4.7 Muayene ve Recete sınıfları



Şekil 4.8 SistemData sınıfı

Terminalden gelen komut APDU'ların INS kodları `HastaApplet`'in `process()` metodunda kontrol edilerek hangi işlemin yerine getirileceği belirlenir ve bu işlemi yerine getirecek metot çağrılır. Metot içerisinde terminale gidecek bilgiyi içeren sonuç APDU'su hazırlanır ve gönderilir.

Komut Adı	**INS Kodu (Hex değeri olarak)**
COMMAND_VERIFY_HASTA_PIN	0x10
COMMAND_CHANGE_HASTA_PIN	0x12
COMMAND_UNBLOCK_HASTA_PIN	0x14
COMMAND_GET_HASTA_CORE_DATA	0x16
COMMAND_SET_HASTA_CORE_DATA	0x18
COMMAND_GET_DIGER_HASTA_BILGILERI	0x20
COMMAND_SET_DIGER_HASTA_BILGILERI	0x22
COMMAND_GET_ACIL_DURUM_KONTAKT_BILGISI	0x24
COMMAND_SET_ACIL_DURUM_KONTAKT_BILGISI	0x26
COMMAND_GET_SIGORTA_BILGISI	0x28
COMMAND_SET_SIGORTA_BILGISI	0x30
COMMAND_GET_SON_RECETE_CORE_DATA	0x32
COMMAND_SET_SON_RECETE_CORE_DATA	0x34
COMMAND_GET_SON_RECETE_ICERIGI	0x36
COMMAND_SET_SON_RECETE_ICERIGI	0x38
COMMAND_GET_SON_RECETE_ONAY_DURUMU	0x40
COMMAND_SET_SON_RECETE_ONAY_DURUMU	0x42
COMMAND_GET_SON_MUAYENE_CORE_DATA	0x44
COMMAND_SET_SON_MUAYENE_CORE_DATA	0x46
COMMAND_GET_SON_MUAYENE_ACIKLAMASI	0x48
COMMAND_SET_SON_MUAYENE_ACIKLAMASI	0x50
COMMAND_GET_HASTA_ALERJILERI	0x52
COMMAND_SET_HASTA_ALERJILERI	0x54
COMMAND_GET_HASTA_ASILARI	0x56
COMMAND_SET_HASTA_ASILARI	0x58
COMMAND_GET_HASTA_HASTALIKLARI	0x61
COMMAND_SET_HASTA_HASTALIKLARI	0x62

COMMAND_GET_HASTA_ILACLARI	0x64
COMMAND_SET_HASTA_ILACLARI	0x66
COMMAND_GET_AMELIYAT_BILGISI	0x68
COMMAND_SET_AMELIYAT_BILGISI	0x70
COMMAND_GET_KART_VERILIS_TARIHI	0x72
COMMAND_SET_KART_VERILIS_TARIHI	0x74
COMMAND_GET_KART_SON_GUNCELLEME_TARIHI	0x76
COMMAND_SET_KART_SON_GUNCELLEME_TARIHI	0x78
COMMAND_GET_HASTA_DB_SUNUCU_RMI_ADRESI	0x80
COMMAND_SET_HASTA_DB_SUNUCU_RMI_ADRESI	0x82
COMMAND_GET_HASTA_DB_SUNUCU_ADI	0x84
COMMAND_SET_HASTA_DB_SUNUCU_ADI	0x86
COMMAND_GET_HASTA_DES_KEY	0x88
COMMAND_SET_HASTA_DES_KEY	0x89

Tablo 4.7 Hasta kartı komut APDU'larına ait INS kodları

Komut APDU'lara göz atıldığında ilk üç INS dışındakilerin standart bilgi alma ve yazma komutları oldukları anlaşılmaktadır. İlk üç komut ise hasta bilgilerine erişim şifresini içeren `OwnerPIN` nesnesi ile ilgilidir. 0x10h kodlu komut APDU terminal tarafından veri olarak gönderilen şifreyi içerir. Bu şifre `OwnerPIN` nesnesinde saklanan kart PIN'i ile karşılaştırılır. Şifre aynı ise terminale 0x90h yani PIN doğrulamanın başarı ile gerçekleştirildiğine dair yanıt iletilir. Eğer girilen PIN yanlış ise `OwnerPIN` nesnesi deneme sayısı özelliğini bir azaltır ve kalan deneme hakkını da bildiren hata kodunu terminale iletir. Uygulama hasta kartına erişimde arka arkaya 2 kez yanlış PIN girme hakkı verilmiştir. 3. denemede de girilen PIN yanlış ise kart uygulaması kendini kilitler. Bloke olmuş kartların ise sistem yönetim birimi tarafından açılması tasarlanmıştır. Yeni PIN bilgisini içeren 0x14h kodlu komut APDU, bloke olmuş kartı açar ve gelen PIN'i yeni kart PIN'i olarak belirler. Benzer şekilde 0x12h kodlu komut APDU ise kullanıcıların kart PIN'lerini değiştirmelerine imkan vermektedir.

4.1.3 Doktor akıllı kartı

4.1.3.1 Kart üzerinde saklanan bilgiler

Doktor kartı üzerinde saklanan bilgiler hasta kartındaki kadar fazla miktarda olmayıp doktora ait genel bilgiler ve sağlık sisteminin çalışmasında kullanılan sistem bilgilerinden oluşmaktadır. Tutulan toplam bilgi miktarı *1333* bayttır.

Doktor genel bilgileri:

Kart üzerinde toplam 238 baytta saklanan doktor genel bilgileri Tablo 4.8'de gösterilmiştir.

Bilgi	**Uzunluk (Bayt)**	**Açıklama**
Doktor ID	11	Doktoru tanımlayan sistem geneli numara
PIN	4	Kart kullanım şifresi
Ad	20	
Soyad	20	
Bölüm	50	Doktorun çalıştığı klinik veya bölümün adı
Adres	100	
Ev Telefonu	11	###-####### formatında (alan kodu – numara)
İş Telefonu	11	###-####### formatında (alan kodu – numara)
Cep Telefonu	11	###-####### formatında (alan kodu – numara)

Tablo 4.8 Kart üzerinde yer alan doktor genel bilgileri

Kart sistem bilgileri:

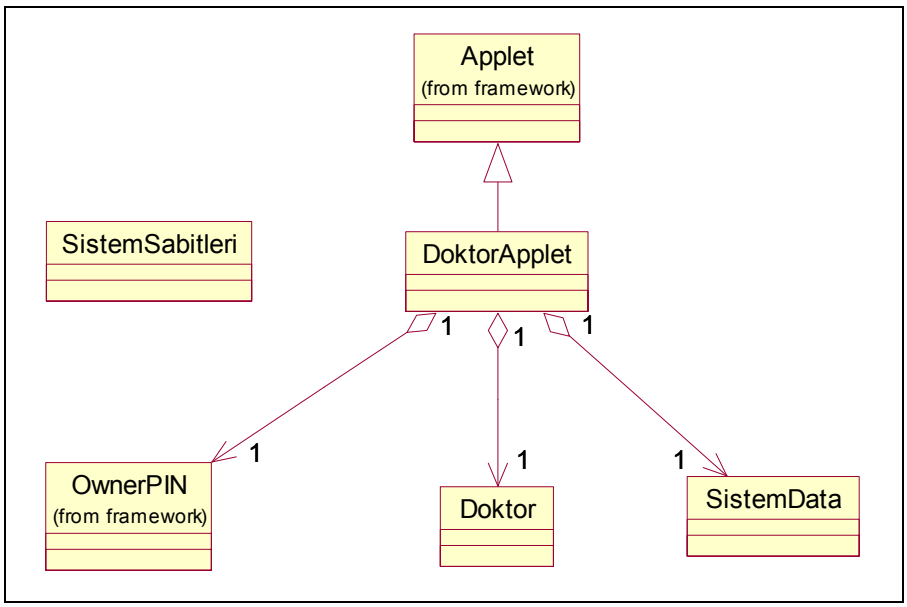
Akıllı kart sağlık sisteminin çalışmasında kullanılan sistem bilgileri kart üzerinde toplam 1095 baytlık bir alanda tutulmakta olup bu bilgiler Tablo 4.9'da verilmiştir.

Bilgi	**Uzunluk (Bayt)**	**Açıklama**
Kart Veriliş Tarihi	10	GG/AA/YYYY formatında
Kart Son Güncelleme Tarihi	10	GG/AA/YYYY formatında
Merkez Veritabanının Adı	30	Doktora ait bilgilerin tutulduğu sunucunun adı
Merkez Veritabanı RMI Adresi	21	Doktora ait bilgilerin tutulduğu sunucunun ağ adresi. #[##].#[##].#[##].#[##][:#####] formatında
Doktor DSA Özel Anahtarı	1024	Doktorun verileri imzalamada kullanacağı ve sistemde doktorun yetkilendirilmesini sağlayan özel anahtar

Tablo 4.9 Kart üzerinde yer alan sistem bilgileri

4.1.3.2 Doktor kartı yazılımı

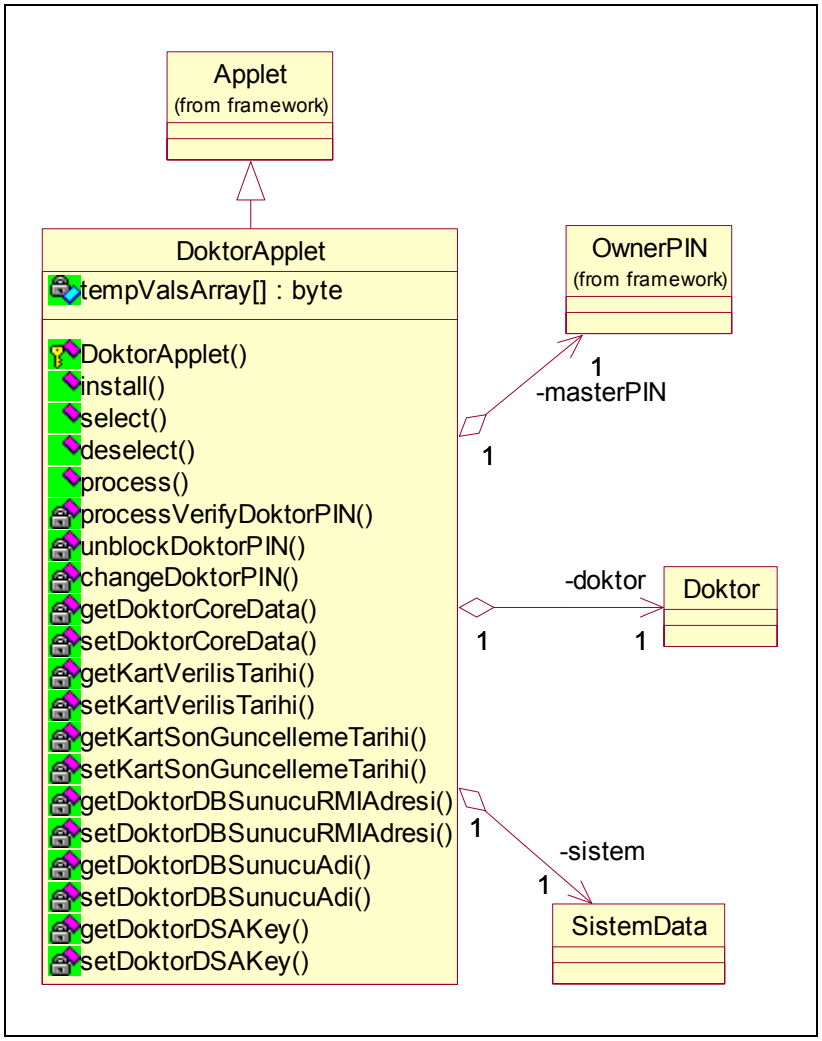
Doktor kartı içerisinde çalışan sistem yazılımı *tr.edu.ege.ube.akss.doktorkarti* olarak adlandırılan Java paketinden oluşmaktadır. Paket içerisinde istemci terminal uygulamaları ile iletişimde bulunan Java kart *"applet"*i ve doktor bilgilerinin saklanmasında kullanılan nesne sınıfları yer almaktadır. Hasta kartı yazılımı ile karşılaştırıldığında oldukça basit bir nesne tasarıma sahip olan paketin nesne modeli Şekil 4.9'da gösterilmiştir.



Şekil 4.9 Doktor kartı yazılımı nesne modeli

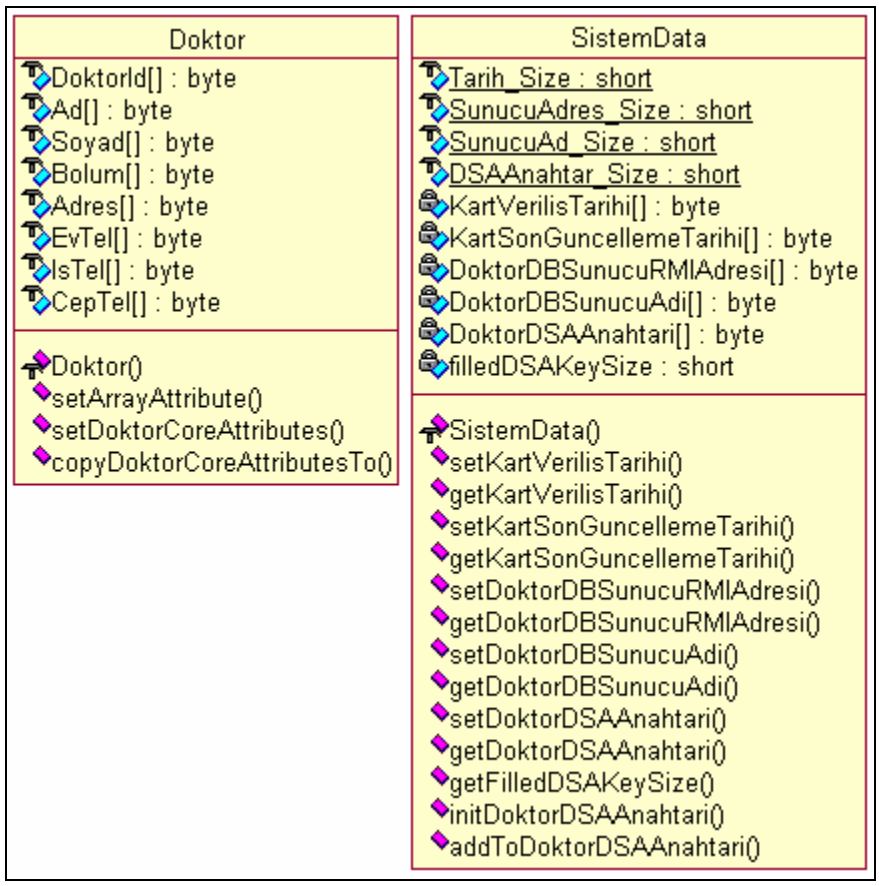
Terminaller ile iletişimde bulunacak olan Java kart *"applet"*i `javacard.framework.Applet` sınıfının bir uzantısı olan `DoktorApplet`'dir. Doktor genel bilgileri `Doktor` nesnesinde saklanırken, kart sistem bilgileri ise `SistemData` nesnesinde saklanmaktadır. Kart erişim şifresini ise Java Card çatısında yer alan `OwnerPIN` sınıfından türetilen nesne tutar.

Şekil 4.10'da görülen `DoktorApplet` sınıfından türetilen nesne hasta kartı yazılımındaki `HastaApplet` nesnesine benzer şekilde modelde görev alır.



Şekil 4.10 DoktorApplet sınıfı

Şekil 4.11'de ise doktor genel bilgilerinin saklandığı `Doktor` nesnesi ve sistem bilgilerinin saklandığı `SistemData` nesnelerinin türetildiği sınıflar gösterilmiştir.



Şekil 4.11 Doktor ve SistemData sınıfları

Kart yazılımının terminal uygulamaları ile iletişimi:

Doktor kartı yazılım paketi dosyaları CAP formatında karta transfer edildikten ve yüklendikten sonra kart içerisinde `DoktorApplet` nesnesi ve içerdiği diğer tüm nesneler kartın EEPROM'unda kalıcı nesneler olarak yaşamaya başlarlar.

Yazılım paketinin AID'si "85 66 69 65 75 83 83 68 81 00 00 00 00 00 00 01", içerdiği Java kart appletinin (`DoktorApplet`) AID'si de "5 66 69 65 75 83 83 68 81 00 00 00 00 00 00 02" olarak belirlenmiştir. Hasta

kartında olduğu gibi tüm komut APDU'ların CLA kodları 0x90h olarak belirlenmiştir. Bu APDU'lara ait INS kodları ise Tablo 4.10'da verilmiştir.

Komut Adı	**INS Kodu (Hex değeri olarak)**
COMMAND_VERIFY_DOKTOR_PIN	0x10
COMMAND_CHANGE_DOKTOR_PIN	0x12
COMMAND_UNBLOCK_DOKTOR_PIN	0x14
COMMAND_GET_DOKTOR_CORE_DATA	0x16
COMMAND_SET_DOKTOR_CORE_DATA	0x18
COMMAND_GET_KART_VERILIS_TARIHI	0x20
COMMAND_SET_KART_VERILIS_TARIHI	0x22
COMMAND_GET_KART_SON_GUNCELLEME_TARIHII	0x24
COMMAND_SET_KART_SON_GUNCELLEME_TARIHI	0x26
COMMAND_GET_DOKTOR_DB_SUNUCU_RMI_ADRESI	0x28
COMMAND_SET_DOKTOR_DB_SUNUCU_RMI_ADRESI	0x30
COMMAND_GET_DOKTOR_DB_SUNUCU_ADI	0x32
COMMAND_SET_DOKTOR_DB_SUNUCU_ADI	0x34
COMMAND_GET_DOKTOR_DSA_KEY	0x36
COMMAND_SET_DOKTOR_DSA_KEY	0x38

Tablo 4.10 Doktor kartı komut APDU'larına ait INS kodları

Tablo 4.10'da verilen INS kodları ve diğer kart içi sabit değerler paket içerisindeki `SistemSabitleri` sınıfında özellik olarak yer almaktadır.

4.2 Kart İstemci Ara Yazılımı

Akıllı Kart Sağlık Sistemi'nde akıllı kartlar ile iletişimde bulunan ve istemci bilgisayar üzerinde çalışması tasarlanan yazılım bileşenleri *tr.edu.ege.ube.akss.client* paketinde toplanmıştır. Bu bölümde bu ara yazılım ve bileşenleri hakkında bilgi verilmiştir.

4.2.1 Kart istemci ara yazılımının sistemdeki rolü

Kart istemci yazılımı adından da anlaşılacağı gibi üzerinde çalıştığı terminale bağlı olan kart kabul cihazlarına yerleştirilen akıllı kartlar ile iletişimi sağlamaktadır. Yazılımın hazırlanmasında OpenCard Çatısı'ndan (OCF) yararlanılmıştır.

Sistem mimarisi bölümünde de anlatıldığı gibi OpenCard Çatısı aracılığı ile bilgisayara bağlı bulunan kart kabul cihazlarının tanınması ve cihaz özelliklerine erişim ve gerekli kart iletişim servislerinin kullanılması mümkün olmaktadır.

Sistem kullanıcıları ile etkileşimde bulunan ara yüz yazılımı bileşenlerinin de kart terminallerine erişimleri, APDU paketleri aracılığı ile akıllı kartlar ile iletişimleri ve bilgisayar üzerindeki akıllı kart ortamını yönetmeleri mümkündür. Ancak bu sistem genelinde uygulanan MVC mimarisine aykırıdır ve sistemin modülerliğini ortadan kaldırır. Böyle bir durumda ara yüz bileşenlerinin, kart işlemlerini yapan nispeten alt seviye işlemler ile de uğraşmaları gerekir. Bu nedenlerden dolayı kart istemci bileşenlerinin ayrı olarak tasarlanması ve bu bileşenlerin sistem kullanıcıları ile etkileşimde bulunan ara yüz tarafından kullanılması sağlanmıştır. Böylelikle ara yüz bileşenleri (dolaysıyla bunları hazırlayan ara yüz geliştiriciler) APDU iletişimi gibi alt seviye kart işlemlerinden soyutlanmışlardır.

4.2.2 Kart istemci ara yazılımının mimarisi

İstemci ara yazılımı paketinde yer alan sınıflar kart ortamının bilgisayar üzerinde başlatılmasında, terminale yerleştirilen akıllı kartlar üzerindeki uygulamaların ("applet") seçilmesinde ve bu uygulamalar ile APDU paketleri aracılığıyla iletişimde görev alırlar.

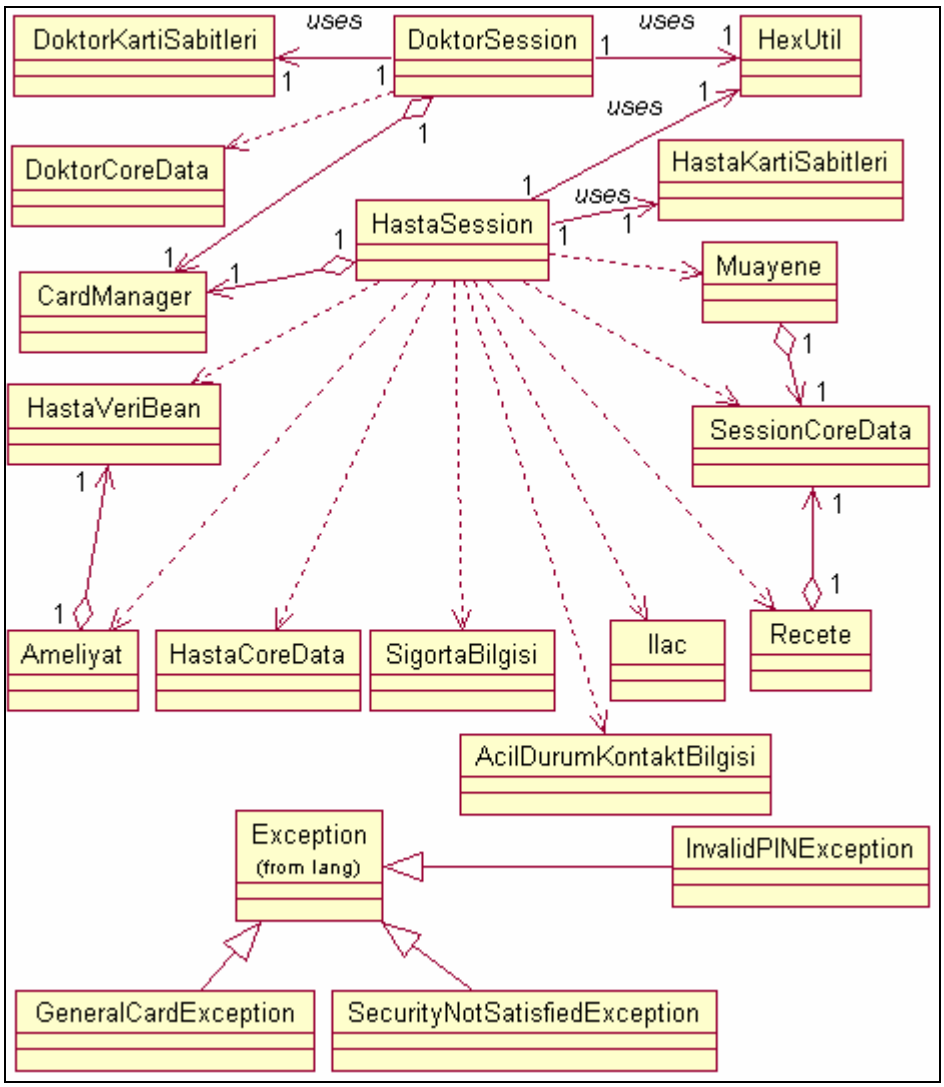
Şekil 4.12'de nesne modeli gösterilen yazılım paketinde en önemli görevleri `CardManager`, `HastaSession` ve `DoktorSession` sınıfları üstlenmektedir. Aşağıdaki alt başlıklar içerisinde bu sınıflardan türetilen nesnelerin sistem içerisindeki rolleri, ve pakette yer alan diğer nesneler ile etkileşimleri hakkında bilgiler yer almaktadır. Son alt başlıkta ise hata kontrolünde görev alan sınıflar açıklanmıştır.

4.2.2.1 CardManager sınıfı

Şekil 4.13'de gösterilen CardManager sınıfından türetilen nesne sistem içinde başlıca şu fonksiyonları yerine getirmektedir:

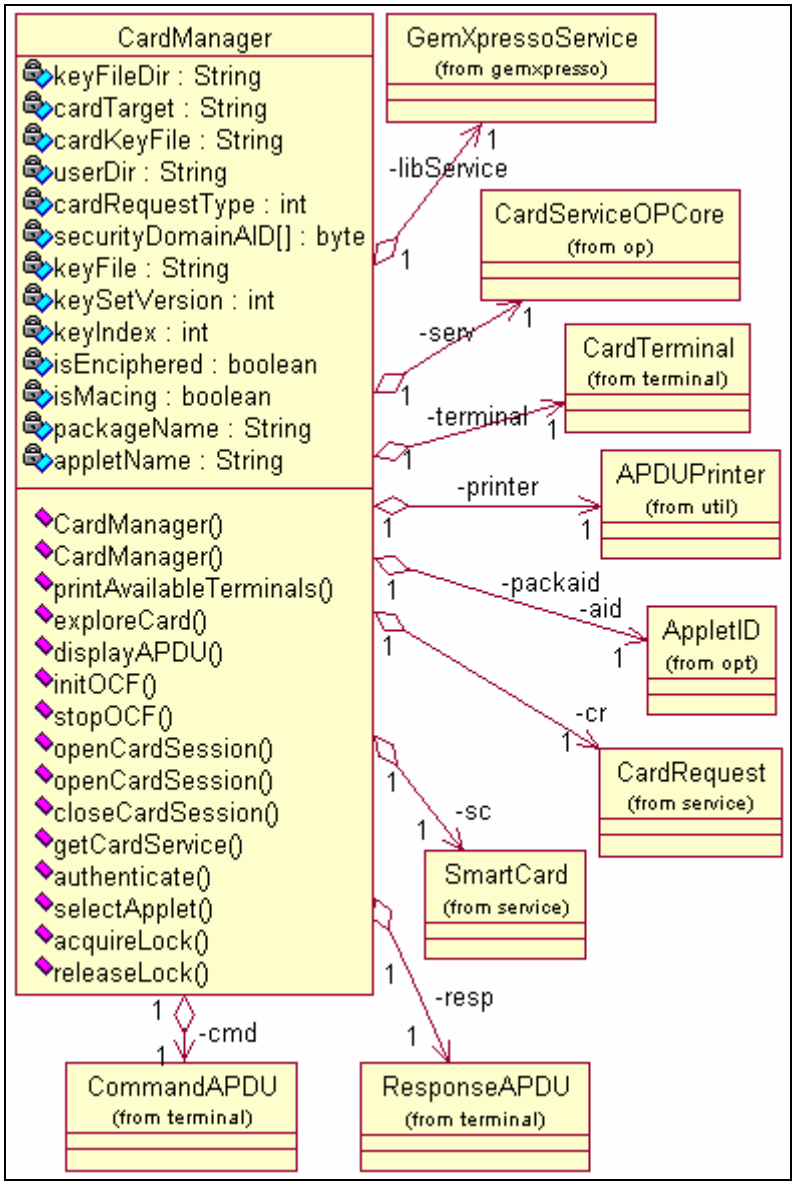
- Bilgisayar üzerinde akıllı kart ortamının başlatılması ve sonlandırılması
- Akıllı kartlar ile güvenli iletişim kanalının kurulması ve kapatılması
- İletişimde bulunulacak akıllı kart "applet"lerinin seçilmesi
- Kart oturumlarının başlatılması ve sonlandırılması
- Kart "applet"leri ile APDU aracılığı ile iletişimin sağlanması

CardManager nesnesi, sistemde oluşturulurken ilgili özellik dosyasından iletişimde bulunacağı kart okuyucu ünitesi (veya ünitelerine) ilişkin bilgileri alır ve akıllı kart ortamını başlatır. Ayrı bir özellik dosyasından ise kart iletişiminde kullanacağı servisler ve kart tipleri ile ilgili bilgileri alır ve göreve hazır hale gelir.



Şekil 4.12 Kart İstemci Ara Yazılımı Nesne Modeli

Nesnenin kart oturumlarını açması, hem olaya dayalı hem de uygulamaya dayalı modelleri destekleyecek şekilde tasarlanmıştır. Uygulamaya dayalı model kullanılmak istenirse `CardManager` nesnesi ve bu nesne ile haberleşen diğer nesneler kart okuyucuya yeni bir kart takılana kadar bloke olmaktadır ki bu tek iş parçacıklı ve görsel ara yüz bileşenleri kullanmayan uygulamalar için uygundur.



Şekil 4.13 CardManager sınıfı

Olaya dayalı model de ise daha önce de bahsedildiği gibi olay dinleme mekanizmaları kullanılmaktadır. Tez kapsamında yer alan uygulamada da bu model kullanılmıştır. Buna göre `CardManager`, akıllı

kartın kart terminaline yerleştirilmesi olayından haberdar olur olmaz ilgili kart oturumunu başlatmaktadır.

Kart oturumu başlatıldıktan sonra iletişimde bulunulacak akıllı kart sistem üzerinde `CardManager`'in `SmartCard` nesnesi ile temsil edilir. Tüm iletişimlerde bundan sonra bu nesne ve sunduğu kart servisleri görev alacaktır.

Oturum başlatıldıktan sonra üst seviye API kütüphane nesnesi olan `GemXpressoService` nesnesine `SmartCard` nesnesinden elde edilen servis özellik olarak aktarılır ve bundan sonraki kart iletişimlerinde bu servis (`CardServiceOPCore` nesnesi) kullanılır.

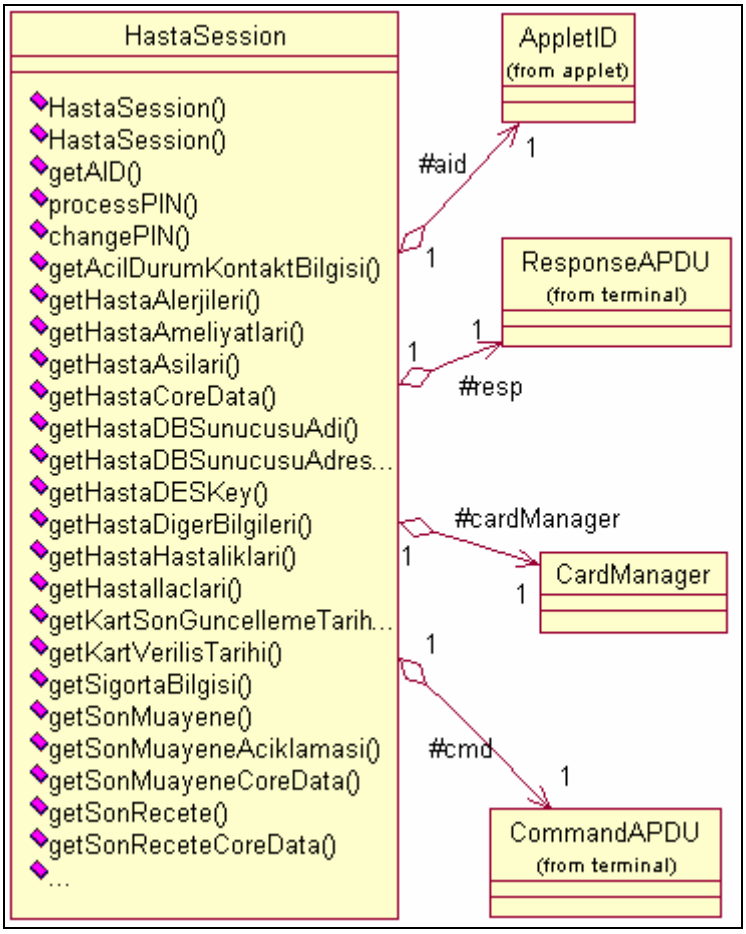
4.2.2.2 HastaSession sınıfı

Şekil 4.14'te gösterilen `HastaSession` sınıfı, hasta kartları ile iletişimde gerekli olan tüm işlemleri yerine getiren sınıftır.

Hasta kartı oturumu açıldıktan sonra hasta kartı ile bilgi alışverişinde gerekli olan tüm APDU paketlerinin hazırlanıp gönderilmesi ve alımı bu sınıftan türetilen nesne tarafından gerçekleştirilir. Servis olarak `CardManager`'in sunduğu kart servisi kullanılır.

Karta bilgi gönderiminde ya da karttan bilgi alımında `HastaSession` nesnesine ait metotların kullanılması kart iletişimi alt yapısından soyutlanmayı sağlamaktadır. Örneğin hastaya ait genel bilgilerin (id, ad, soyad, adresi vb.) karttan elde edilmesindeki tüm APDU alışverişini bu nesneye ait `getHastaCoreData()` metodu yerine getirir. Böylelikle bu bilgiye ihtiyaç duyan istemci ara yüz bileşeni nesnenin, kart iletişimi ve on altılı kodlardan oluşan bayt dizinleri ile uğraşmasına gerek kalmamaktadır. Bu metot içerisinde; gidecek olan komut APDU nesnesi hazırlanır, kart servisi aracılığı ile APDU karta gönderilir ve karttan gelen cevap değerlendirilir. Cevap APDU'daki hasta

bilgilerinin on altılı bayt dizinlerinden *int, String*, vb. veri tiplerine dönüştürülmesi ve ara yüz bileşeninin kullanımına sunulması da yine bu metot içerisinde gerçekleştirilir.



Şekil 4.14 HastaSession sınıfı

Aynı durum karta bilgi yazılmasında da geçerlidir: Ara yüz bileşeni kullanıcıdan aldığı hasta genel bilgilerini `HastaSession` nesnesinin ilgili metoduna parametre olarak gönderir. Bu metot içerisinde gelen hasta bilgileri on altılı kodlara dönüştürülür, gidecek olan APDU hazırlanır, kart servisi aracılığı ile paket karta gönderilir ve kartın yanıtı alınır. Yanıt da metodu çağıran ara yüz bileşenine aktarılır.

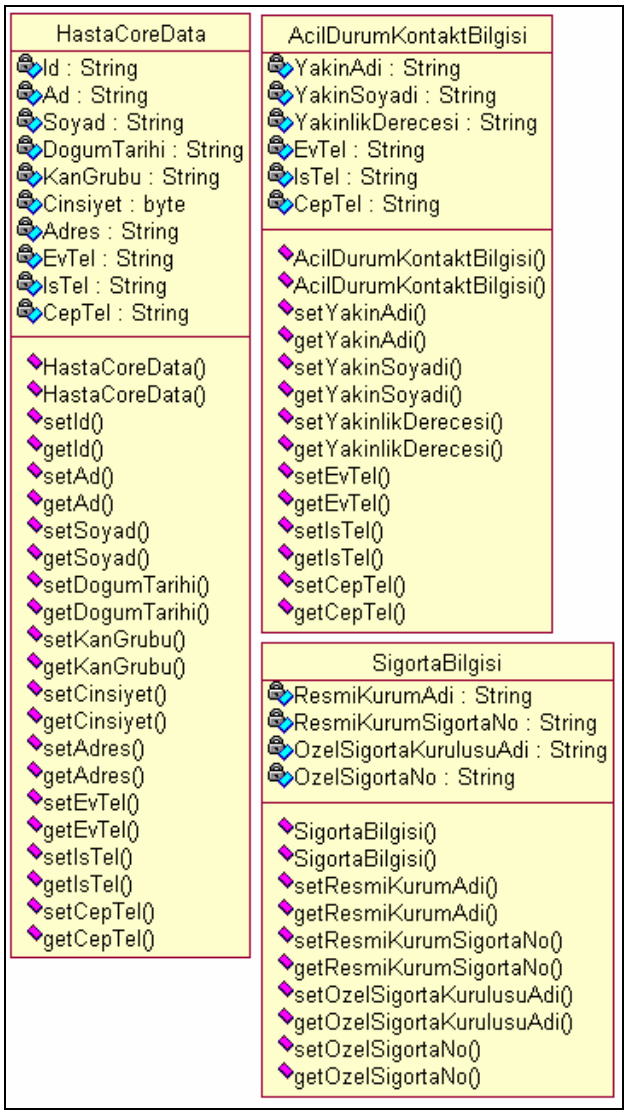
Yukarıda anlatılan mekanizma karttan tüm bilgi alımlarında ve karta bilgi yazılmasında rol oynamaktadır. Genel tasarım; ara yüz bileşeninin karttan bilgi istediğinde, `HastaSession` nesnesinin metodunu çağırarak bu bilgileri, sistem yazılımına özel nesnelerin özelliği olarak elde etmesi ve karta bilgi yazımında da yine özel nesnelerin sakladığı bilgileri ilgili `HastaSession` metotlarına parametre olarak göndererek karta yazması şeklindedir.

Şekil 4.15'te gösterilen `HastaCoreData`, `AcilDurumKontaktBilgisi` ve `SigortaBilgisi` sınıflarından türetilen nesneler, ilgili hasta bilgilerinin `HastaSession` nesnesi ile ara yüz bileşen nesneleri arasında iletişiminde görev alırlar. Örneğin yukarıda da belirtildiği gibi hasta genel bilgilerini isteyen ara yüz bileşenine yanıt olarak (eğer bir hata durumu mevcut değilse) bu bilgileri barındıran `HastaCoreData` nesnesi döndürülür. Karta yazılmak istenen yeni hasta genel bilgileri de yine bu nesne altında `HastaSession` nesnesine iletilirler.

Hasta alerji, aşı ve hastalık bilgileri sistemde birer `HastaVeriBean` nesnesi ile temsil edilirler. Örneğin hasta alerjileri karttan istendiğinde, `HastaSession` nesnesi her bir alerji kaydına ait bilgileri bir `HastaVeriBean` nesnesine özellik olarak atamakta ve bu `HastaVeriBean` nesnelerinden oluşan koleksiyonu istemci bileşene iletmektedir. Bu bilgilerin koleksiyondan alınıp kullanıcıya görüntülenmesi ise istemci konumdaki ara yüz bileşeninin görevidir. Karta alerji bilgileri yazılmak istendiğinde ise benzer biçimde yazılacak kayıtları temsil eden `HastaVeriBean` nesnelerinin oluşturduğu koleksiyon `HastaSession` nesnesinin ilgili metoduna parametre olarak aktarılmakta; bu metot içerisinde, koleksiyondan alınan nesnelerin özelliklerini içeren APDU bayt dizinleri oluşturulmakta ve bu dizinler karta yazılmak üzere gönderilmektedirler.

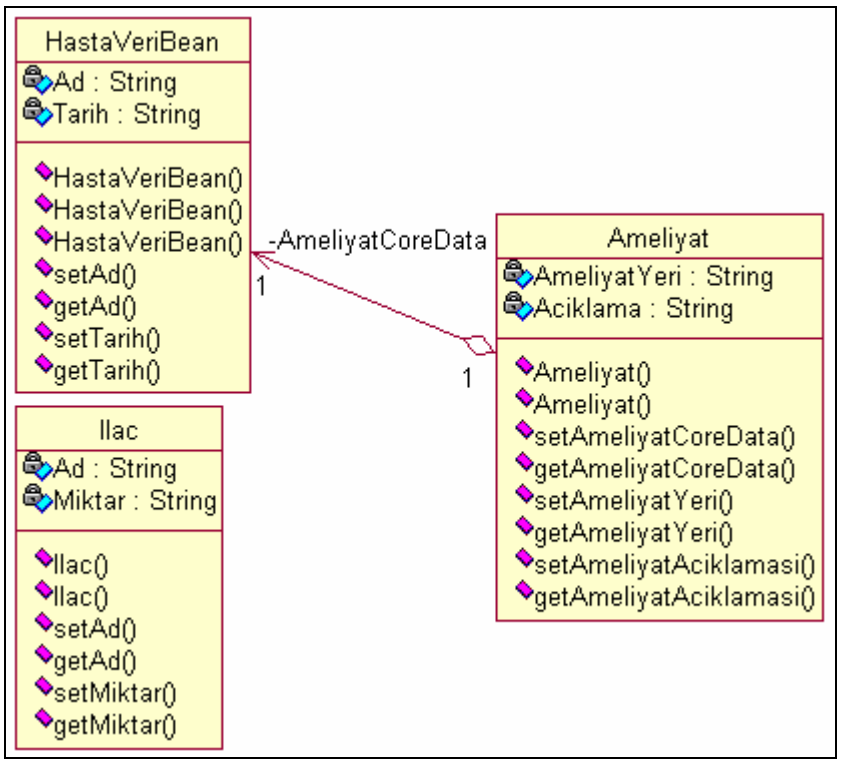
İstemci pakette alerji, aşı ve hastalık bilgilerinden başka ameliyatlara ait çekirdek (core) bilgiler de bu `HastaVeriBean` nesnelerinde yer almaktadır. Sürekli kullanılan ilaç kayıtlarının her biri ise `Ilac` nesneleri ile temsil edilmektedirler. Tüm bu bilgilerin karttan alınması ve yazılması

yukarıda anlatılan prosedürde gerçekleşmektedir. `HastaVeriBean`, `Ameliyat` ve `Ilac` sınıfları Şekil 4.16'da gösterilmiştir.

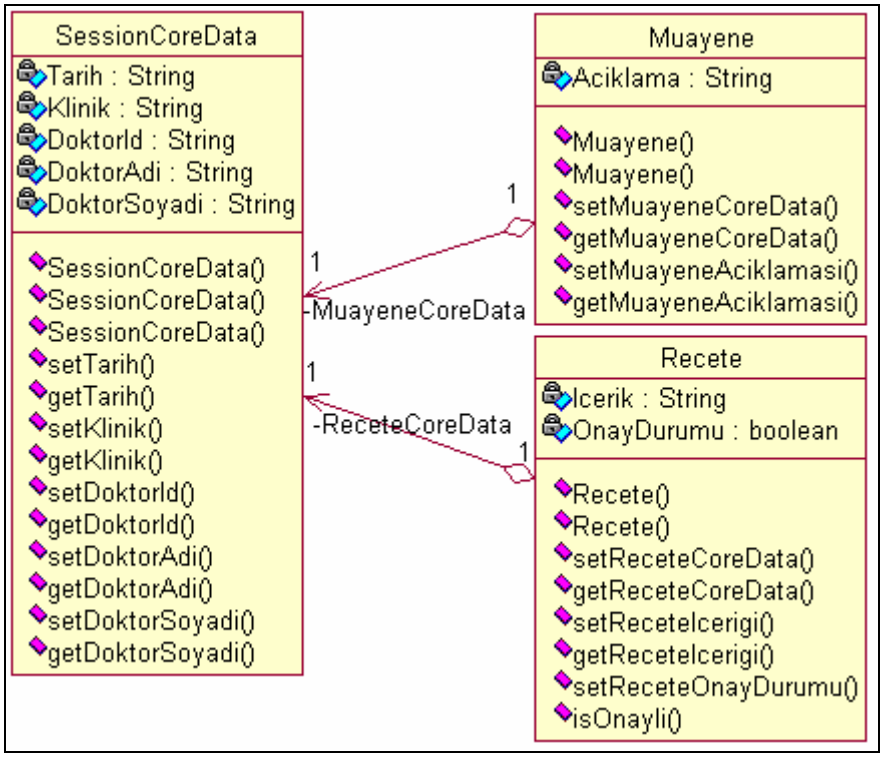


Şekil 4.15 HastaCoreData, AcilDurumKontaktBilgisi ve SigortaBilgisi sınıfları

Kartta yer alan hasta son muayene ve son reçetesine ait bilgiler pakette sırası ile `Muayene` ve `Recete` sınıflarından türetilen nesneler ile temsil edilirler (Şekil 4.17). `HastaSession` nesnesi istek durumunda son muayene ve son reçeteye ait çekirdek bilgileri –ki bunlar `SessionCoreData` nesneleri içerisinde barındırılır – ve muayene için açıklama, reçete için onay durumu ve içerik gibi özel bilgileri istemciye bu nesnelerin özelliği olarak gönderir. İstemci bileşen bu nesneler içerisinde barındırılan nesne özelliklerini istediği formatta görüntüleyebilir. Kullanıcıdan alınan muayene ve reçete bilgileri de yine bu sınıflardan türetilen nesnelere özellik olarak atanır. Türetilen bu nesneler de, özellikleri akıllı karta yazılmak üzere `HastaSession` nesnesinin ilgili metotlarına parametre olarak gönderilir.



Şekil 4.16 HastaVeriBean, Ameliyat ve Ilac sınıfları



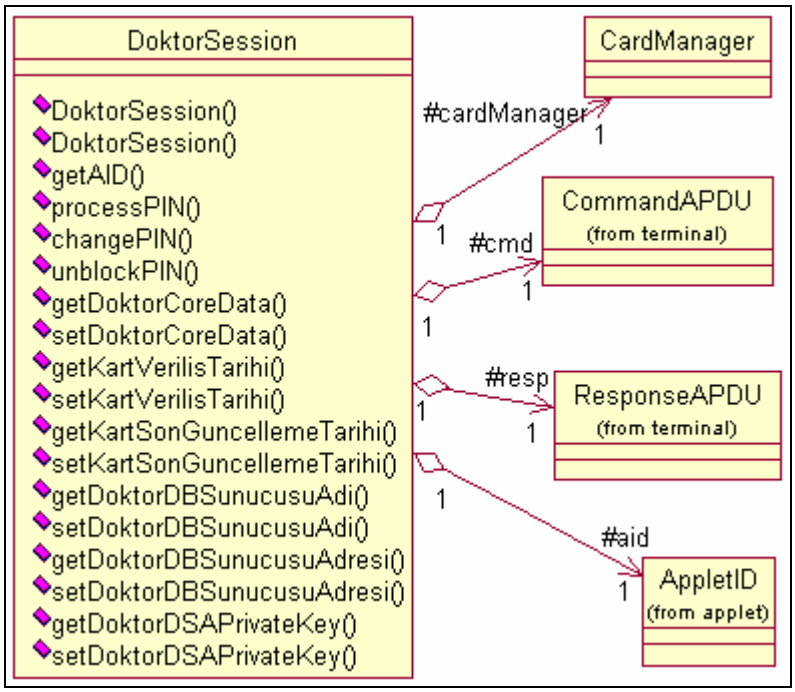
Şekil 4.17 SessionCoreData, Muayene ve Recete sınıfları

Yukarıda sözü edilen hasta bilgileri dışındaki diğer bilgilerin alışverişinde ise standart Java nesneleri görev alır. Örneğin hasta bilgilerinin tutulduğu merkez veritabanına ait bağlantı bilgisi `java.lang.String` nesnesi içerisinde transfer edilir. Yine hastaya ait dijital şifreleme anahtarı gerekli DES anahtarı oluşturulmak üzere istemci bileşene bayt dizisi olarak iletilir.

Özetle; istemci ara yazılımı paketinde yer alan `HastaSession` nesnesi kart ara yüz bileşenleri yerine hasta kartı ile APDU iletişiminde bulunmakta; karttan gelen bilgileri üst seviye nesne yapılarına gömerek bileşenlere göndermekte ve ara yüzden gelen bilgileri de bayt dizinlerine çevirerek bu dizinleri yazılmak üzere kart uygulamasına yollamaktadır. Özellikle on altılı kodlardan oluşan APDU paketlerinin oluşturulmasından ve Türkçe karakter kodlaması (encoding) gibi detaylı işlemlerden ara yüz bileşenlerinin soyutlanması sağlanmıştır.

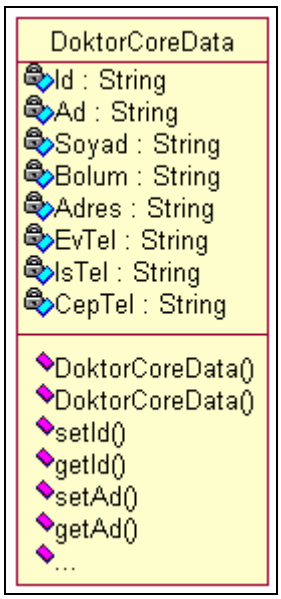
4.2.2.3 DoktorSession sınıfı

Sistemde yer alan `DoktorSession` sınıfından türetilen nesne de `HastaSession` nesnesi gibi akıllı kart iletişimlerinde görev almaktadır. Bir önceki bölümde detayları verilen `HastaSession` nesnesi ile tasarımı aynıdır. Tek fark `DoktorSession` nesnesi, doktor akıllı kartları ile ilgili iletişim işlemlerini yerine getirir. Bu nesnelerin türetildiği sınıf Şekil 4.18'de gösterilmiştir.



Şekil 4.18 DoktorSession sınıfı

Doktor kartında yer alan doktor genel bilgilerinin istemci ara yüz bileşenlerine iletiminde `DoktorSession` nesnesi istemcilere bu bilgileri `DoktorCoreData` nesneleri içerisinde göndermekte; karta yazılmak üzere istemcilerden gönderilen bilgileri de yine bu nesneler içerisinde kabul etmektedir. `DoktorCoreData` sınıfı Şekil 4.19'da gösterilmiştir.



Şekil 4.19 DoktorCoreData sınıfı

Doktor genel bilgileri dışında kalan ve kart üzerinde yer alan diğer bilgiler ise (Doktora ait dijital imza, iletişime geçilecek merkez veritabanı sunucusu adresi, vb.) Java standart nesneleri ile taşınmaktadır.

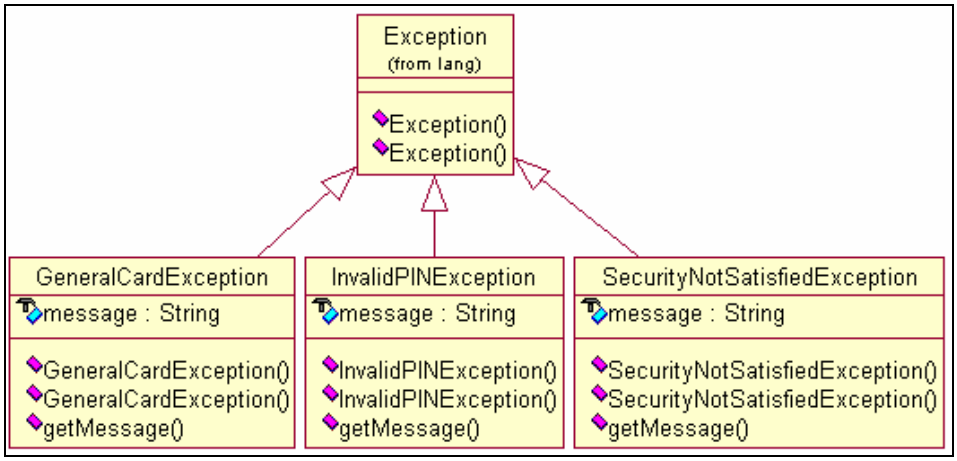
4.2.2.4 Hata kontrolü

Akıllı kart iletişiminde ortaya çıkan hataların kontrolünü sağlamak amacıyla Şekil 4.20'de gösterilen sınıflar hazırlanmıştır. Sınıflar `java.lang.Exception` sınıfının alt sınıfları olup bu sınıfının özelliklerini miras almışlardır (*inheritance* ilişkisi).

Hem `HastaSession` nesnesi hem de `DoktorSession` nesneleri karttan gönderilen hata mesajı kodlarının kontrolünden de diğer sistem bileşenlerini soyutlamışlardır. Örneğin hatalı şifre girişi gibi bir durumda, gelen hata kodu bu nesnelerde değerlendirilerek istemci bileşene `InvalidPINException` sınıfından türettiği bir nesneyi gönderir. Bu nesnenin mesaj özelliğinde kullanıcının kaç şifre deneme hakkı kaldığını

bildiren hata mesajı yer alır. Bu mesajı istemci bileşen dilediği formatta kullanıcıya aktaracaktır.

Kart istemcisi bileşenler herhangi bir PIN girişi gerçekleştirmeden kart bilgilerine erişmek isteyebilirler. Bu durumda kart içerisindeki yazılım bunu kontrol edip ilgili hata kodunu, işlemi istemci için gerçekleştiren `HastaSession` veya `DoktorSession` nesnesine gönderir. Gelen kodu işleyen bu nesneler de çağırımı yapan istemciye `SecurityNotSatisfiedException` nesnesini yanıt olarak döndürür. Güvenlik dışında oluşabilecek kart çalışma zamanı hataları ise `GeneralCardException` sınıfından türetilen nesneler aracılığı ile istemcilere bildirilir.



Şekil 4.20 GeneralCardException, InvalidPINException ve SecurityNotSatisfied sınıfları

5 SİSTEM SUNUCU BİLEŞENLERİ

Bu bölümünde Akıllı Kart Sağlık Sistemi'nde yer alan sunucu bileşenleri ve tasarlanan hastane merkez veritabanı hakkında bilgiler yer almaktadır. İlk bölümde sistem RMI protokolü ve sunucu katmanda görev alan yazılımın mimarisi ortaya konmuştur. İkinci bölümde ise sistem veritabanı, bu veritabanının tasarımı ve sistemde hangi nesneler ile temsil edildiği hakkında bilgiler verilmiştir.

5.1 Sistem Sunucu Katmanı Yazılımı

Bu bölümde; sistemde kart terminallerinin veritabanlarında bulunan bilgilere erişmesini ve gerekli güncellemeleri yapmalarını sağlayan sistem sunucularına ait, Java RMI teknolojisine dayalı yazılım modeli ve bu modelde yer alan bileşenler hakkında bilgi verilmiştir.

5.1.1 Sunucu katmanı nesne modeli

Daha önce de bahsedildiği gibi muayenehanelerde yer alan kart terminalleri ile söz konusu sunucular arasındaki iletişim Java RMI teknolojisine dayalı bir protokol üzerinden gerçekleştirilmektedir. Bu protokol istemcilerin sistem veritabanlarına güvenli ve yetkilendirilmiş olarak erişmelerini sağlamaktadır.

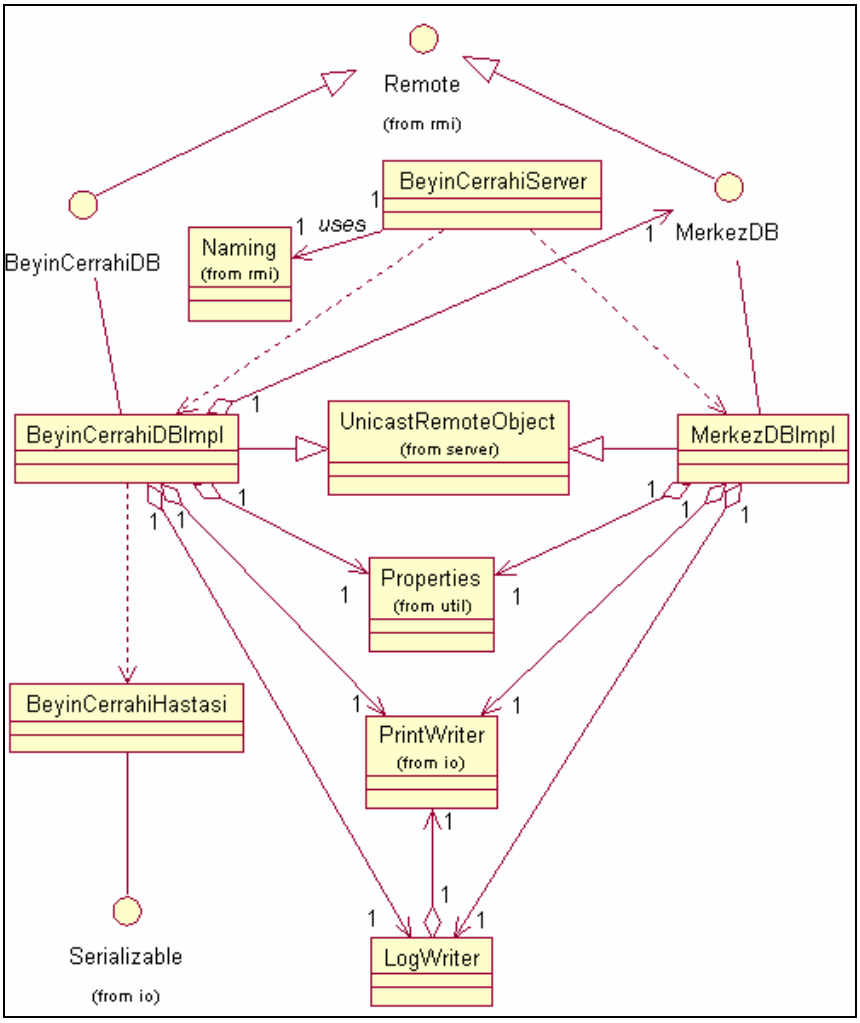
Sistemde her hastane bölümünde veya klinikte bir RMI sunucusunun olması düşünülmektedir. Bir hastane bölümünde yer alan kart terminalleri ilgili bölümde yer alan sistem sunucusu ile iletişimde bulunacaktır. Bu sunucular, istemcilerin hem kliniğe ait lokal veritabanına hem de hastane merkez veritabanına bağlanmalarını sağlamaktadır.

Sistem mimarisinin anlatıldığı bölümde de bahsedildiği gibi, RMI sunucularında yer alan sistem bileşenleri, MVC desenine dayalı olarak gerçekleştirilen sistem mimarisinde denetleyici görevini üstlenmektedirler. Sunucular üzerinde, ilgili işlemleri gerçekleştirecek olan uzak metot uyandırmalarına cevap veren nesneler yaşamaktadır. İstemci terminaller bu nesnelerin uzak ara yüzlerini kullanarak nesnelere erişmekte ve bu nesnelerden hizmet almaktadırlar. Terminaller sadece uzak nesnenin ilgili metodunu çağırmaktadır. Veritabanına erişim, ilgili sorguların gerçekleştirilmesi ve yetkilendirme kontrollerinin hepsi bu uzak nesneler tarafından gerçekleştirilir.

Her RMI sunucu üzerinde istemcilerin hastane merkez veritabanına erişmelerini sağlayacak, uzak metot çağrımlarına hazır bir nesnenin yaşaması tasarlanmıştır. Bunun yanı sıra sunucunun yer aldığı bölüme özel veritabanına bağlantıları ise ayrı bir uzak nesne gerçekleştirmektedir.

Tez kapsamında, ortak çalışmada bulunulan Ege Üniversitesi Tıp Fakültesi Beyin Cerrahi Ana Bilim Dalı için örnek bir RMI sunucu yazılımı hazırlanmıştır. Yazılıma ait nesne modeli Şekil 5.1'de gösterilmiştir.

Terminallerin kullandığı uzak nesnelere ait ara yüzler `MerkezDB` ve `BeyinCerrahiDB`'dir. Bu ara yüzleri `MerkezDBImpl` ve `BeyinCerrahiDBImpl` sınıflarından türetilen nesneler uygulamaktadırlar (*implement* ilişkisi). `MerkezDBImpl` ve `BeyinCerrahiDBImpl` sınıflarından türetilen birer nesne `BeyinCerrahiServer` nesnesi tarafından sunucu üzerinde yer alan RMI kayıtçısına (*rmiregistry*) bağlanırlar. Bağlanan bu iki nesne bundan sonra uzak ara yüzleri aracılığı ile istemci terminaller tarafından kullanılmaya ve istekleri karşılamaya hazırdırlar. Aşağıdaki alt başlıklarda, hazırlanan bu uzak ara yüzler ve bu ara yüzleri kullanan nesneler hakkında detaylı bilgi yer almaktadır. Tasarımın dayalı olduğu Java RMI teknolojisi ile ilgili bilgiler ise tez alt yapı çalışmaları bölümünde verilmiştir.

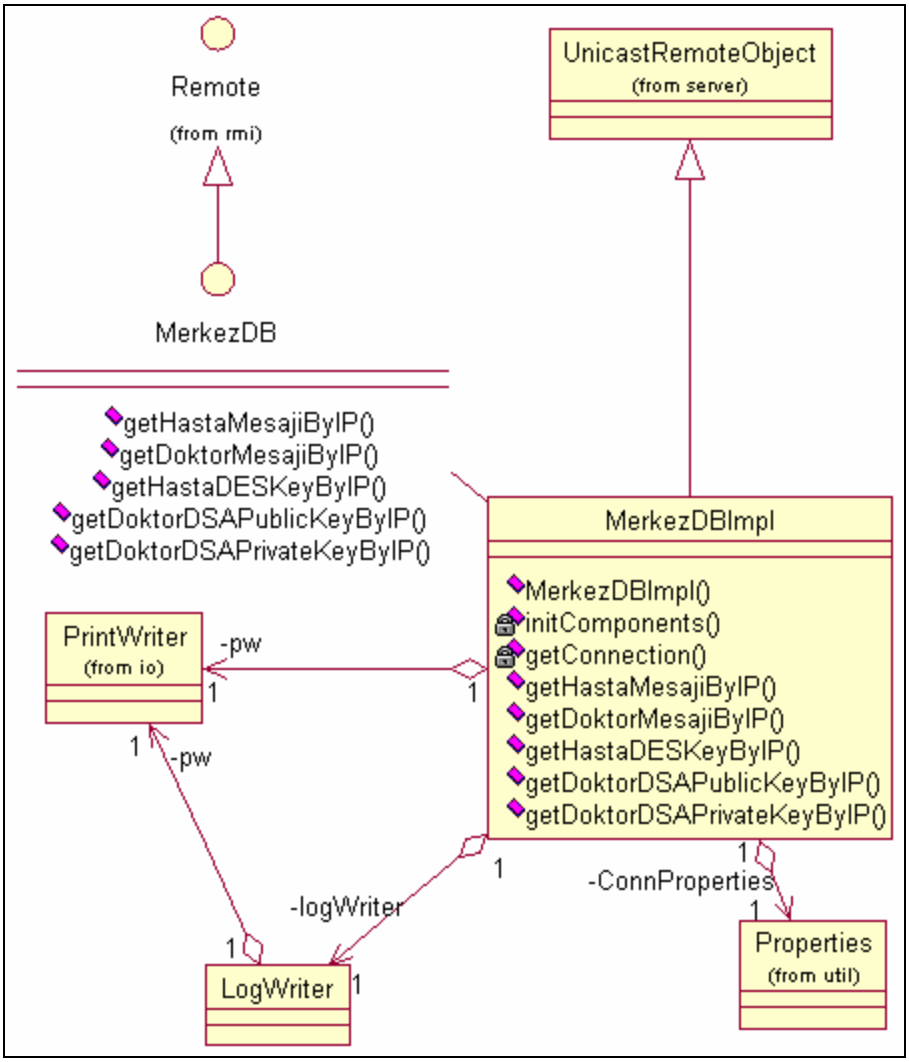


Şekil 5.1 Sistem RMI bileşenlerine ait nesne modeli

5.1.1.1 MerkezDB ara yüzü ve MerkezDBImpl nesnesi

`MerkezDB` ara yüzüne sahip uzak nesneleri hem kart terminalleri hem de RMI sunucuları üzerinde çalışan diğer uzak nesneler kullanır. Kart terminalleri, üzerinde işlem gördükleri akıllı kartların sahiplerine ait mesajların bu kart sahiplerinin kayıtlarının tutulduğu merkez veritabanından alınmasını ve kullanıcıya iletimini `MerkezDB` ara yüzlü

uzak nesnenin ilgili metodunu çağırarak gerçekleştirirler. Diğer uzak nesneler de sistem kullanıcılarına ait şifreleme ve imzalama anahtarlarını –ki bunlar da yine kullanıcının kayıtlarının tutulduğu merkez veritabanında saklanır – işlemlerde kullanmak için yine bu ara yüze sahip nesneleri kullanmaktadır. Şekil 5.2'de `MerkezDB` ara yüzü ve bunu uygulayan `MerkezDBImpl` sınıfı gösterilmiştir.



Şekil 5.2 MerkezDB ara yüzü ve bunu uygulayan MerkezDBImpl sınıfı

`java.rmi.Remote` ara yüzünün bir uzantısı olarak hazırlanan `MerkezDB` ara yüzünü sistemde `MerkezDBImpl` sınıfı uygulamaktadır. Bu sınıf aynı zamanda `java.rmi.server.UnicastRemoteObject` sınıfının bir alt sınıfı olarak tasarlanmıştır. Böylece `MerkezDBImpl` sınıfından türetilen nesneler birer uzak nesne olarak görev yapmakta ve metot çağırmalara cevap vermektedir. `MerkezDBImpl` sınıfından türetilen nesnelerin `LogWriter` ve `java.io.PrintWriter` türündeki özellikleri, sunucu tarafı işlemlerin kayıtlarının tutulmasında görev alır. `MerkezDBImpl`, veritabanına bağlantı ve sürücü bilgilerini ise ilgili sistem dosyasından alıp `java.util.Properties` türündeki nesne içerisinde saklamakta ve işlemlerde kullanmaktadır.

Şekil 5.2'de `MerkezDB` ara yüzünün içerdiği ve uygulayıcıları tarafından içlerinin doldurulması gereken metotlar görülmektedir. Bu metotları sistemde uygulayan `MerkezDBImpl`, istemcilerin merkez veritabanından hastalara ve doktorlara ait mesajları ve şifreleme ve imzalama anahtarlarını almasını sağlar. Tüm metotlara parametre olarak ilgili kullanıcıya ait sistem geneli ID ve bu kullanıcıya ait kayıtların tutulduğu merkez veritabanına bağlantı adresi istemciler tarafından geçirilmektedir. RMI kanalı üzerinden aktarılan parametreleri alan `MerkezDBImpl` nesnesi tüm veritabanı bağlantılarını ve kontrolleri sağlayarak terminal tarafından istenen bilgileri aynı kanal üzerinden gönderir.

`MerkezDBImpl` sınıfının RMI derleyicisine parametre olarak verilmesi ile oluşturulan ve *"stub"* görevini görecek olan sınıf `MerkezDBImpl_Stub` adını almaktadır. Bu sınıf hem istemci bilgisayarlarda hem de RMI sunucusu üzerinde yer alır.

Beyin Cerrahi Bölümü için hazırlanan RMI sunucu yazılımında `BeyinCerrahiServer` adlı uygulama uzak metot çağırımlarında görev alacak `MerkezDBImpl` nesnesini oluşturmakta ve *"rmiregistry"*e kaydetmektedir. İstemci terminal uygulaması öncelikle RMI sunucusuna bağlanarak *"rmiregistry"* üzerinde kayıtçıya bağlı bulunan `MerkezDBImpl` nesnesini belirler. Eğer nesne kayıtçıya bağlanmışsa bu

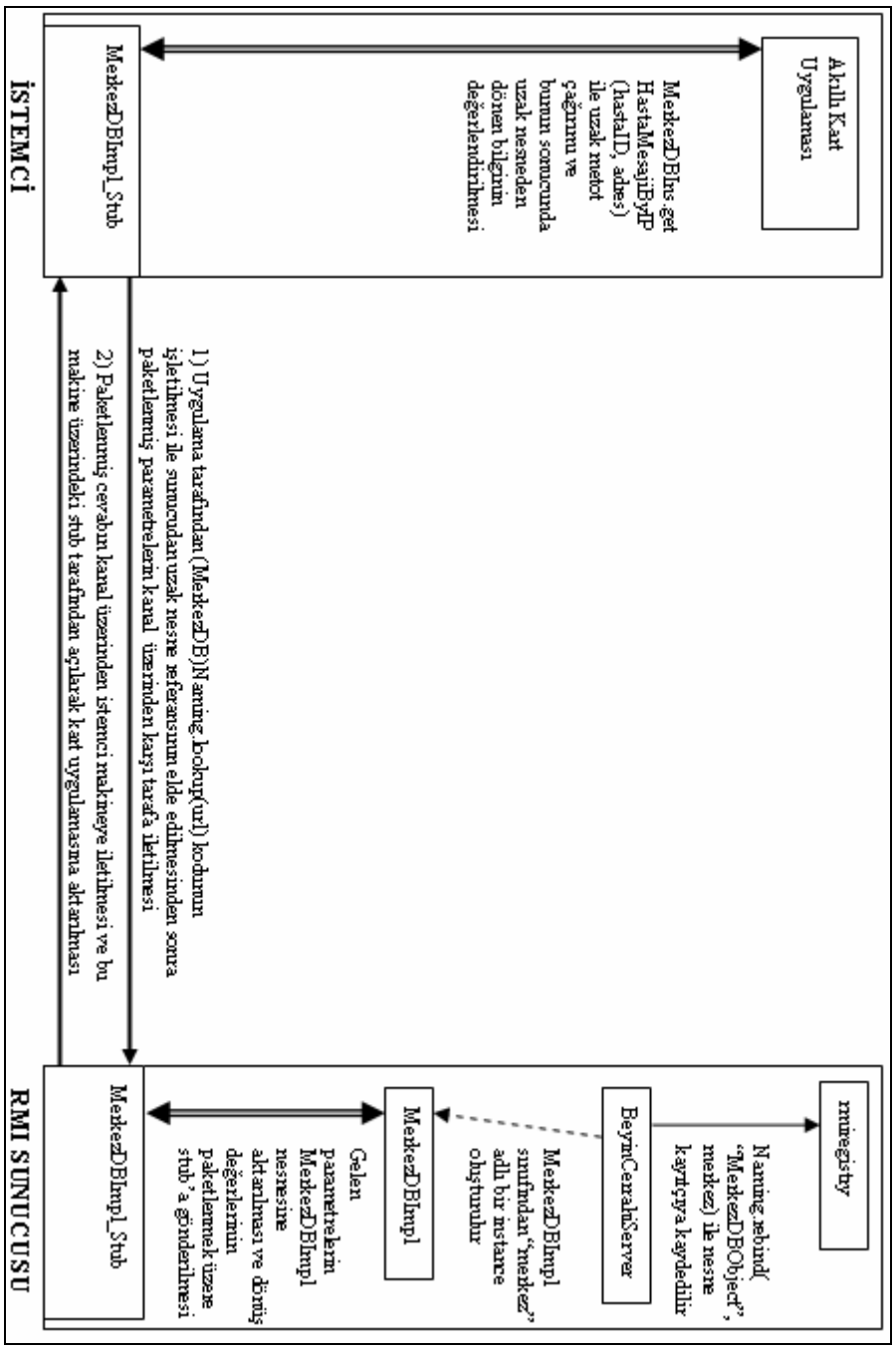
nesneye `MerkezDB` ara yüzü aracılığı ile erişerek istediği işlemleri yerine getirir. Örneğin istemci uygulama, üzerinde oturumu açılan hastaya ait merkez veritabanı mesajını almak istediğinde `MerkezDBImpl` uzak nesnesinin `getHastaMesajiByIP()` metodunu hastanın ID'si ve veritabanı bağlantı adresi parametreleri ile çağırır. Verilen parametreler istemci makine üzerine yer alan `MerkezDBImpl_Stub` tarafından RMI kanalında gönderilmek üzere paketlenir (parametre kodlama) ve RMI sunucusuna gönderilir. Karşı taraftaki *"stub"* kod da gelen bu parametreleri açar ve `MerkezDBImpl` nesnesine iletir. Nesne gerekli veritabanı bağlantılarını kurar ve elde ettiği mesajı (veya bir hata ile karşılaşıldığında bu hata bilgisini) geri gönderir. Dönüş değeri olan mesaj (veya hata bilgisi) yine *"stub"* tarafından paketlenir ve istemci makineye iletilir. Benzer şekilde bu makinede yer alan *"stub"* parametreyi açar ve istemci uygulamaya iletir. (Şekil 5.3)

`MerkezDBImpl` nesnesi kendisine gelen her isteği, geliş zamanları ve istemci adres bilgisi ile beraber kaydetmektedir. Bunun yanı sıra yerine getirdiği veritabanı işlemlerini (hangi veritabanına bağlanıldığı, alınan bilgi, vb.) ve diğer kontrol işlemlerini de yine işlem zamanları ile beraber kaydetmektedir.

5.1.1.2 BeyinCerrahiDB ara yüzü ve BeyinCerrahiDBImpl nesnesi

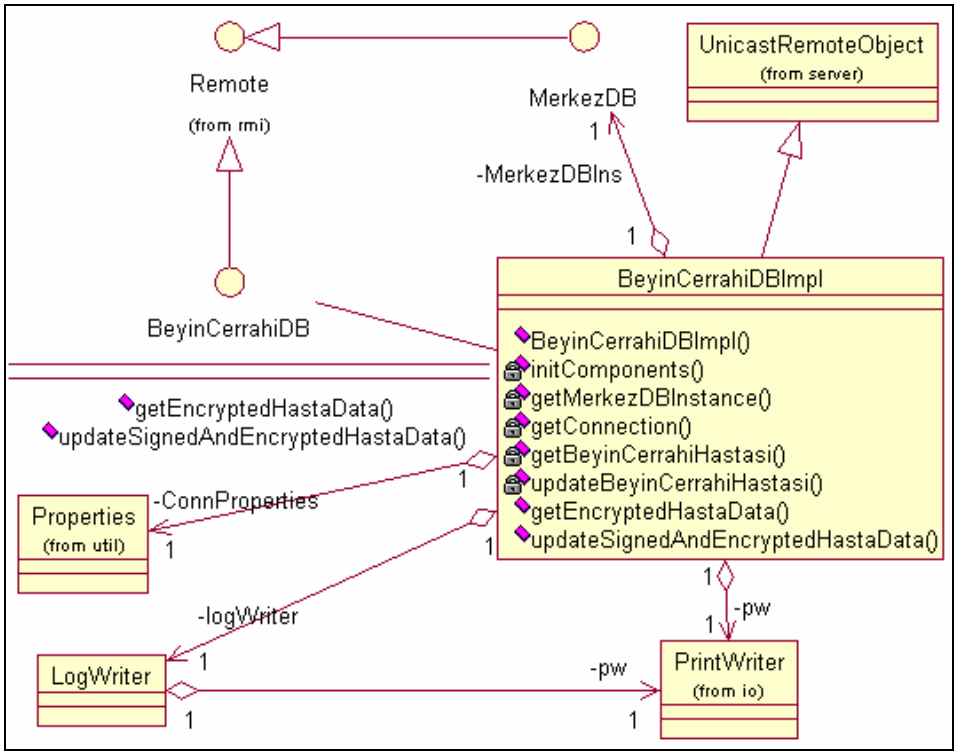
Daha önce de belirtildiği gibi sistemde tasarlanan; her hastane bölümünün kendine ait veritabanının olması ve hastaların bu bölümlere ait kayıtlarının bu lokal sistemde tutulmasıdır. Bu amaçla her bölümde yer alan RMI sunucusu bu veritabanı üzerinde işlem gerçekleştirecek uzak nesneleri barındıracaktır.

Bu bölüme kadar anlatılan sunucu yazılımı bileşenleri her bir RMI sunucusu için ortak ihtiyaçları karşılamaktadır. Bu bölümde ise hastane bölümlerine özel bileşenlerin tasarımı yer almaktadır. Sistemin uygulamasında örnek olarak Ege Üniversitesi Beyin Cerrahi Bölümü'nde çalışacak sunucu bileşenleri hazırlanmıştır.



Şekil 5.3 MerkezDBImpl uzak nesnesinin hazırlanması ve istemciler tarafından kullanılması

Beyin Cerrahi bölümünde yer alan kart terminallerinde hasta muayeneleri gerçekleştirilirken doktorlar, hastanın bu bölüme özel bilgilerine ulaşmak ve gerekli durumlarda güncellemek istemektedirler. Bu istekleri güvenli ve yetkilendirilmiş olarak yerine getiren uzak nesne Şekil 5.4'de gösterilen `BeyinCerrahiDBImpl` sınıfından türemektedir. Bu sınıf `BeyinCerrahiDB` ara yüzünde tanımlamaları verilen metotları uygular.



Şekil 5.4 BeyinCerrahiDB ara yüzü ve bunu uygulayan BeyinCerrahiDBImpl sınıfı

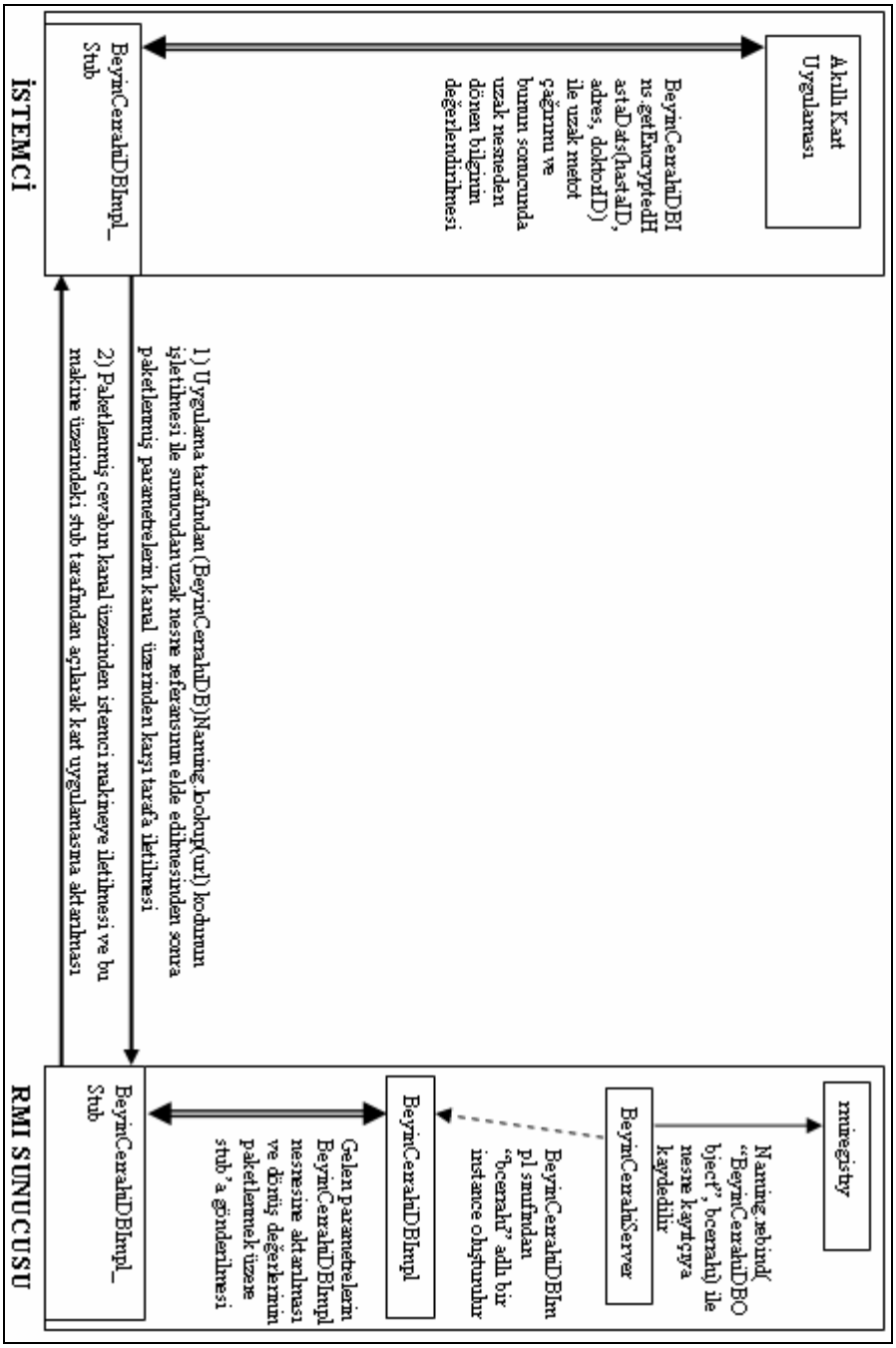
`BeyinCerrahiDB` ara yüzü, `MerkezDB` ara yüzü gibi `java.rmi.Remote` ara yüzünün bir uzantısıdır ve kendisini uygulayacak olan uzak nesneler için metot tanımlarını içerir. Bu ara yüzü uygulayan `BeyinCerrahiDBImpl` ise aynı zamanda `java.server.UnicastRemoteObject` sınıfının bir alt sınıfıdır. Bu

yapısı ile `BeyinCerrahiDBImpl`, uzak metot çağırımlarına cevap vermektedir. `MerkezDBImpl` ile benzer şekilde, gerçekleştirilen işlemlere ait kayıtların ilgili sistem dosyasında tutulmasında `LogWriter` nesnesi ve `java.io.PrintWriter` nesnesi görev almaktadır. Nesne, veritabanı bağlantısı ve sürücü bilgilerini de yine özel bir dosyadan okumakta ve `java.util.Properties` nesnesi içerisinde saklamaktadır.

`BeyinCerrahiDBImpl` nesnesi kart terminalleri için uzak nesne görevi görmesinin yanı sıra kendisi de bir RMI istemcisidir. Merkez veritabanlarına ulaşmak istediğinde bu nesne de `MerkezDB` ara yüzünü kullanarak aynı sunucuda (ya da başka bir RMI sunucusunda) görev alan `MerkezDBImpl` uzak nesnesinin ilgili metodunu çağırmaktadır.

`BeyinCerrahiDBImpl` sınıfının RMI derleyicisinde derlenmesi sonucu oluşan *"stub"* sınıfı `BeyinCerrahiDBImpl_Stub`'dır. Bu *"stub"* kodu hem sunucuda hem de istemcilerde yer almaktadır.

Şekil 5.5'te de görüldüğü gibi, yine `BeyinCerrahiServer` uygulaması `BeyinCerrahiDBImpl` nesnesinden oluşturarak RMI kayıtçısına bağlar. İstemciler de sunucuya bağlanarak hazır olan bu nesneye ait referansı elde ederler. Bu aşamadan sonra istemciler kendi metotlarıymış gibi bu nesnenin metotlarını kullanarak işlem görürler. Örneğin Beyin Cerrahi veritabanından hastaya ait bilgileri şifreli olarak elde etmek isteyen terminal gerekli parametrelerle bu uzak nesnenin metodunu çağırır. Parametreler istemci taraftaki *"stub"* tarafından paketlenir ve kanaldan gönderilir. Karşı taraftaki *"stub"* da gelen parametreleri açarak `BeyinCerrahiDBImpl` nesnesine aktarır. Nesne, veritabanı bağlantılarını gerçekleştirip gereken bilgiyi hazırlar ve sunucudaki *"stub"*a geri yollar. Benzer şekilde *"stub"* bilgileri paketler ve aynı yoldan istemciye gönderir. İstemci makinede yer alan *"stub"* kod da gelen bilgileri açarak uzak metodu çağıran uygulamaya iletir.



Şekil 5.5 BeyinCerrahiDBImpl uzak nesnesinin hazırlanması ve istemciler tarafından kullanılması

`MerkezDBImpl` nesnesine benzer şekilde `BeyinCerrahiDBImpl` de kendisine gelen isteklerin ve bu istekleri karşılarken kendisinin yerine getirdiği işlemlerin zamanları ile beraber kaydını tutmaktadır.

`BeyinCerrahiDBImpl` nesnesinin `BeyinCerrahiDB` ara yüzünün metotlarını nasıl uyguladığı aşağıdaki yer almaktadır.

getEncryptedHastaData() metodunun uygulanması:

İstemciler `BeyinCerrahiDBImpl`'e ait `getEncrypted HastaData()` metodunu çağırırken bilgilerini istedikleri hastaya ait ID'yi, bu hastanın kayıtlarının tutulduğu merkez veritabanının adresini ve istemci doktorun ID'sini parametre olarak gönderirler. `BeyinCerrahiDBImpl` nesnesi hastaya ait bilgileri kliniğe ait veritabanından alır ve bu bilgileri Şekil 5.6'da gösterilen `BeyinCerrahiHastasi` sınıfından türettiği bir nesneye özellik olarak atar.

Bilgileri şifrelemek için gereken, hastaya ait DES anahtarı `MerkezDB` uzak nesnesinin ilgili metodu çağrılarak elde edilir. İletişime geçilecek merkez veritabanının adresi yukarıda da belirtildiği gibi parametre olarak gelmektedir. Bayt akışına (Byte stream) dönüştürülen `BeyinCerrahiHastasi` nesnesinin içeriği bu anahtar ile şifrelenir ve elde edilen şifreli bayt dizisi istemciye döndürülür. Bahsedilen tüm bu işlemler sırasında bir hata ile karşılaşıldığında (örneğin hasta bilgilerinin veritabanından elde edilememesi, veri şifreleme hatası, vb.) hata `java.rmi.RemoteException` nesnesi içerisinde istemci terminale bildirilir.

updateSignedAndEncryptedHastaData() metodunun uygulanması:

Hastaya ait beyin cerrahi bilgilerinin veritabanında güncellenmesi gerektiğinde istemci uygulamada bu bilgiler gönderilmeden önce

şifrelenir ve imzalanır. Daha sonra yine `BeyinCerrahiDBImpl`'e ait `updateSignedAndEncryptedHastaData()` metodu çağrılır. Metoda, şifreli ve imzalı bilgiler dışında, bilgilerin sahibi olan hastanın ID'si ve bu hastaya ait kayıtların tutulduğu merkez veritabanı adresi, güncellemeyi gerçekleştirmek isteyen doktorun ID'si ve bu doktora ait kayıtların tutulduğu merkez veritabanı adresi bilgileri parametre olarak geçirilir.

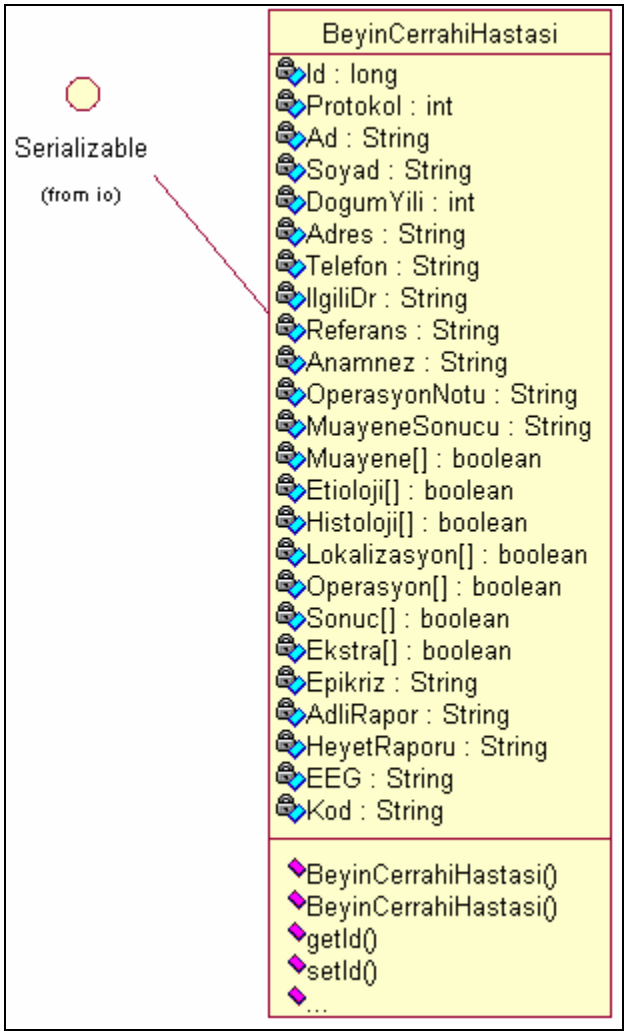
Metot içerisinde ilk olarak `MerkezDB` uzak nesnesi kullanılarak doktora ait DSA genel anahtarı merkez veritabanından elde edilir. Doktorun DSA özel anahtarı ile imzalanmış verilerin bu anahtar ile doğrulaması gerçekleştirilir. Bu işlemden sonra hastaya ait DES anahtarı yine aynı uzak nesnenin ilgili metodu çağrılarak elde edilir ve onaylanmış şifreli veriler açılır. Veriler açıldığında elde edilen `BeyinCerrahiHastasi` nesnesi bu hastaya ait güncellenmiş beyin cerrahi bilgilerini içermektedir. Bu bilgiler lokal veritabanında güncellenerek işlem sonucu istemci terminale döndürülür. İşlemler sırasında bir hata ile karşılaşılırsa güncelleme gerçekleştirilmez ve hata yine `java.rmi.RemoteException` nesnesi aracılığı ile istemciye bildirilir.

5.1.1.3 BeyinCerrahiHastasi nesnesi

Hastalara ait beyin cerrahi bilgileri veritabanından alındığında veya veritabanına güncellenmek istendiğinde alınan veya güncellenmek istenen bilgiler `BeyinCerrahiHastasi` nesnesi içerisinde saklanmaktadır. Bu nesnelerin türetildiği sınıf Şekil 5.6'da gösterilmiştir. Nesne özellikleri, halihazırda Ege Üniversitesi Beyin Cerrahi Bölümü'nde kullanılmakta olan veritabanında her bir hastanın kaydında yer alan alanlardır.

Dikkat edilirse `BeyinCerrahiHastasi` sınıfı `java.io.Serializable` ara yüzünü uygular. Böylelikle nesne ağ üzerinde bayt akışı halinde transfer edilebilmektedir. Örneğin istemci uygulama, kullanıcı ara yüzünden aldığı hastaya ait beyin cerrahi

bilgilerini bu nesne içerisinde saklamaktadır. Daha sonra bu nesne, bayt akışına dönüştürülmüş ve şifrelenmiş olarak RMI kanalı üzerinden sunucu makineye iletilmektedir. Sunucu tarafında da gelen akışdan `BeyinCerrahiHastasi` nesnesi yeniden elde edilmekte ve işlemlerde kullanılmaktadır.



Şekil 5.6 BeyinCerrahiHastasi sınıfı

5.1.2 Sunucu yazılımının yerleştirilmesi ve çalıştırılması

Sistemin uygulamasında RMI sunucusu olarak Intel P4 1.4 GHz işlemci ve 256 Mb SDRAM'e sahip, üzerinde Microsoft Windows XP işletim sistemi ve JRE 1.4.0 bulunan bir PC kullanılmıştır. Önceki bölümde haklarında detaylı bilgi verilen yazılım bileşenlerinin tamamı bu makine üzerinde yer almıştır. Bağlantıya geçecek istemcilerde ise kullanılacak uzak nesnelere ait ara yüzler (`MerkezDB` ve `BeyinCerrahiDB`), ilgili *"stub"* kodları (`MerkezDBImpl_Stub` ve `BeyinCerrashiDBImpl_Stub`) ve iletişimde bulunurken kullanılan, serileştirilebilir özellikteki nesneye (`BeyinCerrahiHastasi`) ait sınıflar yer almıştır.

5.1.2.1 RMI kayıtçısının çalıştırılması ve uzak nesnelerin hazırlanması

Sunucu üzerinde RMI kayıtçısının çalıştırılması JRE bünyesinde yer alan *"rmiregistry"* komutunun işletilmesi ile gerçekleştirilir (Ek 3.1). Komut çalıştırılırken herhangi bir bağlantı noktası numarası verilmediğinden sunucu RMI isteklerini 1099 nolu varsayılan (default) bağlantı noktasından alacaktır.

Kayıtçı çalıştırıldıktan sonra istemcilere hizmet verecek uzak nesneleri ortamda oluşturacak ve bunları kayıtçıya bağlayacak `BeyinCerrahiServer` uygulaması çalıştırılır (Ek 3.2).

Bu aşamadan sonra iki adet nesne uzak metot çağırımlarına hazırdır.

5.1.2.2 İstemcilerin uzak nesneleri belirlemesi

RMI sunucusunda yer alan uzak nesneleri kullanmak isteyen istemcilerin öncelikle bu nesnelerin hazır olup olmadıklarını belirlemeleri ve referanslarını elde etmeleri gerekir. Bunun için istemci makine hizmet almadan önce sunucu üzerindeki kayıtçı ile iletişime geçerek referansları elde eder. Ek 3.3'te 155.223.41.231 IP adresine sahip bir RMI istemcisinin sunucuya bağlantısı sonucunda elde edilen ekran görüntüsü yer almaktadır.

5.1.2.3 MerkezDBImpl uzak nesnesinin istemciler tarafından kullanılması

Terminal üzerinde yer alan RMI istemci uygulaması, kart terminallerinde oturumu açılan sistem kullanıcılarına ait mesajların alınmasında `MerkezDBImpl` uzak nesnesini kullanır. Ek 3.4'te yine 155.223.41.231 IP adresine sahip kart terminalinden gelen mesaj alımı isteği görülmektedir.

`MerkezDBImpl` nesnesi isteği karşılamak üzere yerine getirdiği işlemleri sunucu üzerindeki ilgili kayıt dosyasına zamanları ile beraber kaydeder (Ek 3.5).

5.1.2.4 BeyinCerrahiDBImpl uzak nesnesinin istemciler tarafından kullanılması

Beyin Cerrahi bölümüne özel istekleri karşılayan `BeyinCerrahiDBImpl` nesnesi kart terminallerinde oturumu açılan hastaya ait beyin cerrahi bilgilerinin ağ üzerinden güvenli olarak alınmasında ve yine güvenli ve yetkilendirilmiş olarak güncellenmesinde kullanılmaktadır.

Ek 3.6'da görüldüğü gibi 155.223.41.231 IP adresine sahip kart terminali şifreli hasta bilgilerini almak istemektedir. Dikkat edilirse RMI sunucu makinesinin kendisi de merkez veritabanı erişimi için RMI istemcisi konumuna geçmektedir. Çünkü `BeyinCerrahiDBImpl` nesnesi, merkez veritabanı erişimi için `MerkezDBImpl` uzak nesnesini kullanır.

İsteği karşılamada görev alan iki uzak nesne de yerine getirdikleri işlemleri kaydetmektedirler (Ek 3.7 ve Ek 3.8).

Terminallerden gelen bilgi güncelleme isteği de yine `BeyinCerrahiDBImpl` uzak nesnesi tarafından yerine getirilir. Gelen isteğin ekran görüntüsü Ek 3.9'da yer almaktadır. Ek 3.10 ve Ek 3.11'de ise uzak nesnelerin bu isteği yerine getirirken gerçekleştirdikleri işlemlere ait kayıtlar gösterilmiştir.

5.2 Sistem Veritabanı

Akıllı Kart Sağlık Sistemi'nde, sistem geneli hasta ve doktor bilgileri merkezi bir veritabanında saklanmaktadır. Hazırlanan bu veritabanının tasarımı, tablo ilişkileri ve sistem yazılımlarında bu veritabanının nasıl nesneye dayalı olarak temsil edilip kullanıldığı aşağıdaki alt bölümlerde açıklanmıştır.

5.2.1 Veritabanının tasarımı

Sistem veritabanı, kayıtlı olan hastalara ve doktorlara ait sistem geneli tüm bilgileri içermektedir. Sistem geneli bilgi ile kastedilen tüm hastane bölümleri için ortak olan bilgilerdir. Şekil 5.7'de bu veritabanında yer alan tablolar ve ilişkileri yer almaktadır. Tablo alanları olarak şimdilik sadece anahtarlar gösterilmiştir.

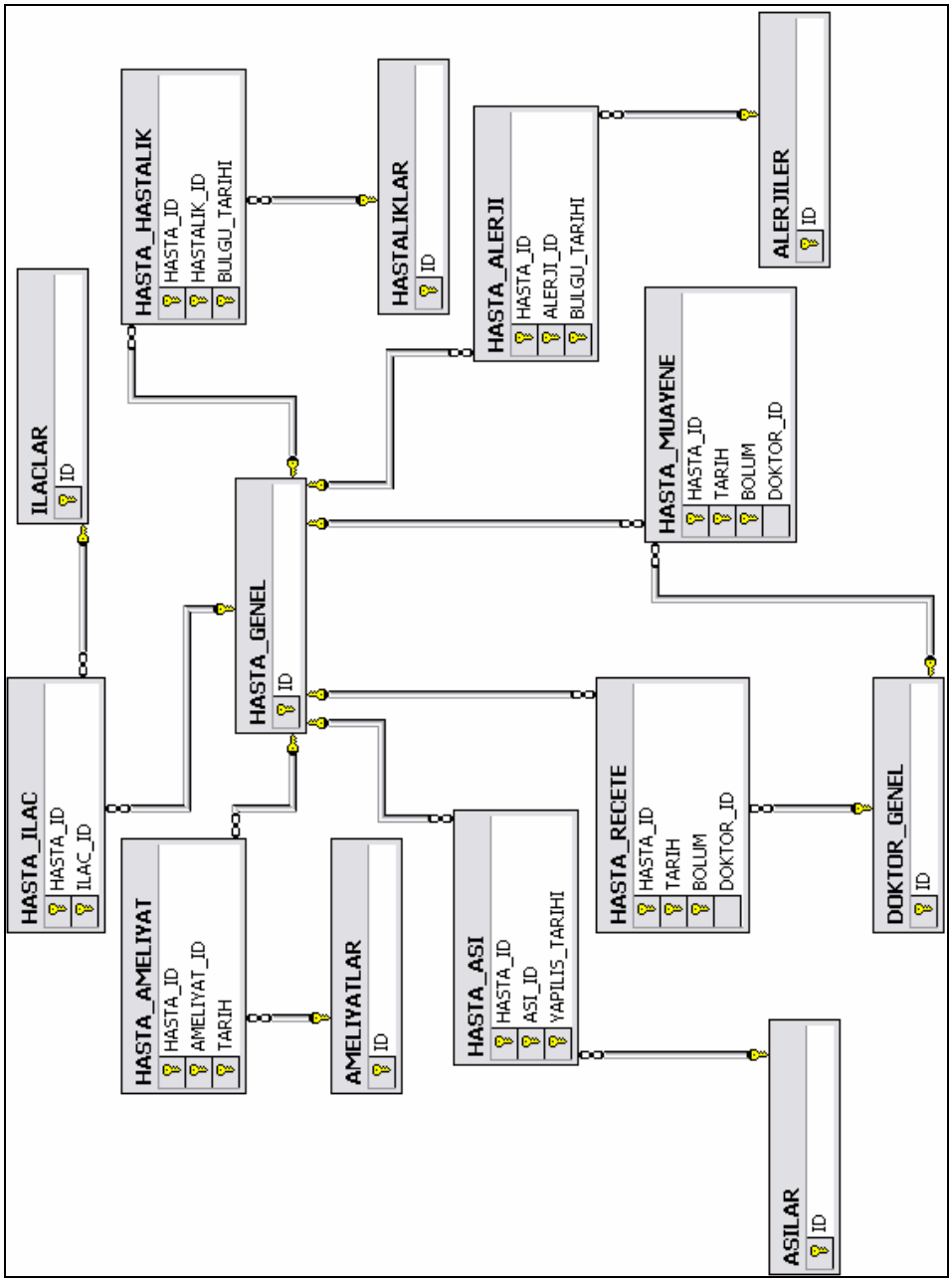
Hastalara ve doktorlara ait genel bilgiler (ad, soyad, adres, vb.) sırası ile HASTA_GENEL ve DOKTOR_GENEL adı verilen tablolarda yer alır. Tablolarda birincil anahtar (primary key) bu sistem kullanıcılarının sistem geneli tanımlama numaralarıdır.

Sistemde alerji, aşı, hastalık, vb. gibi sağlık bilgilerinin kodlanması ve ayrı tablolarda tutulması tasarlanmıştır. Sistem yönetimi tarafından sağlık bilgilerine tanımlama numarası verilir ve sağlık bilgisi ilgili tabloda yerini alır. Alerji, ameliyat, aşı, hastalık ve ilaçlar sırası ile ALERJILER, AMELIYATLAR, ASILAR, HASTALIKLAR ve ILACLAR tablolarında saklanmaktadır.

Herhangi bir hasta ile bir sağlık bilgisinin ilişkisi ayrı kayıt (enrollment) tablosunda yer alır. Örneğin tüm hastalara ait alerji bulguları HASTA_ALERJI tablosunda yer alır. Burada yer alan kayıtların birincil anahtarını ise hasta ve alerjiye ait sistem geneli id ile beraber bulgu tarihi oluşturur. Benzer şekilde hastalara ait ameliyat, aşı, hastalık ve sürekli kullandıkları ilaç kayıtları sırası ile HASTA_AMELIYAT, HASTA_ASI, HASTA_HASTALIK ve HASTA_ILAC tablolarında yer alır.

Daha önce de belirtildiği gibi hastalar muayene olduktan sonra muayene ve varsa reçetelerine ait bilgilerin doktor tarafından hasta akıllı kartlarına yazılması ve hastanın, kartı ile sistem yönetim birimine başvurarak bu bilgilere onay alması tasarlanmıştır. Aynı zamanda bu bilgiler sistem veritabanında da tutulacaktır. HASTA_MUAYENE ve HASTA_RECETE tablolarında hastalara ait bu muayene ve reçete kayıtları yer alır.

Tasarlanan bu veritabanı sistemde, MS SQL Server 2000 veritabanı yazılımı üzerinde hayata geçirilmiştir. SQL Server'in bu sürümü özellikle saf (pure) JDBC sürücüsü sağladığı için seçilmiştir. Böylelikle istemciler tarafından veritabanına erişim JDBC-ODBC köprüsüne oranla daha hızlı gerçekleşmektedir.



Şekil 5.7 Sistem veritabanı tabloları ve tablo ilişkileri

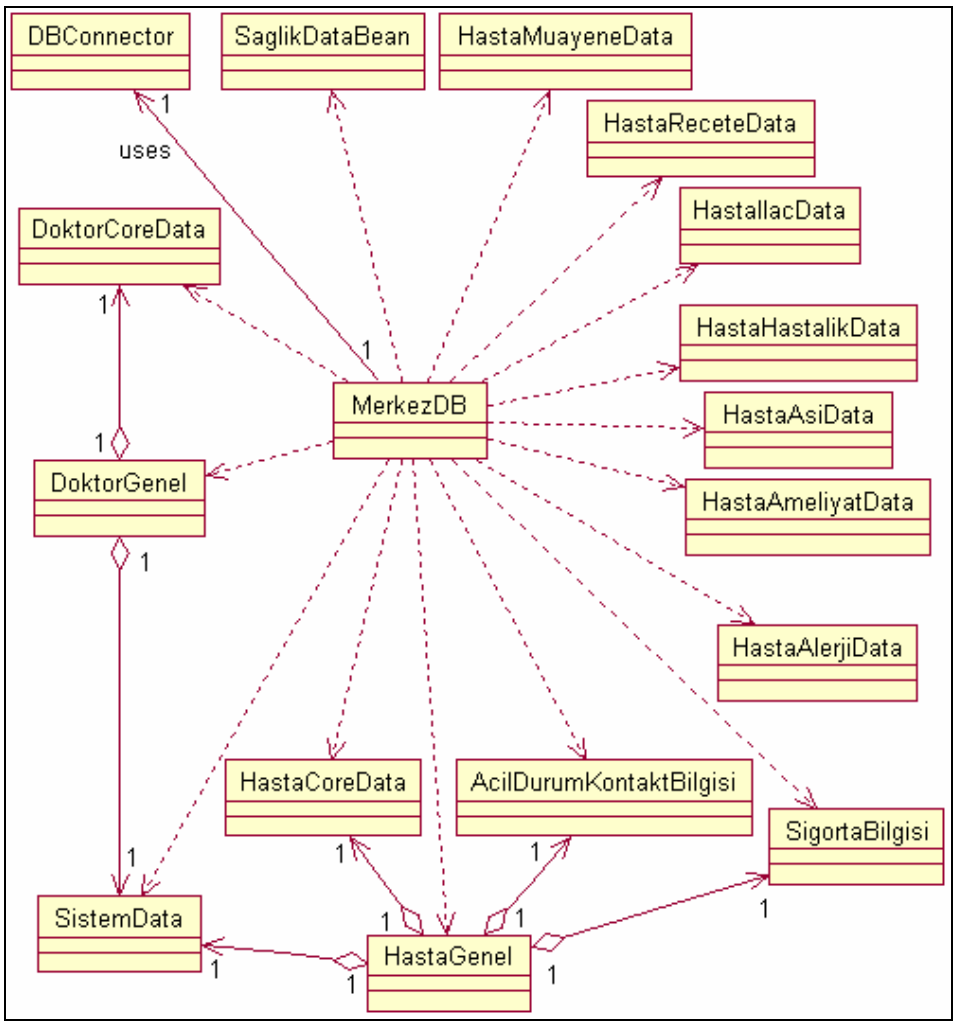
5.2.2 Veritabanı Erişim Yazılımı

Sistem veritabanına uygulamalar üzerinden erişimde görev alan yazılım paketi `tr.edu.ege.ube.akss.merkezdb` adını almıştır. Bu paket içerisinde veritabanı bağlantısını gerçekleştiren, sorgulama yapan ve veritabanındaki kayıtları temsil eden nesnelerin türetildiği sınıflar yer almaktadır.

Nesneye dayalı olarak hazırlanan sistemde, sistem kayıtlarının da nesneler ile temsil edilmesi gerekmektedir. Hazırlanan veritabanı erişim paketinde sistem ilişkisel veritabanındaki kayıtların sistemde kullanılmasını sağlayan sınıflar yer almaktadır. Şekil 5.8'de bu pakette yer alan ve sistemin veri modelini oluşturan sınıflar görülmektedir.

Sistem MVC mimarisinde model katmanını Şekil 5.8'de gösterilen sınıflardan türetilen nesneler ve bu nesneler arasındaki ilişkiler oluşturmaktadır. `MerkezDB` ve `DBConnector` sınıfları dışındaki diğer sınıflar ilişkisel veritabanındaki tablolara bilgi kaydını ve bu tablolardaki bilgilerin sistemde kullanılmasını sağlamaktadır. Diğer bir deyişle her bir kayıt kendisine karşılık gelen nesne ile sistemde temsil edilir.

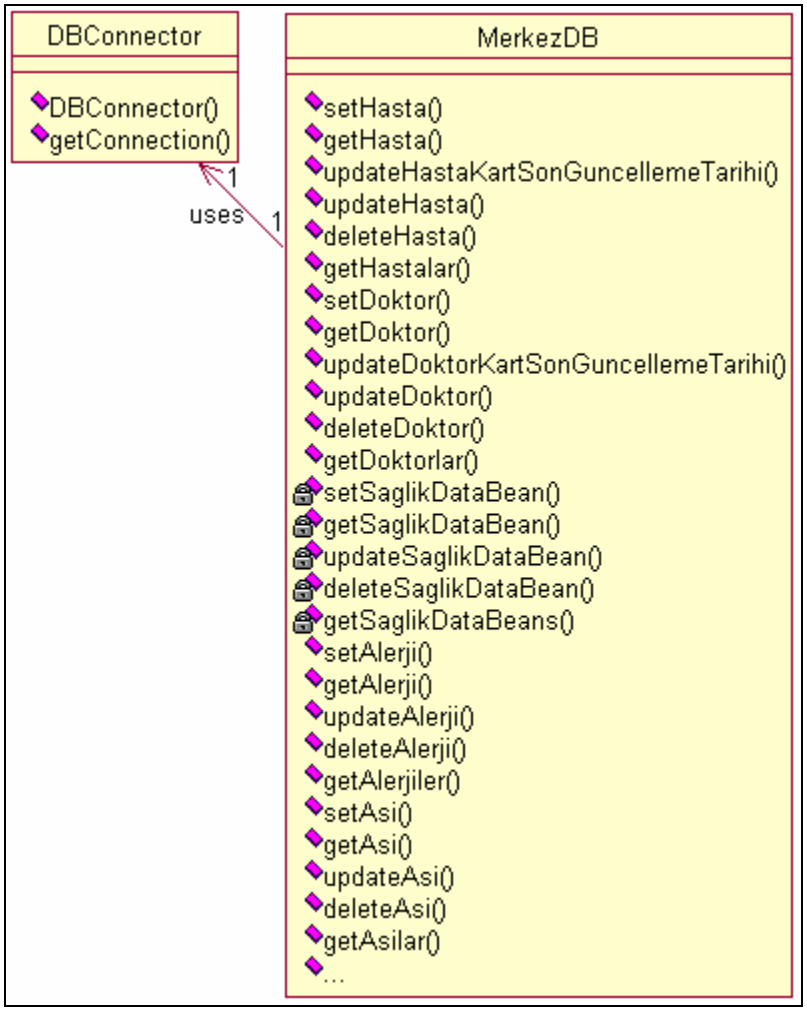
`DBConnector` ve `MerkezDB`, *tek örnek (singleton)* tasarım deseni kullanılarak hazırlanmış ve sistemde sadece tek bir örneği (instance) bulunan sınıflardır (Şekil 5.9). `DBConnector`, gerekli sistem özellik dosyasındaki veritabanı erişim bilgilerini – ki bunlar içerisinde veritabanının adresi, kullanıcı adı ve şifresi bulunur – okuyarak buna uygun veritabanı bağlantı nesnelerini (`java.sql.Connection`) hazırlar. `MerkezDB` de `DBConnector`'un hazırlamış olduğu bağlantı nesnelerini kullanarak veritabanı üzerinde gerekli sorguları gerçekleştirmektedir.



Şekil 5.8 Veritabanı erişim paketi nesne modeli

Veritabanına bilgi kaydı veya veritabanından bilgi elde etme isteğinde bulunan sistem bileşenleri sadece statik yapıdaki `MerkezDB` nesnesi ile iletişimde bulunurlar. Örneğin bir hastaya ait genel bilgileri veritabanından elde etmek isteyen bir bileşen `MerkezDB`'nin ilgili metodunu hastanın tanımlama numarası parametresi ile çağırır. `MerkezDB`, `DBConnector`'den elde ettiği bağlantı nesnesini kullanır ve veritabanı üzerinde ilgili sorguyu gerçekleştirerek hastaya ait bilgileri

elde eder. Bu bilgileri de türettiği `HastaGenel` nesnesine özellik olarak atar ve istemci bileşene gönderir. İstemci bileşen bu `HastaGenel` nesnesinin *getX* metotlarını kullanarak bilgilere erişir.



Şekil 5.9 DBConnector ve MerkezDB sınıfları

Görüldüğü gibi böyle bir tasarım özellikle sistem ara yüz bileşenlerinin veritabanı tablo yapılarından bağımsız olmalarına imkan vermiştir. Çünkü gerekli olan tüm veritabanı sorguları erişim paketinde yer alan tek bir sınıfta yer almaktadır – ki bu sınıf `MerkezDB`'dir. Benzer

şekilde `MerkezDB` de sadece tablo yapıları ile ilgilidir ve standart SQL (Sequential Query Language) cümlelerini içerir. Yani kullanılan veritabanı yazılımından bağımsızdır. Örneğin sistem veritabanı MS SQL Server yerine başka bir veritabanına taşınsa da bundan etkilenmeyecektir. Çünkü sorguları işletmek üzere kullandığı bağlantı nesnelerini `DBConnector`'den hazır olarak alır. `DBConnector` de bağlantı nesnesi özelliklerini gerekli özellik dosyasından parametre halinde okuduğu için veritabanının değiştirilmesi gibi bir durumdan etkilenmeyecektir.

5.2.3 Veritabanı kayıtlarının sistem yazılımlarında kullanılması

Bu alt bölümde sistem ilişkisel veritabanında yer alan hangi kayıtların erişim paketindeki sınıflardan türetilen hangi nesneler ile sistemde temsil edildikleri ve kullanıldıkları yer almaktadır.

5.2.3.1 HASTA_GENEL tablosu kayıtlarının kullanılması

Sistem kullanıcısı olan hastalara ait genel bilgiler Şekil 5.10'da gösterilen HASTA_GENEL tablosu içerisinde saklanmaktadır. Bu tabloda yer alan her bir kayıt ise sistemde, `tr.edu.ege.ube.akss.merkezdb.HastaGenel` sınıfından türetilen bir nesne ile temsil edilmektedir.

Şekil 5.11'de gösterilen `HastaGenel` nesnesi hastaya ait diğer hasta bilgileri ve DES anahtarını doğrudan özellik olarak saklarken diğer genel bilgileri `HastaCoreData`, `AcilDurumKontaktBilgisi`, `SigortaBilgisi` ve `SistemData` sınıflarından türettiği birer örnekte barındırmaktadır.

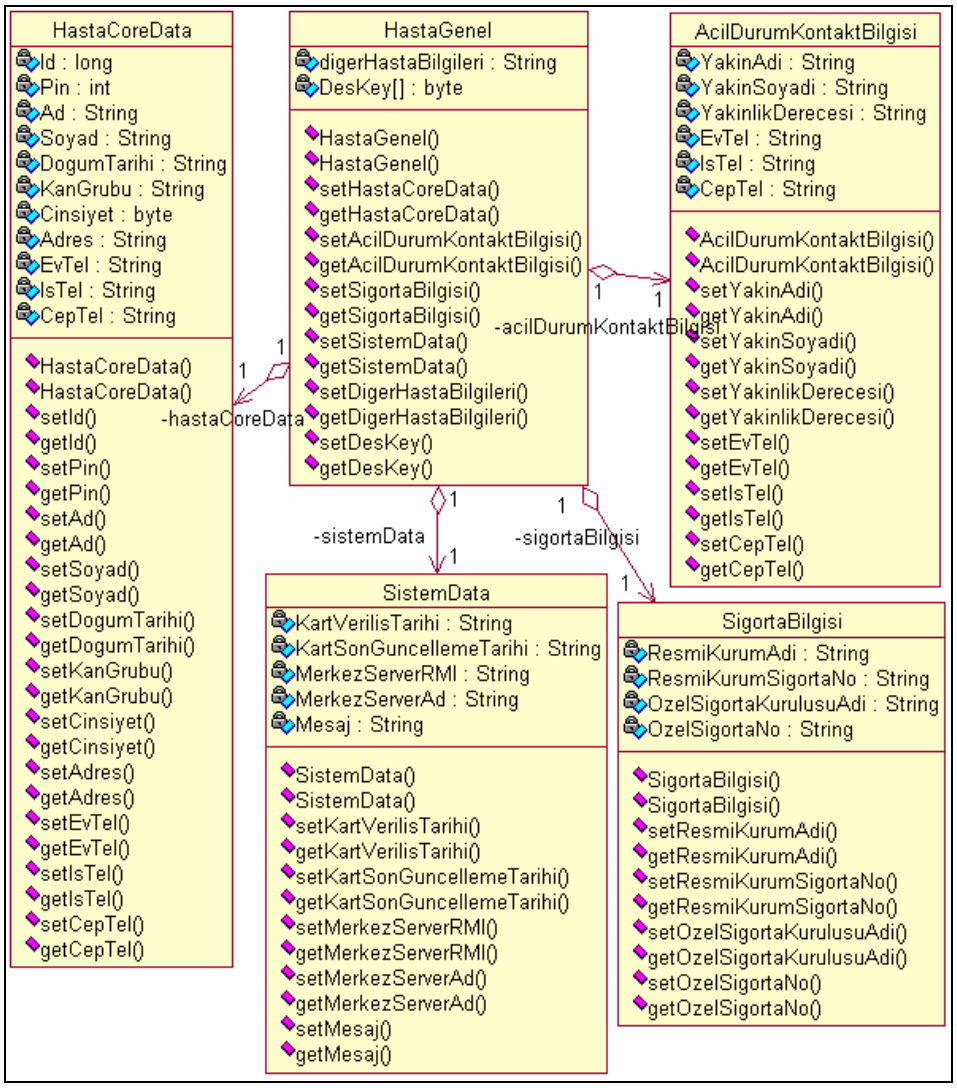
Sisteme, hastaya ait genel bilgilerin kaydedilmesi, bu bilgilerin elde edilmesi ve güncellemesi `HastaGenel` nesneleri aracılığı ile

gerçekleşmektedir. Daha önce de belirtildiği gibi hasta genel bilgisi isteğinde bulunan bileşen `MerkezDB`'nin `getHasta()` metodunu, bilgilerini istediği hastanın tanışmama numarası ile çağırır. Bu metottan kendisine dönen, ilgili hastaya ait genel bilgileri barındıran `HastaGenel` nesnesidir. Yeni bir hasta kaydında kaydedilecek bilgiler yine türetilen bir `HastaGenel` nesnesi içerisinde saklanarak bu nesne `MerkezDB`'nin `setHasta()` metoduna parametre olarak gönderilir. Gelen nesnenin özellikleri `MerkezDB` tarafından veritabanına aktarılır. Bilgi güncellemesi de set işlemi ile benzerlik göstermektedir.

HASTA_GENEL

Column Name	Data Type	Length	Allow Nulls
ID	bigint	8	
PIN	int	4	✓
AD	nvarchar	20	✓
SOYAD	nvarchar	20	✓
DOGUM_TARIHI	smalldatetime	4	✓
KAN_GRUBU	nvarchar	10	✓
CINSIYET	nchar	1	✓
ADRES	nvarchar	100	✓
TEL_EV	nchar	11	✓
TEL_IS	nchar	11	✓
TEL_CEP	nchar	11	✓
KART_VERILIS_TARIHI	smalldatetime	4	✓
KART_SON_GUNCELLEME_TARIHI	smalldatetime	4	✓
SIGORTA_KURUM_AD	nvarchar	30	✓
SIGORTA_KURUM_NO	nvarchar	20	✓
SIGORTA_OZEL_AD	nvarchar	30	✓
SIGORTA_OZEL_NO	nvarchar	20	✓
YAKIN_AD	nvarchar	20	✓
YAKIN_SOYAD	nvarchar	20	✓
YAKIN_DERECE	nvarchar	20	✓
YAKIN_EV_TEL	nchar	11	✓
YAKIN_IS_TEL	nchar	11	✓
YAKIN_CEP_TEL	nchar	11	✓
DIGER_HASTA_BILGILERI	nvarchar	255	✓
MERKEZ_SERVER_RMI	nvarchar	21	✓
MERKEZ_SERVER_AD	nvarchar	30	✓
DES_KEY	varbinary	1024	✓
MESAJ	nvarchar	500	✓

Şekil 5.10 HASTA_GENEL tablosu

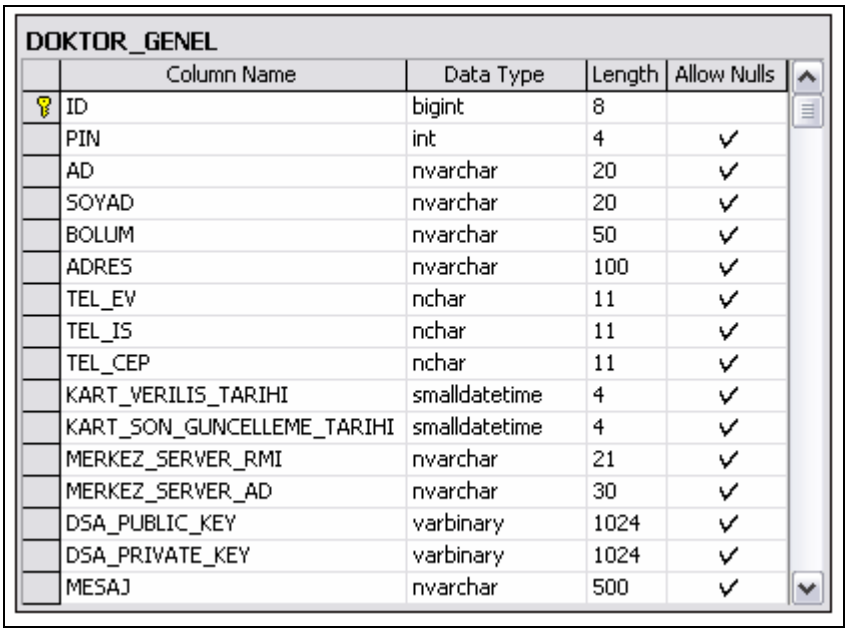


Şekil 5.11 HASTA_GENEL tablosu içerisindeki bilgileri temsil eden sınıflar

5.2.3.2 DOKTOR_GENEL tablosu kayıtlarının kullanılması

Sistem kullanıcısı olan doktorlara ait genel bilgiler Şekil 5.12'de gösterilen DOKTOR_GENEL tablosu içerisinde saklanmaktadır. Bu tabloda yer alan her bir kayıt ise sistemde, `tr.edu.ege.ube.akss.merkezdb.DoktorGenel` sınıfından türetilen bir nesne ile temsil edilmektedir.

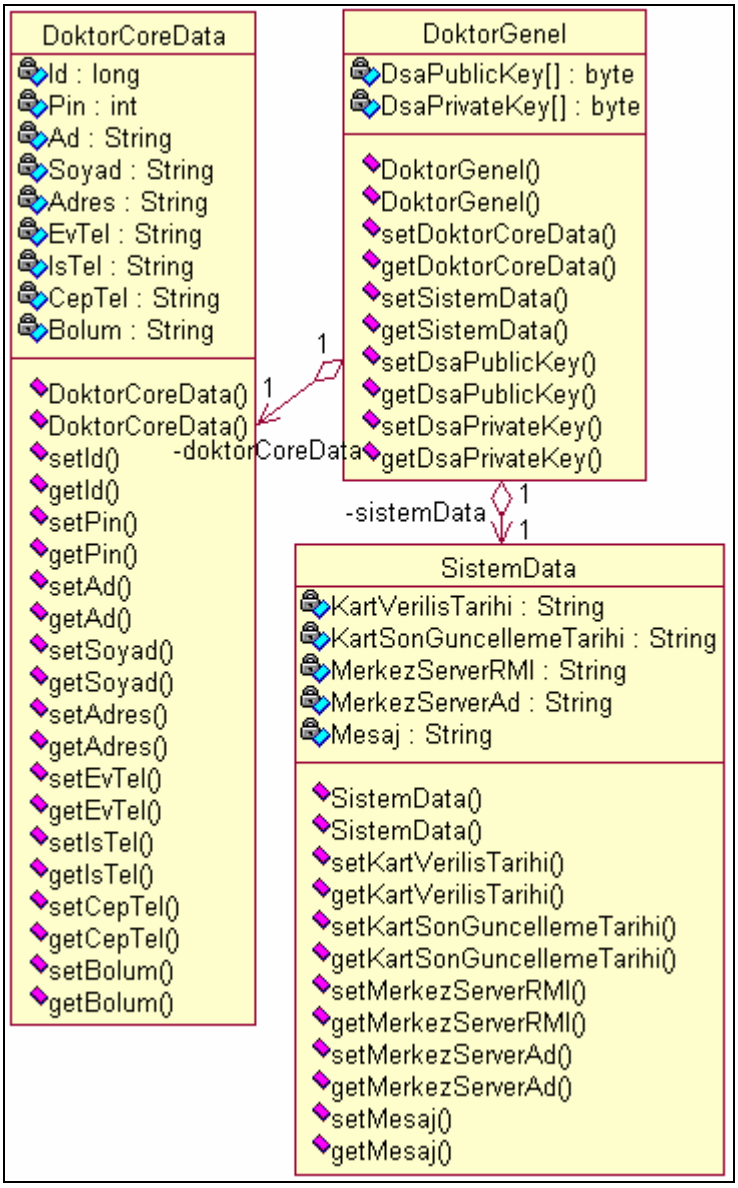
Şekil 5.13'te gösterilen `DoktorGenel` nesnesi doktora ait dijital imzalama anahtarlarını doğrudan özellik olarak saklarken diğer genel bilgileri `DoktorCoreData` ve `SistemData` sınıflarından türettiği birer örnekte barındırmaktadır.



DOKTOR_GENEL

Column Name	Data Type	Length	Allow Nulls
ID	bigint	8	
PIN	int	4	✓
AD	nvarchar	20	✓
SOYAD	nvarchar	20	✓
BOLUM	nvarchar	50	✓
ADRES	nvarchar	100	✓
TEL_EV	nchar	11	✓
TEL_IS	nchar	11	✓
TEL_CEP	nchar	11	✓
KART_VERILIS_TARIHI	smalldatetime	4	✓
KART_SON_GUNCELLEME_TARIHI	smalldatetime	4	✓
MERKEZ_SERVER_RMI	nvarchar	21	✓
MERKEZ_SERVER_AD	nvarchar	30	✓
DSA_PUBLIC_KEY	varbinary	1024	✓
DSA_PRIVATE_KEY	varbinary	1024	✓
MESAJ	nvarchar	500	✓

Şekil 5.12 DOKTOR_GENEL tablosu



Şekil 5.13 DOKTOR_GENEL tablosu içerisindeki bilgileri temsil eden sınıflar

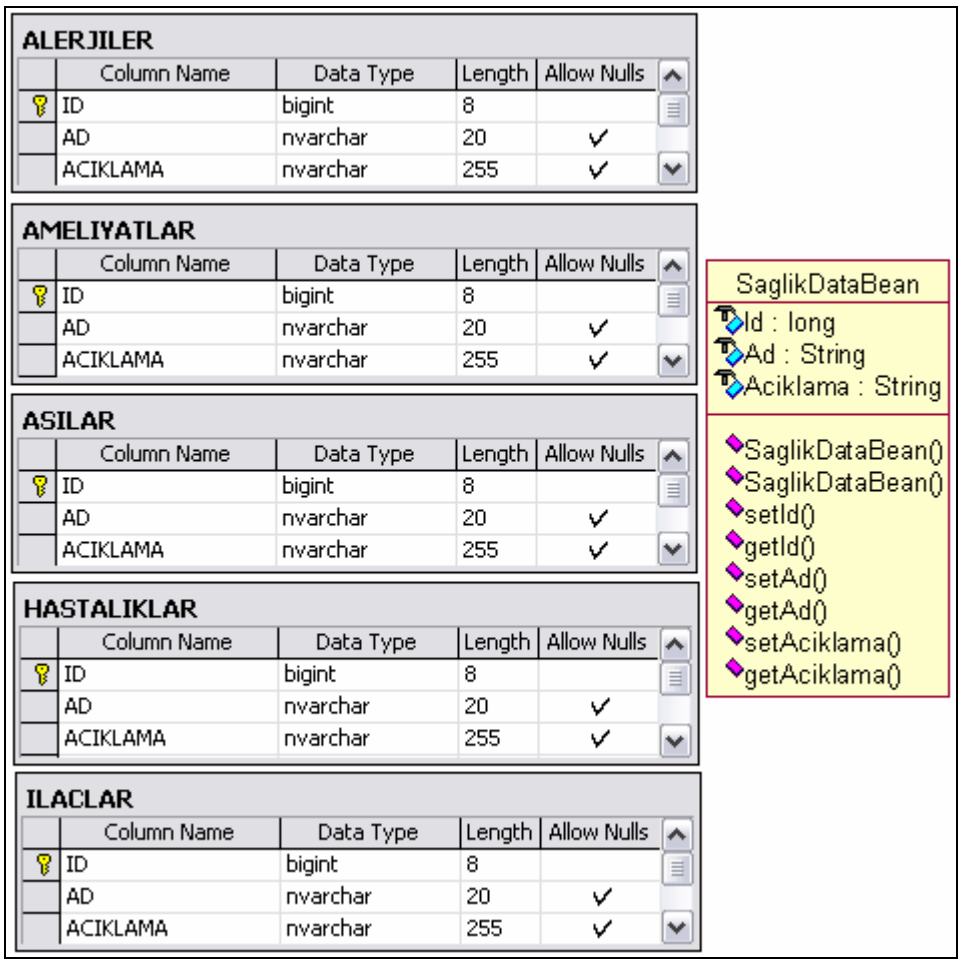
Doktora ait genel bilgilerin kaydedilmesi, bu bilgilerin elde edilmesi ve güncellemesi `DoktorGenel` nesneleri aracılığı ile gerçekleşmektedir. Tıpkı hastalarda olduğu gibi bilgi kaydetmek isteyen sistem bileşenleri elde ettikleri doktor bilgilerini türettikleri bir

`DoktorGenel` nesnesine özellik olarak atayarak bu nesneyi `MerkezDB`'nin ilgili *setX* metoduna gönderirler. Bu metot içerisinde gelen nesnenin özellikleri alınarak DOKTOR_GENEL tablosuna kaydedilir. Tablodan bir doktora ait bilgiler alınmak istendiğinde yine `MerkezDB`, ilgili tablo satırını temsil eden `DoktorGenel` nesnesini hazırlayarak bu kayda ait bilgileri nesneye özellik olarak atar ve bilgilere erişmek isteyen bileşene hazırlanan bu `DoktorGenel` nesnesini gönderir.

5.2.3.3 Sağlık bilgisi tablolarında yer alan kayıtların kullanılması

Daha önce de belirtildiği gibi sistem veritabanında alerji, ameliyat, aşı, hastalık ve ilaç bilgileri sırası ile ALERJILER, AMELIYATLAR, ASILAR, HASTALIKLAR ve ILACLAR tablolarında yer alır. Bu tablolarda yer alan kayıtlar sistemde, `tr.edu.ege.ube.akss.merkezdb.SaglikDataBean` sınıfından türetilen nesneler ile temsil edilmektedir. Şekil 5.14'te bu sağlık bilgisi tabloları ve bunlarda yer alan kayıtları temsil eden `SaglikDataBean` sınıfı gösterilmiştir.

Sisteme yeni bir alerji bilgisi kaydedilmek istendiğinde bu bilgiler türetilen bir `SaglikDataBean` nesnesi içerisine konularak `MerkezDB`'nin `setAlerji()` metoduna parametre olarak gönderilir. Kayıt işlemi için gereken sorgu bu metotta yer alır. Veritabanından alerji bilgisi ise yine MerkezDB'nin `getAlerji()` metodu kullanılarak alınır. Bu metottan dönen de istenilen bilgileri barındıran `SaglikDataBean` nesnesidir. Benzer kullanım diğer sağlık bilgileri için de geçerlidir.



Şekil 5.14 Sağlık bilgisi tabloları ve bu tablolarda yer alan kayıtları temsil eden SaglikDataBean sınıfı

5.2.3.4 Hasta sağlık bilgisi tablolarında yer alan kayıtların kullanılması

Hastalara ait alerji, ameliyat, aşı, hastalık, sürekli kullandığı ilaç, muayene ve reçete kayıtları sırası ile HASTA_ALERJI, HASTA_AMELIYAT, HASTA_ASI, HASTA_ILAC, HASTA_MUAYENE ve HASTA_RECETE tablolarında yer almaktadır. HASTA_GENEL tablosu ile bu tablolar arasında 1..n ilişkisi

bulunmaktadır. HASTA_GENEL tablosunda birincil anahtar olan hasta tanımlama numarası bu tablolardaki hasta tanımlama numarası alanları ile yabancı (foreign) anahtar olarak ilişkilendirilmiştir. Buna göre sistemde olmayan bir hastaya ait alerji, ameliyat, vb. hiçbir bilgi veritabanında yer alamaz. Aynı durum sağlık bilgisi alanları için de geçerlidir. Şöyle ki, örneğin sistemde kayıtlı bir alerji bilgisi yoksa HASTA_ALERJI tablosunda da bir hasta ile ilişkilendirilmiş bu alerji bilgisine ait kayıt olmaz.

Aşağıdaki alt başlıklarda her bir hasta sağlık bilgisi tablosu ve bu tabloda yer alan kayıtları temsil eden nesnelerin türetildiği sınıflar hakkında bilgiler yer almaktadır.

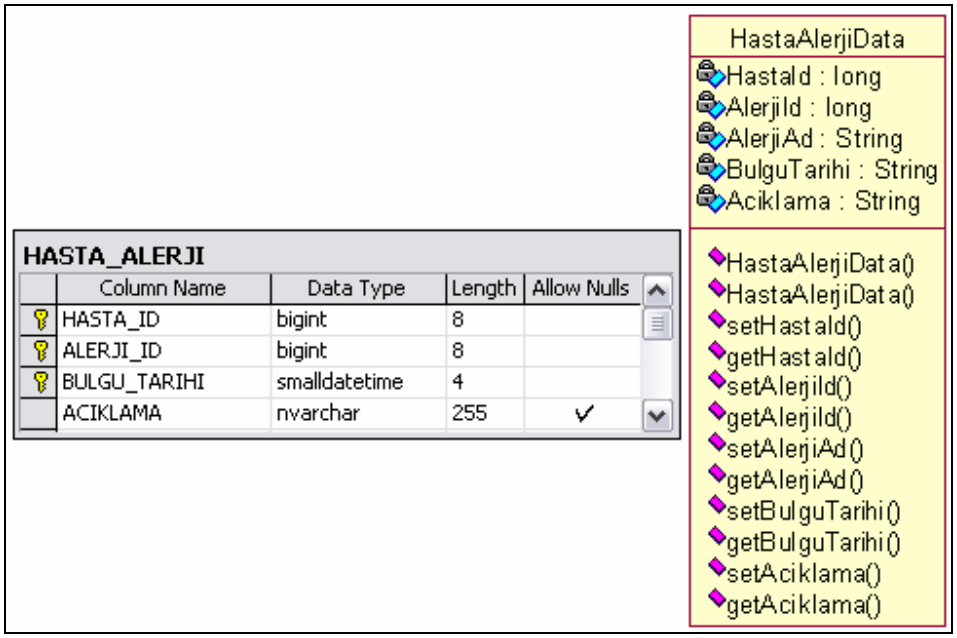
HASTA_ALERJI tablosu ve HastaAlerjiData sınıfı

Şekil 5.15'de sistem veritabanında yer alan HASTA_ALERJI tablosu ve bu tablodaki kayıtları sistemde temsil eden `tr.edu.ege.ube.akss.merkezdb. HastaAlerjiData` sınıfı gösterilmiştir.

Sisteme bir hastaya ait alerji bilgisinin kaydedilmesi bir anlamda HASTA_GENEL tablosunda yer alan bir hasta ile ALERJILER tablosunda yer alan bir kaydın ilişkilendirilmesi anlamına gelir. HASTA_ALERJI tablosunda hasta ve alerji tanımlama numaraları dışında hastadaki alerji bulgusunun tarihi ve varsa açıklama bilgisi yer alır. Hasta ve alerji tanımlama numaraları bulgu tarihi ile beraber tablonun birincil anahtarını oluştururlar.

Veritabanına kaydedilecek hasta alerji bilgisi `HastaAlerjiData` nesnesi içerisinde `MerkezDB`'nin `setHastaAlerji()` metoduna gönderilir. Bir hastaya ait tüm alerji kayıtları elde edilmek istendiğinde `MerkezDB`'nin `getHastaAlerjileri()` metodu kullanılmaktadır. Metoda hastanın tanımlama numarası parametre olarak verilir. Metot içerisinde ilgili sorgu gerçekleştirilir ve her bir kayda karşılık gelen

`HastaAlerjiData` nesneleri hazırlanır. Metottan, bu `HastaAlerjiData` nesnelerinden oluşan `java.util.Vector` nesnesi istemciye dönmektedir.

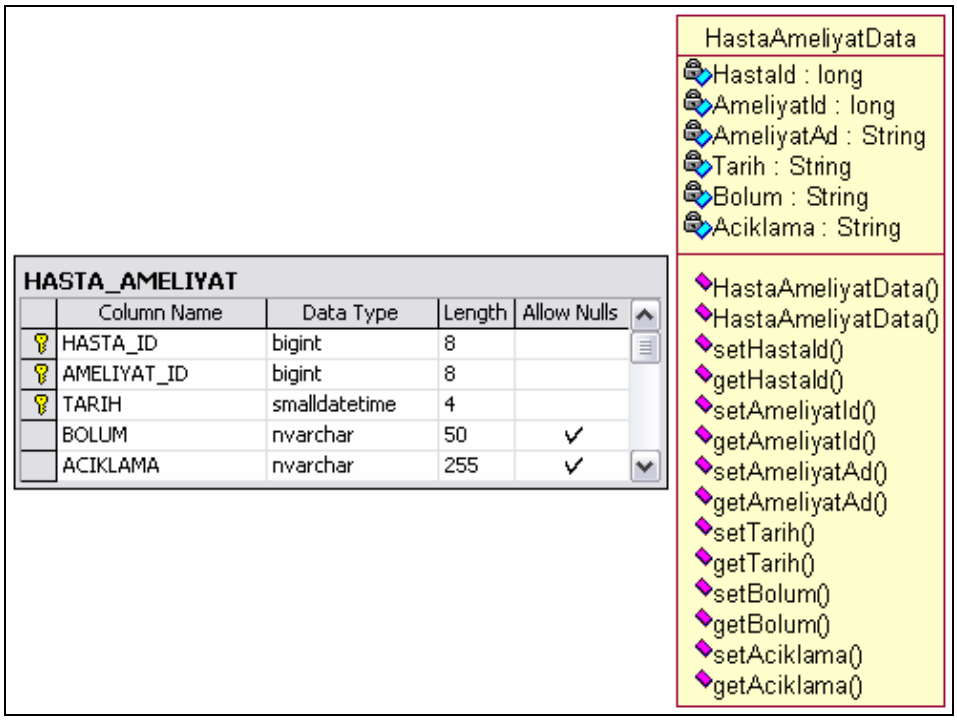


Şekil 5.15 HASTA_ALERJI tablosu ve bu tabloda yer alan kayıtları temsil eden HastaAlerjiData sınıfı

Herhangi bir hasta alerji kaydının düzeltilmesi veya silinmesi parametre olarak `HastaAlerjiData` nesneleri alan `MerkezDB`'nin sırası ile `updateHastaAlerjiData()` ve `deleteHastaAlerjiData()` metotları ile gerçekleştirilir. Hasta alerji kayıtları için anlatılan işleyiş diğer hasta sağlık bilgilerinde de aynıdır. Tek fark iletişimde kullanılan nesneler ve `MerkezDB`'nin ilgili *setX*, *getX*, *updateX* ve *deleteX* metotlarıdır.

HASTA_AMELIYAT tablosu ve HastaAmeliyatData sınıfı:

Şekil 5.16'da sistem veritabanında yer alan HASTA_AMELIYAT tablosu ve bu tablodaki kayıtları sistemde temsil eden nesnelerin türetildiği `tr.edu.ege.ube.akss.merkezdb.HastaAmeliyatData` sınıfı gösterilmiştir.



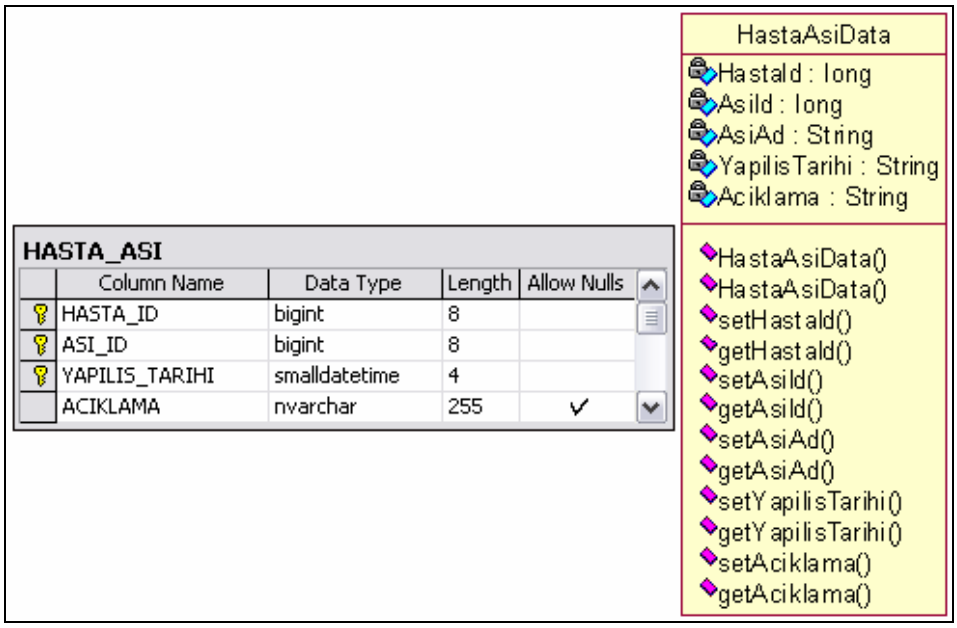
Şekil 5.16 HASTA_AMELIYAT tablosu ve bu tabloda yer alan kayıtları temsil eden HastaAmeliyatData sınıfı

HASTA_AMELIYAT tablosunda yer alan her bir kayıt, bir hastaya ait tek bir ameliyat bilgisini tutmaktadır. Hasta ve ameliyata ait tanımlama numaraları dışında ameliyatın yapıldığı tarih, bölüm ve özet açıklama bilgisi de kayıt içinde yer alır. Hasta ve ameliyat tanımlama numaraları tarih ile beraber tablonun birincil anahtarını oluştururlar.

Sistemde kullanılacak olan her bir hasta ameliyat kaydı `HastaAmeliyatData` sınıfından türetilen bir nesne ile temsil edilir. Bilgilerin veritabanına kaydı, veritabanından alınması, silinmesi ve veritabanında güncellenmesinde `MerkezDB`'nin sırası ile `setHastaAmeliyat()`, `getHastaAmeliyatlari()`, `deleteHastaAmeliyat()` ve `updateHastaAmeliyat()` metotları kullanılır.

HASTA_ASI tablosu ve HastaAsiData sınıfı:

Şekil 5.17'de sistem veritabanında yer alan HASTA_ASI tablosu ve bu tablodaki kayıtları sistemde temsil eden nesnelerin türetildiği `tr.edu.ege.ube.akss.merkezdb.HastaAsiData` sınıfı gösterilmiştir.



Şekil 5.17 HASTA_ASI tablosu ve bu tabloda yer alan kayıtları temsil eden HastaAsiData sınıfı

HASTA_ASI tablosunda yer alan her bir kayıt, bir hastaya ait tek bir aşı bilgisini tutmaktadır. Hasta ve aşıya ait tanımlama numaraları dışında aşının yapıldığı tarih ve varsa açıklama bilgisi de kayıt içinde yer alır. Hasta ve aşı tanışmama numaraları tarih ile beraber tablonun birincil anahtarını oluştururlar.

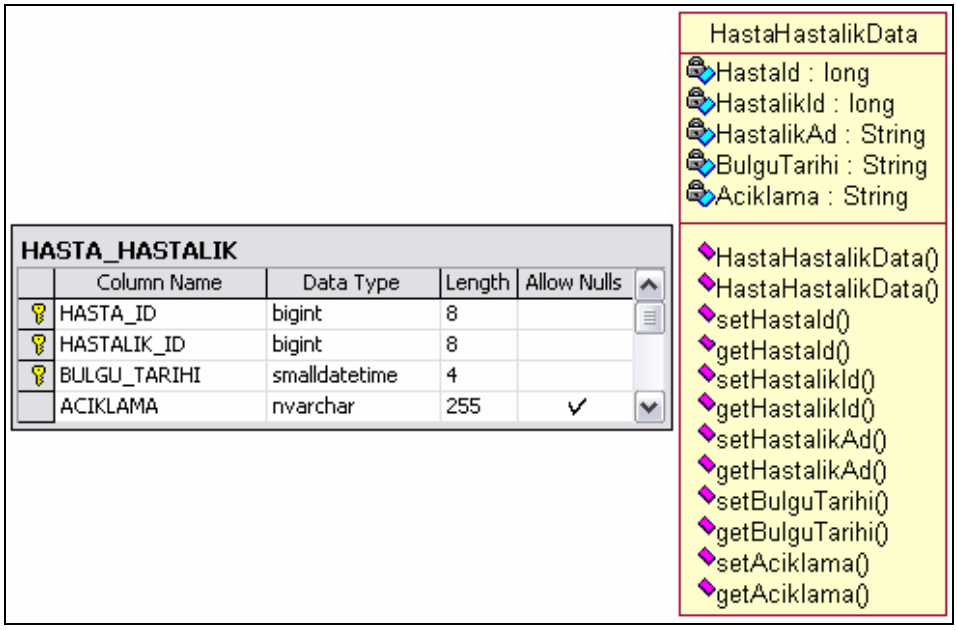
Sistemde kullanılacak olan her bir hasta aşı kaydı `HastaAsiData` sınıfından türetilen bir nesne ile temsil edilir. Bilgilerin veritabanına kaydı, veritabanından alınması, silinmesi ve veritabanında güncellenmesinde `MerkezDB`'nin sırası ile `setHastaAsi()`, `getHastaAsilari()`, `deleteHastaAsi()` ve `updateHastaAsi()` metotları kullanılır.

HASTA_HASTALIK tablosu ve HastaHastalikData sınıfı:

Şekil 5.18'de sistem veritabanında yer alan HASTA_HASTALIK tablosu ve bu tablodaki kayıtları sistemde temsil eden nesnelerin türetildiği `tr.edu.ege.ube.akss.merkezdb.HastaHastalikData` sınıfı gösterilmiştir.

HASTA_HASTALIK tablosunda yer alan her bir kayıt, bir hastaya ait tek bir hastalık bilgisini tutmaktadır. Hasta ve hastalığa ait tanımlama numaraları dışında hastalık bulgu tarihi ve varsa açıklama bilgisi de kayıt içinde yer alır. Hasta ve hastalık tanımlama numaraları tarih ile beraber tablonun birincil anahtarını oluştururlar.

Sistemde kullanılacak olan her bir hasta hastalık kaydı `HastaHastalikData` sınıfından türetilen bir nesne ile temsil edilir. Bilgilerin veritabanına kaydı, veritabanından alınması, silinmesi ve veritabanında güncellenmesinde `MerkezDB`'nin sırası ile `setHastaHastalik()`, `getHastaHastaliklari()`, `deleteHastaHastalik()` ve `updateHastaHastalik()` metotları kullanılır.

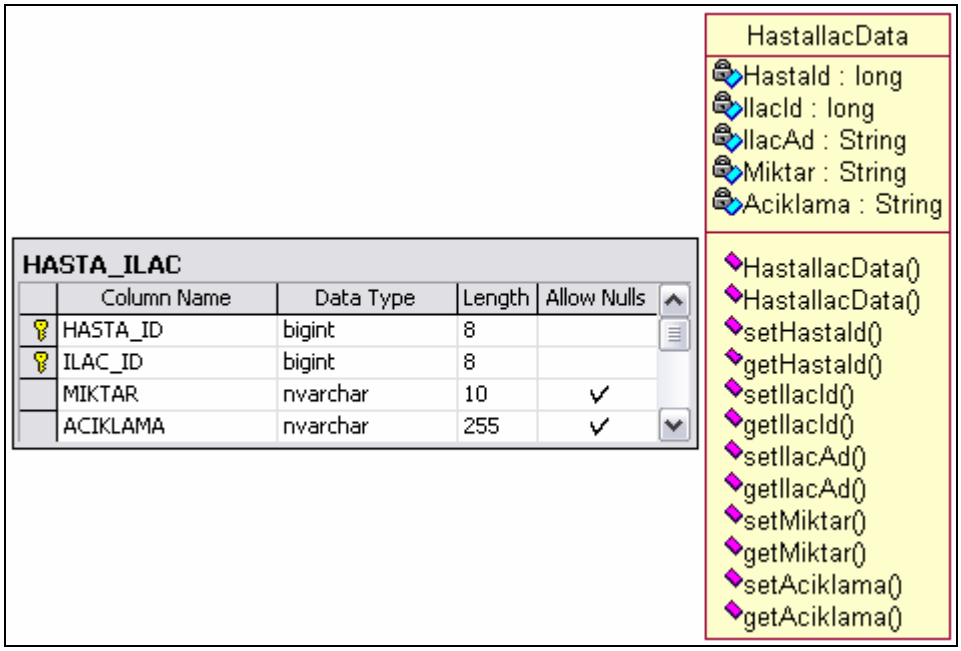


Şekil 5.18 HASTA_HASTALIK tablosu ve bu tabloda yer alan kayıtları temsil eden HastaHastalikData sınıfı

HASTA_ILAC tablosu ve HastaIlacData sınıfı:

Şekil 5.19'da sistem veritabanında yer alan HASTA_ILAC tablosu ve bu tablodaki kayıtları sistemde temsil eden nesnelerin türetildiği `tr.edu.ege.ube.akss.merkezdb.HastaIlacData` sınıfı gösterilmiştir.

HASTA_ILAC tablosunda yer alan her bir kayıt, bir hastanın sürekli kullandığı ilaç bilgisini tutmaktadır. Hasta ve ilaca ait tanımlama numaraları dışında ilacın kullanım miktarı ve varsa açıklama bilgisi de kayıt içinde yer alır. Hasta ve ilaç tanımlama numaraları birlikte tablonun birincil anahtarını oluştururlar.

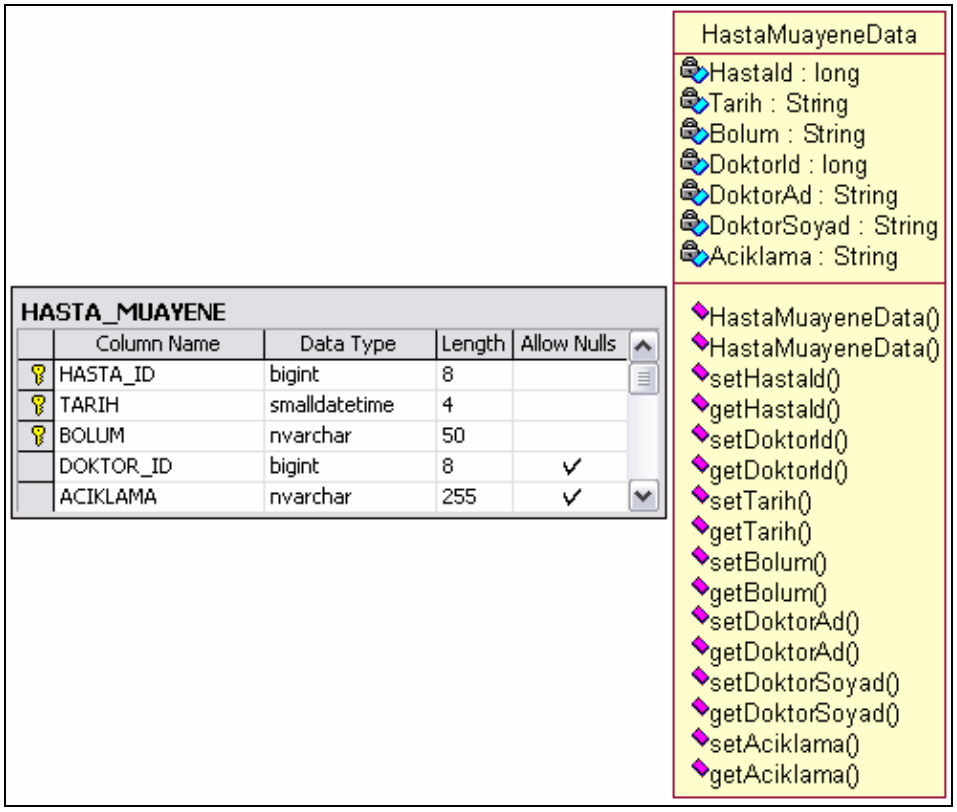


Şekil 5.19 HASTA_ILAC tablosu ve bu tabloda yer alan kayıtları temsil eden HastaIlacData sınıfı

Sistemde kullanılacak olan her bir hasta sürekli kullanılan ilaç kaydı `HastaIlacData` sınıfından türetilen bir nesne ile temsil edilir. Bilgilerin veritabanına kaydı, veritabanından alınması, silinmesi ve veritabanında güncellenmesinde `MerkezDB`'nin sırası ile `setHastaIlac()`, `getHastaIlaclari()`, `deleteHastaIlac()` ve `updateHastaIlac()` metotları kullanılır.

HASTA_MUAYENE tablosu ve HastaMuayeneData sınıfı:

Şekil 5.20'de sistem veritabanında yer alan HASTA_MUAYENE tablosu ve bu tablodaki kayıtları sistemde temsil eden nesnelerin türetildiği `tr.edu.ege.ube.akss. merkezdb.HastaMuayeneData` sınıfı gösterilmiştir.



Şekil 5.20 HASTA_MUAYENE tablosu ve bu tabloda yer alan kayıtları temsil eden HastaMuayeneData sınıfı

HASTA_MUAYENE tablosunda yer alan her bir kayıt, bir hastaya ait tek bir muayene bilgisini tutmaktadır. Hasta tanımlama numarası dışında muayenenin gerçekleştirildiği tarih, hastane bölümü, muayeneyi gerçekleştiren doktorun tanımlama numarası ve varsa açıklama bilgisi de kayıt içinde yer alır. Hasta tanımlama numarası, tarih ve bölüm alanları birlikte tablonun birincil anahtarını oluştururlar.

HASTA_MUAYENE tablosu HASTA_GENEL ile olan n..1 ilişkisi dışında DOKTOR_GENEL tablosu ile de n..1 ilişki içerisindedir. Buna göre hem HASTA_GENEL tablosuna hem de DOKTOR_GENEL tablosuna bağımlıdır.

Sistemde kullanılacak olan her bir hasta muayene kaydı `HastaMuayeneData` sınıfından türetilen bir nesne ile temsil edilir. Bilgilerin veritabanına kaydı, veritabanından alınması, silinmesi ve veritabanında güncellenmesinde `MerkezDB`'nin sırası ile `setHastaMuayene()`, `getHastaMuayeneleri()`, `deleteHastaMuayene()` ve `updateHastaMuayene()` metotları kullanılır.

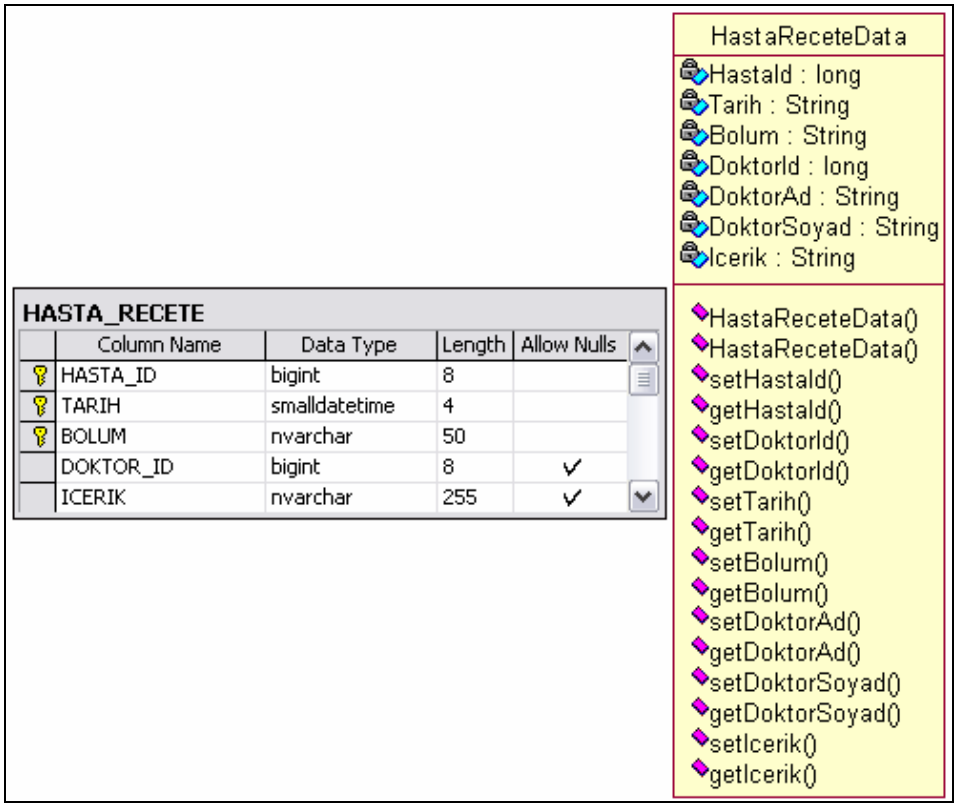
HASTA_RECETE tablosu ve HastaReceteData sınıfı:

Şekil 5.21'de sistem veritabanında yer alan HASTA_RECETE tablosu ve bu tablodaki kayıtları sistemde temsil eden nesnelerin türetildiği `tr.edu.ege.ube.akss. merkezdb.HastaReceteData` sınıfı gösterilmiştir.

HASTA_RECETE tablosunda yer alan her bir kayıt, bir hastaya ait tek bir reçete bilgisini tutmaktadır. Hasta tanımlama numarası dışında reçetenin yazıldığı tarih, hastane bölümü, reçeteyi yazan doktorun tanımlama numarası ve reçete içeriği kayıt içinde yer alır. Hasta tanımlama numarası, tarih ve bölüm alanları birlikte tablonun birincil anahtarını oluştururlar.

HASTA_RECETE tablosu HASTA_MUAYENE tablosu gibi HASTA_GENEL ile olan n..1 ilişkisi dışında DOKTOR_GENEL tablosu ile de n..1 ilişki içerisindedir.

Sistemde kullanılacak olan her bir hasta reçete kaydı `HastaReceteData` sınıfından türetilen bir nesne ile temsil edilir. Bilgilerin veritabanına kaydı, veritabanından alınması, silinmesi ve veritabanında güncellenmesinde `MerkezDB`'nin sırası ile `setHastaRecete()`, `getHastaReceteleri()`, `deleteHastaRecete()` ve `updateHastaRecete()` metotları kullanılır.



Şekil 5.21 HASTA_RECETE tablosu ve bu tabloda yer alan kayıtları temsil eden HastaReceteData sınıfı

6 SİSTEM İSTEMCİ ARA YÜZ BİLEŞENLERİ

Tez kapsamında hazırlanan sistemde kullanıcılar ile etkileşimde bulunacak bir çok yazılım bileşeni bulunmaktadır. İki ayrı pakette yer alan bu ara yüz bileşenleri sistem mimarisinde görünüm katmanını oluşturmaktadırlar.

İzleyen iki alt bölümde sırası ile sistem yönetiminde ve kliniklerde çalışan ara yüz yazılımları hakkında bilgi verilmiştir.

6.1 Sistem Yönetim Yazılımı

Bu bölümde, hazırlanan sağlık sisteminin yönetilmesi için gerekli olan ve kullanıcı ara yüz bileşenlerini içeren sistem yazılımı hakkında bilgi verilmektedir. İlk olarak yazılımın genel tanıtımı ve tasarımı anlatılmıştır. Daha sonra yazılımın yerine getirdiği fonksiyonlar hem kullanıcı gözü ile hem de yazılım mimarisi açısından ele alınmaktadır.

6.1.1 Yazılımın genel tanıtımı

Sistem yönetim yazılımı adından da anlaşılacağı gibi hazırlanan sağlık sisteminin yönetilmesi ile ilgili fonksiyonları yerine getirir. Sistem asıl kullanıcıları olan hasta ve doktorlar ile ilgili temel veritabanı işlemleri bu yazılım aracılığı ile gerçekleştirir.

Tasarımda bu yazılımın hastanenin ilgili yönetim birimlerinde yer alması ve bu birimde bulunan görevliler tarafından kullanılması öngörülmüştür. Yani yazılımın etkileşimde bulunduğu kullanıcı grubu doktorlar veya hastalar değildir.

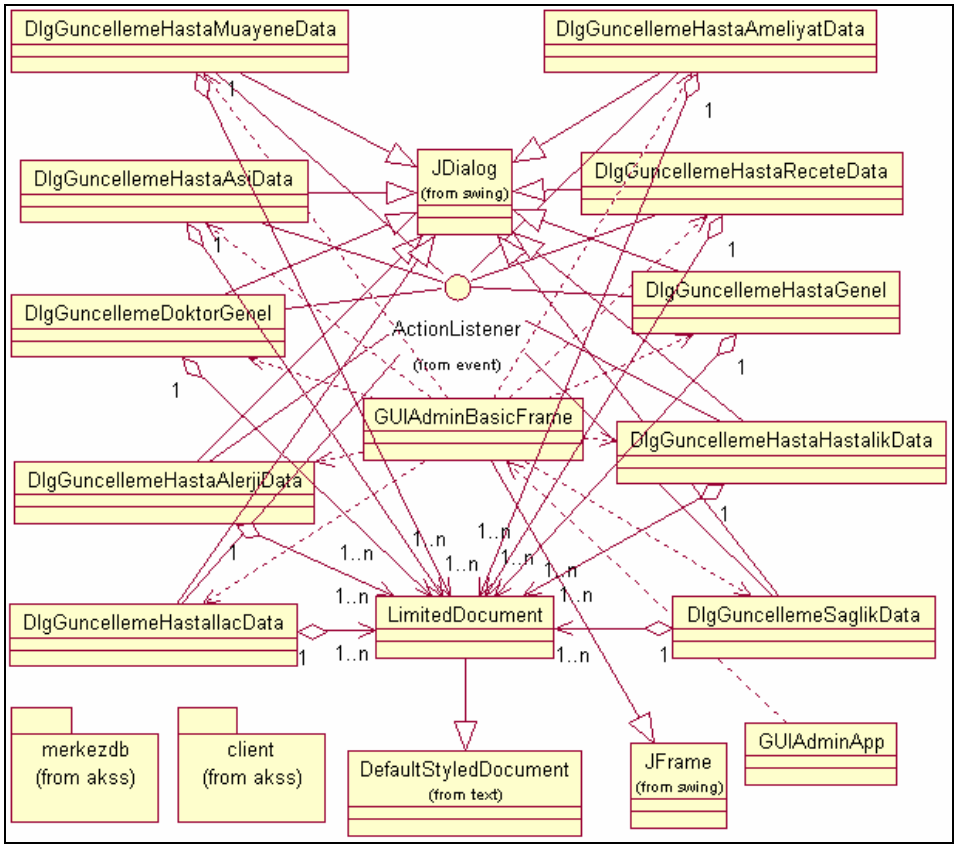
Yazılım, merkez veritabanlarının kullanılması ile ilgili tüm işlemleri yerine getirir. Sisteme hasta ve doktorlara ait bilgilerin kaydedilmesi, bu bilgilerin gerektiğinde güncellenmesi, silinmesi ve gerekli listelerin alınması bu işlemlerden bazılarıdır. Bunların yanı sıra yazılım, hasta ve doktor kartları ile ilgili yönetimsel işlemlerin de yerine getirilmesine olanak tanır. Sisteme kaydolan bir hastaya veya doktora ait akıllı kartların hazırlanarak ilgili kart sahibine teslimi, akıllı kart üzerinde gerekli olan bilgi güncellemeleri, muayene sonucunda hasta kartına kliniklerde doktor tarafından yazılan muayene ve varsa reçete bilgilerinin veritabanına aktarılması ve reçeteye onay verilmesi, vb. akıllı kartlar ile ilgili işlemler yöneticilere sunulan görsel ara yüzler ile yerine getirilmektedir.

6.1.2 Yazılım mimarisi

Sistem yönetim yazılımı, muayenehanelerde yer alması öngörülen klinik yazılımı ile birlikte sistem MVC mimarisinin *görünüm* katmanında yer almaktadır. İş mantığı değil de kullanıcı ile etkileşimi sağlayacak olan görsel kullanıcı ara yüzü bileşenlerini içermektedir.

Yazılım paketi `tr.edu.ege.ube.akss.gui.admin` olarak adlandırılmıştır. Tüm diğer sistem paketleri gibi bu paket de Java programla dili kullanılarak hazırlanmış olup platform bağımsızdır. Pakette yer alan sınıflar çoğunlukla *Java Swing* bileşenlerinin birer uzantısı olarak tasarlanmışlardır.

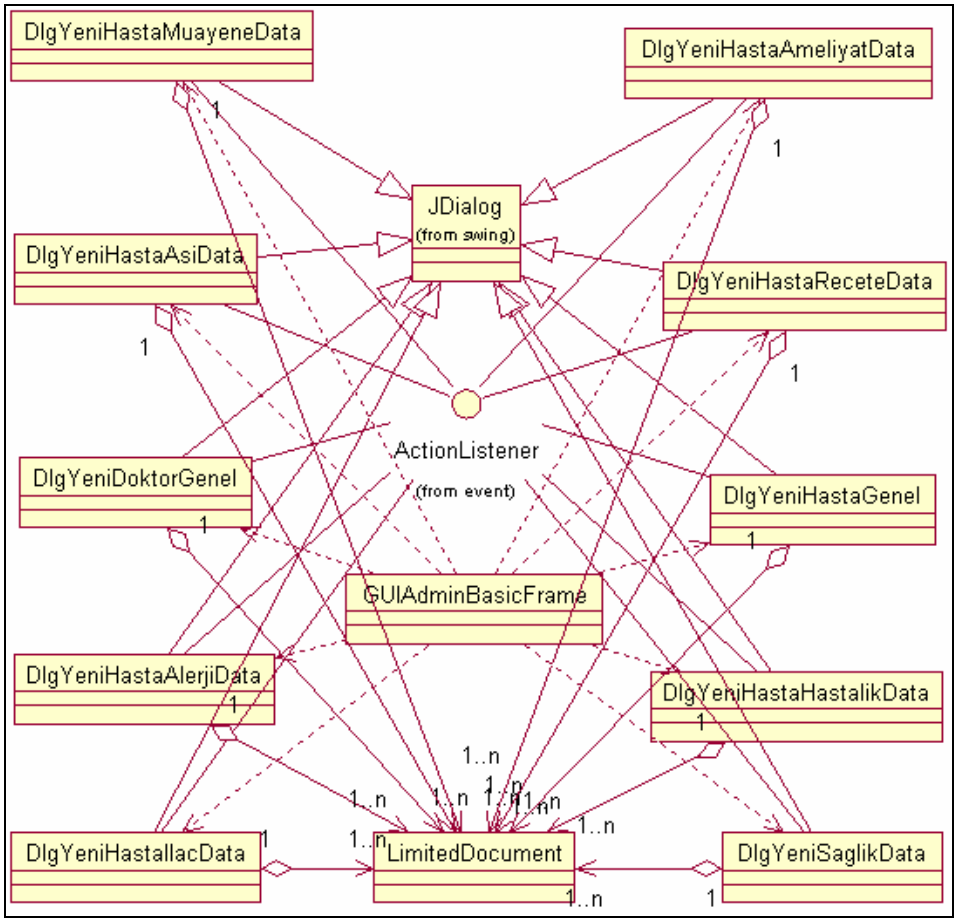
Şekil 6.1, 6.2 ve 6.3'te yazılımın nesne modeli yer almaktadır. Model de uzantı (extend) ve uygulama (implementation) ilişkileri dışında Java Swing bileşenleri ile ilgili olan ilişkiler yer almamıştır. Zaten tez raporunun bu bölümünde vurgulanmak istenen her bir formun ya da diyaloğun içerdiği metin kutusu, komut düğmesi vb. Swing bileşenleri değildir. Özellikle ara yüz yazılımının diğer sistem yazılımları ile nasıl iletişimde bulunduğu ve bu iletişim sonucunda kullanıcı ihtiyaçlarının nasıl karşılandığı hakkında bilgi verilmiştir.



Şekil 6.1 Sistem yönetim yazılımı nesne modeli (1.Kısım)

Yazılım çalıştırıldığına ilk yüklenen nesne `GUIAdminApp` olacaktır. Bu nesne de `GUIAdminBasicFrame` sınıfından bir nesne türeterek görevini tamamlar. Bu aşamadan sonra görsel ara yüz bileşenleri devreye girer.

Görüldüğü gibi modelde `javax.swing.JFrame`'i uzatan tek sınıf `GUIAdminBasicFrame`'dir. Bu sınıftan türetilen nesne, kullanıcı önüne gelecek ana ara yüz bileşenlerini içerir. Bu bileşenlerden biri uygulamaya ait işlemler mönüsüdür. Mönüden her hangi bir istekte bulunulduğunda öncelikle kullanıcı ile etkileşimi sağlayacak ve bu isteği yerine getirecek diyalog penceresi açılır.

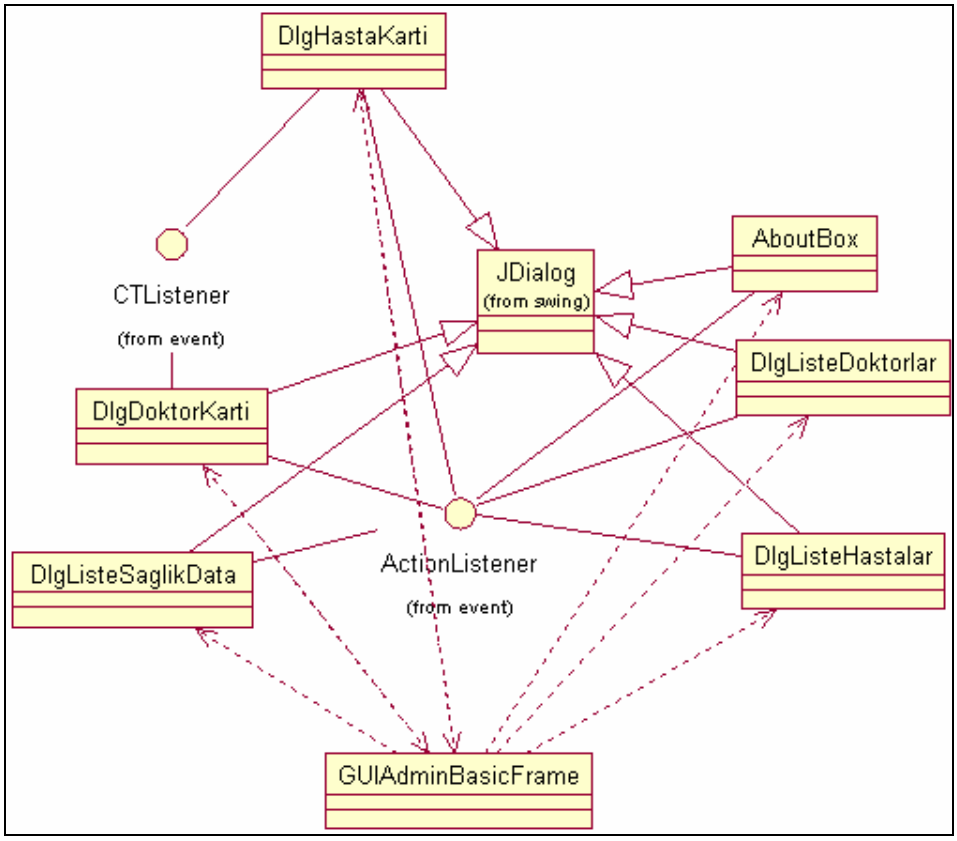


Şekil 6.2 Sistem yönetim yazılımı nesne modeli (2.Kısım)

Sistem modeli incelendiğinde `javax.swing.JDialog`'u uzatan çok sayıda sınıf görülmektedir. Bu sınıflar aynı zamanda kullanıcı etkileşimini sağlama amacıyla `java.awt.event.ActionListener` ara yüzünü uygulamaktadırlar. Adı *"Dlg"* ile başlayan tüm sınıflar için bu özellikler geçerlidir. `DlgHastaKarti` ve `DlgDoktorKarti` sınıfları ise akıllı kart kaynaklı olayları dinleme amacı ile, ek olarak `opencard.core.event.CTListener` ara yüzünü uygularlar.

Yazılım modelinde tüm diyalog pencerelerinin uygulama ana formu görevini üstlenen `GUIAdminBasicFrame` üzerinden açılması gibi

bir tasarım görülmektedir. Kullanıcı işlem isteğini belirtikten sonra bu isteği yerine getirecek olan diyalog nesnesi `GUIAdminBasicFrame` nesnesi tarafından oluşturularak pencere ekrana getirilir.



Şekil 6.3 Sistem yönetim yazılımı nesne modeli (3.Kısım)

Yine model incelendiğinde bir çok diyalog nesnesinin birden fazla `LimitedDocument` sınıfından türetilen nesne ile *toplama* ilişkisinde olduğu görülmektedir. `LimitedDocument` sınıfı `javax.swing.text.DefaultStyledDocument` sınıfının bir alt sınıfı olarak tasarlanmıştır. Ara yüzlerde yer alan ve kullanıcıdan veri alımında kullanılan metin alanlarının formatlanması amacı ile hazırlanmıştır. Örneğin diyaloglarda yer alan `javax.swing.JTextField` bileşenlerinin doküman özelliği olarak bu `LimitedDocument` sınıfından türetilen nesneler atanarak

metin sahasının sadece sayısal veri alması veya belli bir uzunluğu geçmemesi gibi kontroller otomatik olarak gerçekleştirilmektedir.

Ara yüz bileşenleri sistem veritabanına erişiminde ve akıllı kart iletişimlerinde sırası ile `tr.edu.ege.ube.akss.merkezdb` ve `tr.edu.ege.ube.akss.client` sistem paketlerini kullanmaktadır. Her iki paketle ilgili detaylı bilgi önceki bölümlerde verilmiştir. Yönetim yazılımı içerisinde bu paketlerin nasıl kullandığı ise ilerleyen bölümlerde açıklanmıştır.

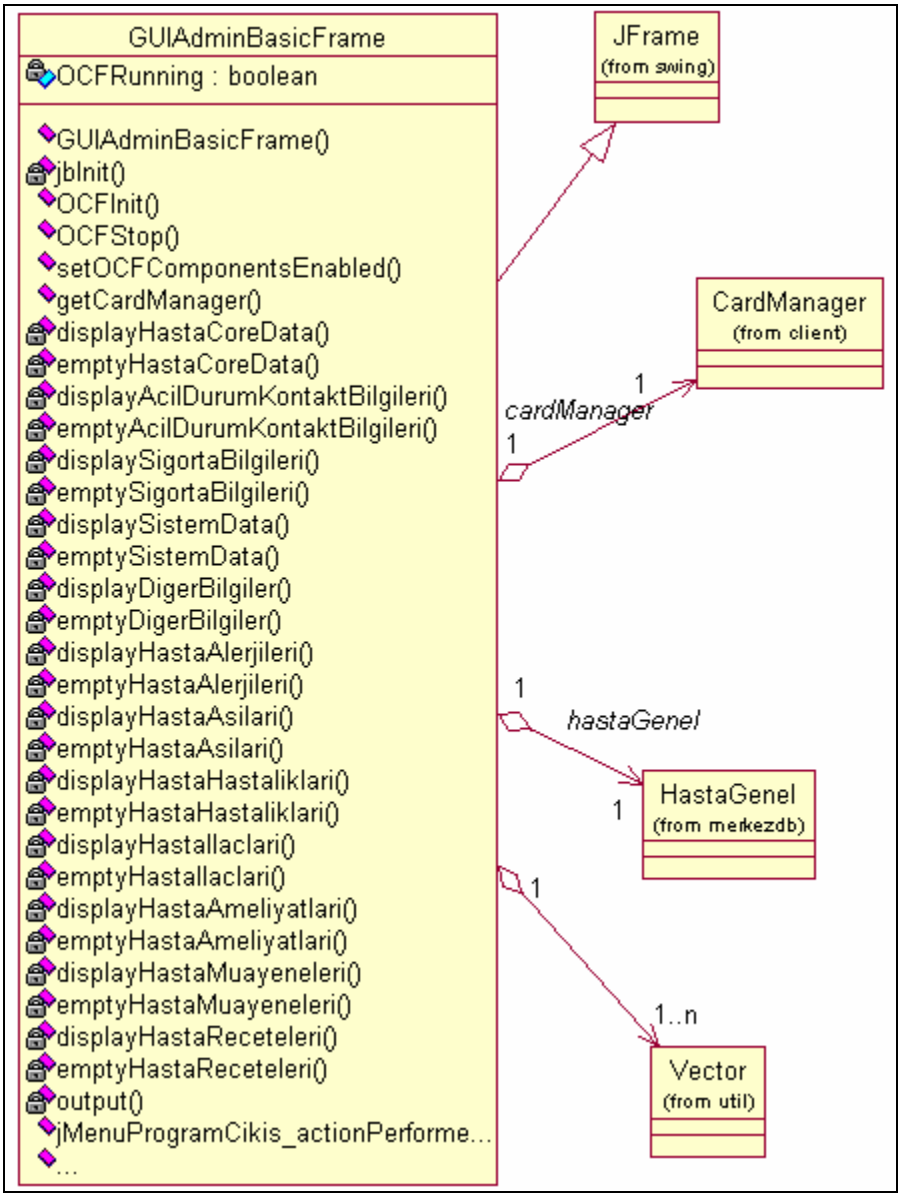
İzleyen bölümlerde yazılım tarafından yerine getirilen sistem yönetim fonksiyonları ve bu fonksiyonların yerine getirilmesinde görev alacak olan yazılım bileşenleri hakkında bilgiler yer almaktadır.

6.1.3 Yazılım ana formu

Yönetim yazılımı çalıştırıldığında ilk yüklenen ara yüz `GUIAdminBasicFrame` sınıfından türetilen nesnedir (Şekil 6.4). Nesne, görüntü bileşenleri dışında akıllı kart ortamını kullanmak amacı ile `tr.edu.ege.ube.akss.client.CardManager` sınıfından türettiği bir nesneyi de özellik olarak barındırmaktadır. `CardManager` nesnesi daha önce de belirtildiği gibi OpenCard Çatısı'nı kullanarak makine üzerinde kart ortamını hazırlamaktadır. Ek 1.1'de uygulamanın yüklenmesinden sonra `GUIAdminBasicFrame` nesnesinin sunduğu ara yüz görülmektedir.

`GUIAdminBasicFrame` nesnesi sunduğu diyaloglar dışında kullanıcılara hasta kayıtları ile ilgili sorgu yapma imkanı da vermektedir. Kullanıcı, Ek 1.2'de görüldüğü gibi hasta tanımlama numarasını girip bilgi isteğinde bulunduğunda nesne `tr.edu.ege.ube.akss.merkezdb` paketinin *tek örnek* yapıdaki `MerkezDB` nesnesi ile iletişime geçerek görüntülenecek hasta bilgilerini almaktadır. `MerkezDB`'den dönen hasta genel bilgileri `GUIAdminBasciFrame`'in `HastaGenel` sınıfından türettiği nesnesinde

saklanmaktadır. Hasta sağlık bilgileri ise `java.util.Vector` nesnesi altında `MerkezDB`'den alınmakta ve dizilerde yer alan nesne özellikleri uygun tablolarda kullanıcıya iletilmektedir.



Şekil 6.4 GUIAdminBasicFrame sınıfı

Tüm veritabanı erişiminde `tr.edu.ege.ube.akss.merkezdb` paketi kullanılmakta bu da `GUIAdminBasicFrame` gibi bir ara yüz nesnesinin sistem veri modeline kontrollü erişimini sağlamakta ve ara yüzü SQL sorgularından soyutlamaktadır. Sistem veritabanı bölümünde anlatıldığı gibi sistem veritabanı istemcisi konumundaki bileşenler sadece `merkezdb` paketinde yer alan `MerkezDB`'nin ilgili metodunu çağırmaktadırlar. Örneğin `GUIAdminBasicFrame`, `MerkezDB.getHasta()` metodunu kullanıcıdan aldığı tanımlama numarası parametresi ile çağırdığında bu tanımlama numarasına sahip hasta genel bilgilerini `HastaGenel` nesnesi altında elde etmektedir. `HastaGenel` nesnesinin özellikleri olarak aldığı bilgileri de istediği formatta Ek 1.2'de gösterildiği gibi kullanıcıya görüntüler. Veritabanı iletişimi sırasında oluşan ve `MerkezDB` tarafından fırlatılan bir hatayı da `GUIAdminBasicFrame` yakalayarak kullanıcıya görüntüler.

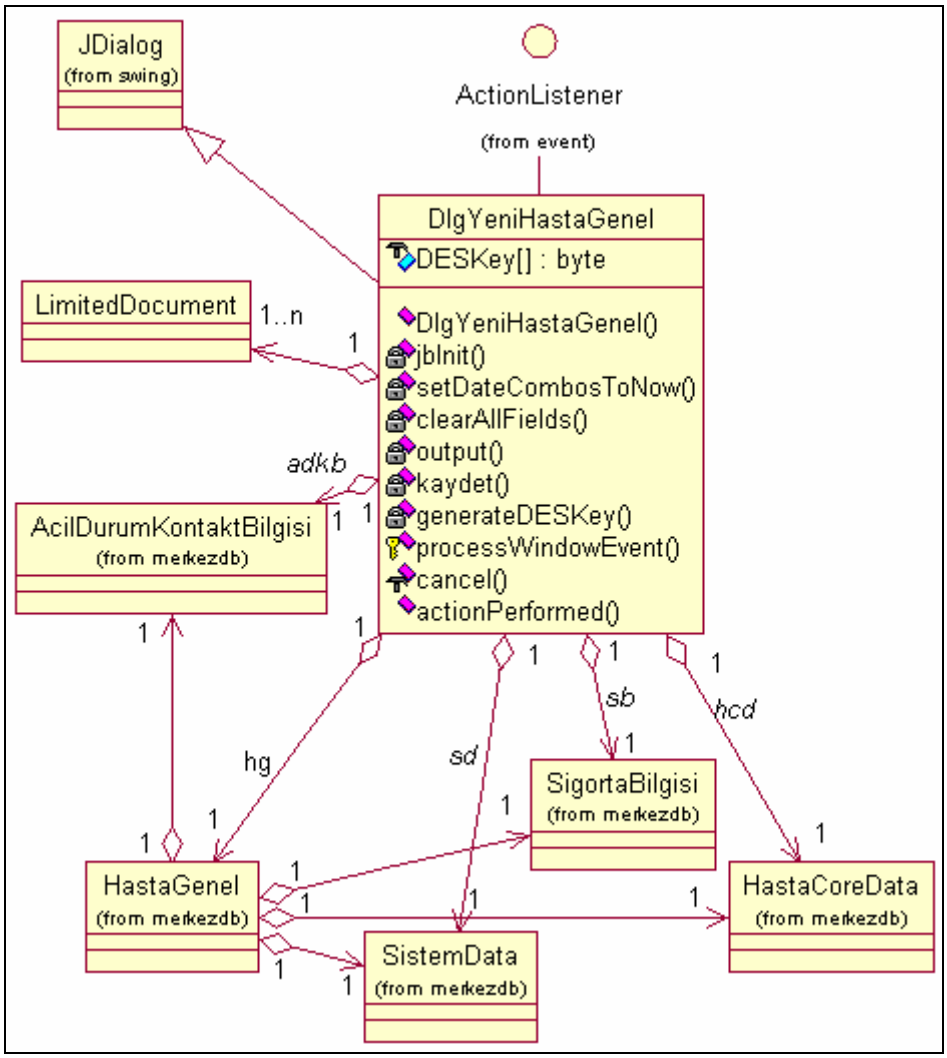
6.1.4 Yeni kayıt işlemleri

Sistem yöneticileri yeni hasta, doktor ve sağlık bilgisi kayıt işlemlerini sistem yönetim yazılımı üzerinden gerçekleştirebilirler. Bu işlemlerde görev yalan yazılım bileşenleri izleyen alt başlıklarda yer almaktadır.

6.1.4.1 Yeni hasta kaydı

Sisteme yeni bir hasta kaydedilmek istendiğinde kullanıcıdan bilgilerin alınması amacıyla yüklenen ara yüz, `DlgYeniHastaGenel` sınıfından (Şekil 6.5) türetilen diyalog nesnesi tarafından sağlanmaktadır.

Ek 1.3'te görülen bir form üreten diyalog nesnesi kullanıcıdan aldığı bilgileri oluşturduğu bir `tr.edu.ege.ube.akss.merkezdb.HastaGenel` nesnesi içerisinde kaydetmek üzere `MerkezDB` nesnesinin ilgili metoduna parametre olarak yollar. Metottan dönen işlem sonucu kullanıcıya görüntülenir (Ek 1.4).



Şekil 6.5 DlgYeniHastaGenel sınıfı

Hasta DES anahtarının üretilmesi:

Kaydedilmek üzere giden bilgiler arasında hastaya ait DES anahtarı da bulunmaktadır. Bu anahtar, `MerkezDB`'ye ait metot çağırımı

yapılmadan hemen önce oluşturulmaktadır: DES anahtarı üretecek olan `javax.crypto.KeyGenerator` nesnesi 56 bitlik DES anahtarı oluşturacak şekilde hazırlandıktan sonra `java.security.Key` nesnesi üretilir. Hazırlanan nesne bayt dizisine dönüştürülerek veritabanında özellikleri saklanacak olan `HastaGenel` nesnesinin DES anahtarı özelliği olarak atanmaktadır.

6.1.4.2 Yeni hasta sağlık bilgisi kaydı

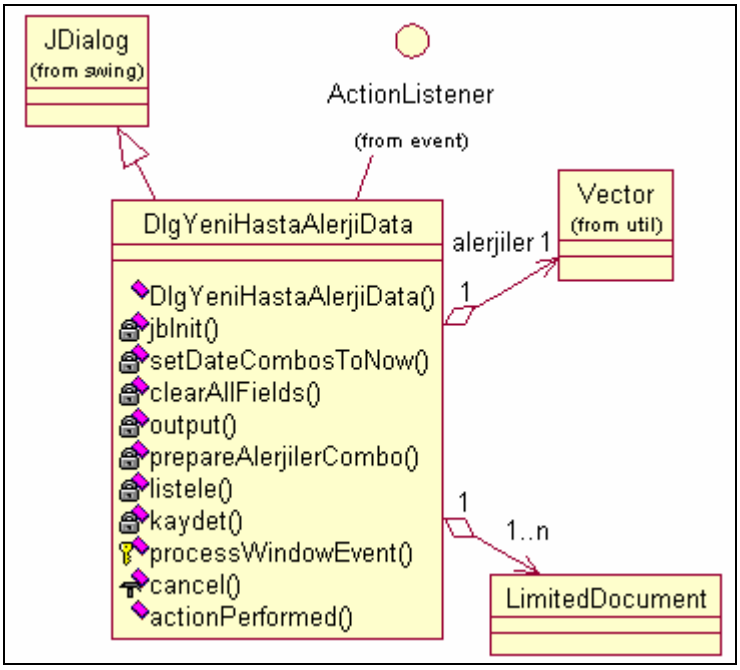
Sistemde kayıtlı olan bir hastaya ait yeni alerji, aşı, vb. sağlık bilgisi kaydı uygun diyalog pencereleri aracılığı ile gerçekleştirilir. Örnek olarak bir hastaya ait yeni bir alerji bilgisinin kaydedilmesi hakkında bilgi verilecektir. Diğer bilgilerin de kaydını sağlayan diyalog nesneleri benzer biçimde tasarlanmış ve kullanılmıştır.

Yeni bir hasta alerji kaydı gerçekleştirilmek istendiğinde, Şekil 6.6'da yer alan `DlgYeniHastaAlerji` sınıfından türetilen nesne, veri giriş ara yüzünü hazırlar. Veritabanında kayıtlı olan alerji sağlık bilgisi kayıtları `SaglikDataBean` nesneleri olarak alınır ve diyalogda yer alan birleşik giriş kutusu (combo box) bileşeninde listelenir. Kullanıcının listeden alerji bilgisini seçmesi, hasta tanımlama numarası ve bulgu tarihi bilgilerini girerek kayıt düğmesine basması sonucu `HastaAlerjiData` nesnesi oluşturularak girilen bilgiler bu nesneye özellik olarak aktarılır. Hazırlanan nesne de veritabanına kaydedilmek üzere `MerkezDB`'ye gönderilir. Ek 1.5'te `DlgYeniHastaAlerji` diyalog nesnesinin hazırladığı ara yüz görülmektedir.

Ara yüzde ayrıca ilgili hastaya ait tüm alerji kayıtları da listelenmektedir: Veritabanından `java.util.Vector` nesnesi içerisinde alınan `HastaAlerjiData` nesnelerine ait özellikler, ara yüzde yer alan tabloda kullanıcıya listelenmektedir.

Hastalara ait yeni ameliyat, aşı, hastalık, ilaç, muayene ve reçete kayıtları da benzer şekilde tasarlanan sırası ile `DlgYeniHastaAsi`,

`DlgYeniHastaAmeliyat`, `DlgYeniHastaHastalik`, `DlgYeni HastaIlac`, `DlgYeniHastaMuayene`, `DlgYeniHastaRecete` diyalog nesnelerinin sunduğu ara yüzler aracılığı ile gerçekleştirilmektedir.

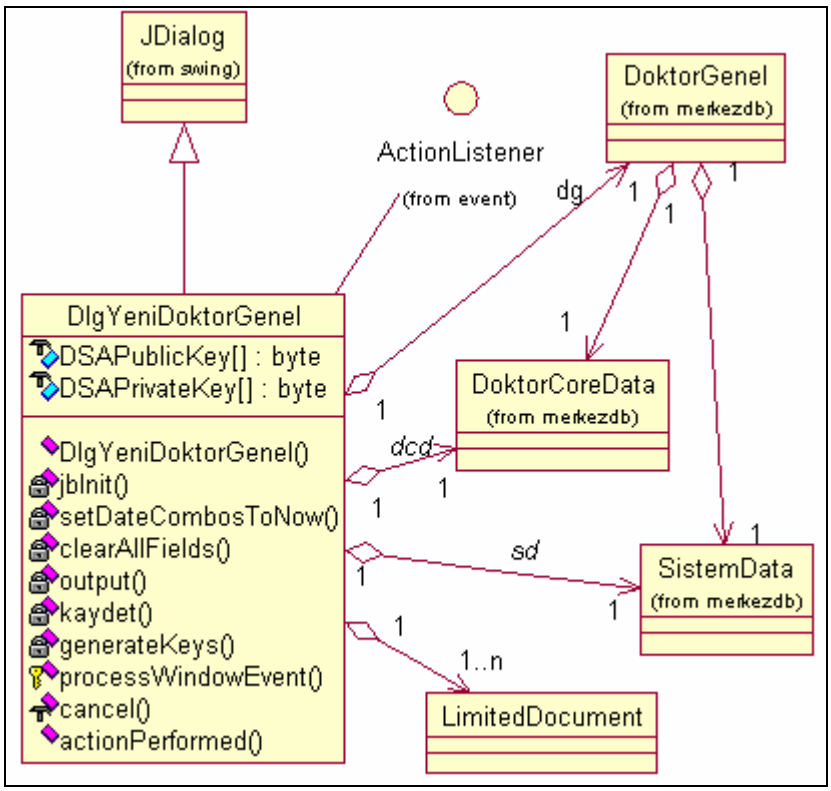


Şekil 6.6 DlgYeniHastaAlerjiData sınıfı

6.1.4.3 Yeni doktor kaydı

Sisteme yeni bir doktor kaydedilmek istendiğinde kullanıcıdan bilgilerin alınması amacıyla yüklenen ara yüz, `DlgYeniDoktorGenel` sınıfından (Şekil 6.7) türetilen diyalog nesnesi tarafından sağlanmaktadır.

Ek 1.6'da görülen bir form üreten diyalog nesnesi kullanıcıdan aldığı bilgileri oluşturduğu bir `tr.edu.ege.ube.akss.merkezdb.DoktorGenel` nesnesi içerisinde kaydetmek üzere `MerkezDB` nesnesinin ilgili metoduna parametre olarak yollar. Metottan dönen işlem sonucu kullanıcıya görüntülenir.



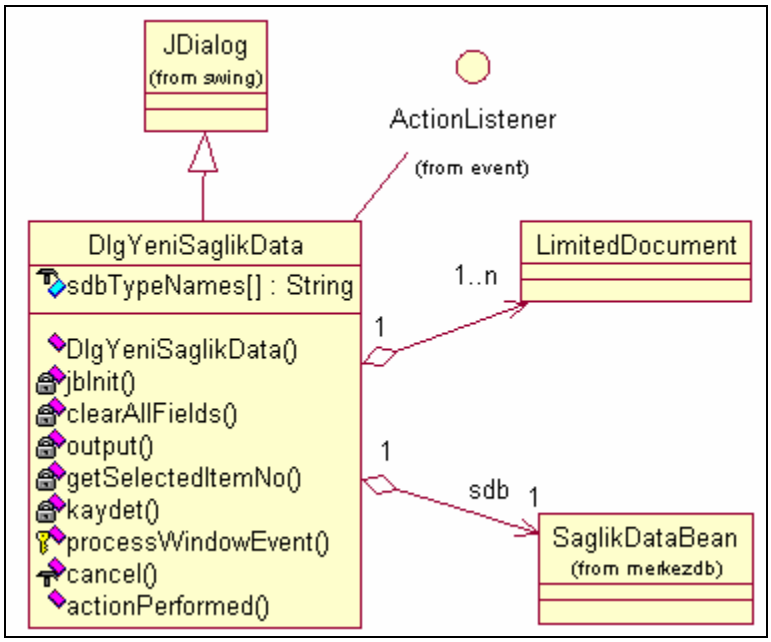
Şekil 6.7 DlgYeniDoktorGenel sınıfı

Doktor DSA anahtarlarının üretilmesi:

Kaydedilmek üzere giden bilgiler arasında doktora ait DSA genel ve özel anahtarları da bulunmaktadır.Bu anahtarların üretilmesi şu şekilde gerçekleşir: DSA anahtarlarını üretecek olan `java.crypto.KeyPairGenerator` nesnesi 512 bit uzunluğa sahip DSA anahtar çiftini oluşturacak şekilde hazırlandıktan sonra bu nesne aracılığı ile `java.security.KeyPair` nesnesi üretilir. `KeyPair` nesnesi biri genel (public) biri de özel (private) olmak üzere iki anahtar nesnesi barındırır. Bu nesneler `java.security.Key` ara yüzü ile alınarak bayt dizinlerine çevrilmekte ve veritabanında özellikleri saklanacak olan `DoktorGenel` nesnesinin DSA genel ve özel anahtarı özellikleri olarak atanmaktadır.

6.1.4.4 Yeni sağlık bilgisi kaydı

Alerji, aşı, vb. sağlık bilgilerinin sisteme kaydedilmesinde Şekil 6.8'de gösterilen `DlgYeniSaglikData` sınıfından türetilen diyaloğa ait kullanıcı ara yüzünden yararlanılmaktadır.



Şekil 6.8 DlgYeniSaglikData sınıfı

Ek 1.7'de görüldüğü gibi kullanıcıdan sağlık bilgisi tipi, tanımlama numarası, ad ve açıklama bilgileri alınmaktadır. Bu bilgiler oluşturulan `tr.edu.ege.ube.akss.merkezdb. SaglikDataBean` nesnesine özellik olarak aktarıldıktan sonra içeriği veritabanına kaydedilmek üzere `MerkezDB` nesnesine gönderilir.

6.1.5 Bilgi güncelleme ve silme işlemleri

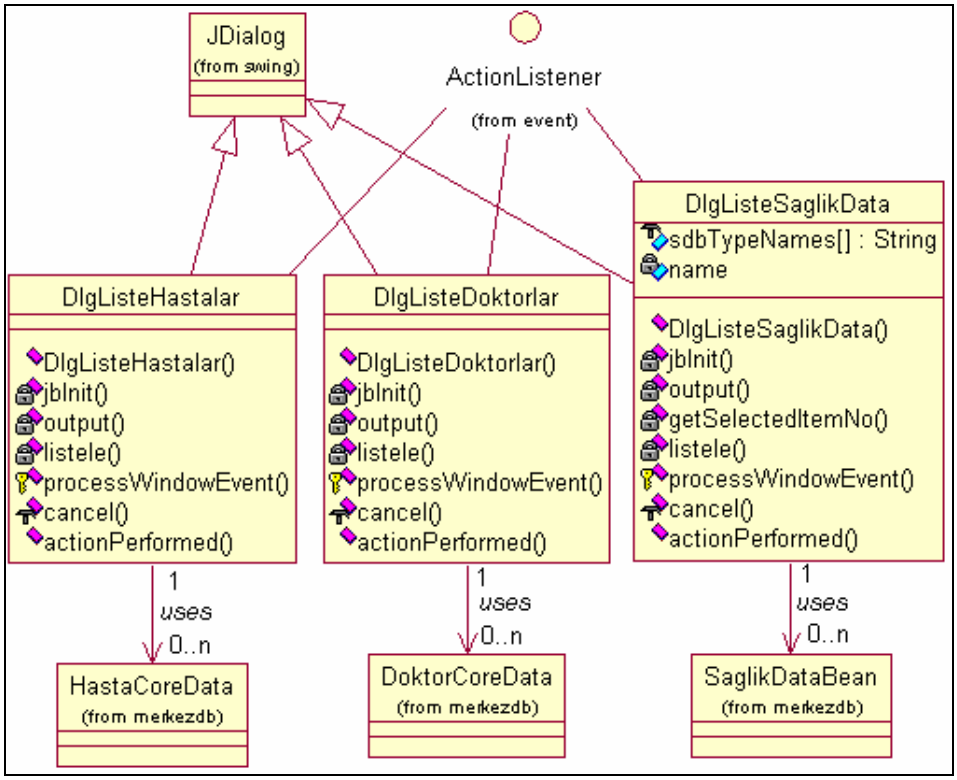
Sistem yöneticileri hasta, doktor ve sağlık bilgilerine ait kayıtları yine sistem yönetim yazılımı üzerinden güncelleme ve silme imkanına sahiptirler.

Bu işlemlerde görev yalan ara yüz bileşenlerinin tasarımı ve diğer paketlerde yer alan nesneler ile olan iletişimleri, yeni kayıt işleminde görev alan bileşenler ile oldukça benzerlik göstermektedir. Örneğin `DlgGuncellemeHastaGenel` diyalog sınıfının tasarımı ve nesne ilişkileri Bölüm 6.1.4.1'de açıklanan `DlgYeniHastaGenel` sınıfı ile güncelleme ve silmeye ait görsel ara yüz bileşenleri hariç aynıdır. Aynı form yapısında bilgileri güncellenecek hastaya ait tanımlama numarası kullanıcıdan alınır. Bu tanımlama numarasına sahip hasta genel bilgileri `tr.edu.ege.ube.akss.merkezdb. HastaGenel` nesnesi içerisinde `MerkezDB`'den alınır. Nesne, ilgili özellikleri değiştirildikten sonra `MerkezDB`'ye veritabanında güncellenmek üzere geri yollanır. Kullanıcı hasta kaydını sistemden silmek istediğinde ise hasta numarası `MerkezDB`'ye gönderilir. Bu durumda hastaya ait sistemdeki tüm bilgiler (sağlık bilgileri de dahil) silinecektir.

Bilgi güncelleme ara yüzlerine örnek olarak herhangi bir hastaya ait alerji kayıtlarının güncellenmesinde ve silinmesinde kullanılan `DlgGuncellemeHastaAlerjiData` nesnesinin sunduğu ara yüz Ek 1.8'de verilmiştir. Görüldüğü gibi hastaya ait alerji bilgileri listelenmekte, listeden seçilen herhangi bir kayıt da güncellenebilmekte veya silinebilmektedir.

6.1.6 Listeleme işlemleri

Sistemde kayıtlı tüm hastalara, doktorlara ve sağlık kayıtlarına ait listeleri kullanıcılara, sırası ile `DlgListeHastalar`, `DlgListeDoktorlar` ve `DlgListeSaglikData` diyalog nesneleri sunmaktadır (Şekil 6.9).



Şekil 6.9 DlgListeHastalar, DlgListeDoktorlar ve DlgListeSaglikData sınıfları

Ek 1.9'da, `DlgListeHastalar` diyalog nesnesinin kullanıcılara sunduğu ara yüz görülmektedir. Nesne tüm hastalara ait çekirdek bilgileri `MerkezDB`'den `tr.edu.ege.ube.akss.merkezdb.HastaCoreData` nesneleri dizisi şeklinde almaktadır. Her bir `HastaCoreData` nesnesi sistemde kayıtlı bulunan bir hastaya ait çekirdek bilgileri içermektedir. `DlgListeHastalar` bu `HastaCoreData` nesnelerinin özelliklerine erişir ve bunları içerdiği bir tablo bileşeninde kullanıcılara sunar.

`DlgListeDoktorlar` nesnesi de aynı prosedürü `MerkezDb`'den elde ettiği `tr.edu.ege.ube.akss.merkezdb.DoktorCoreData` nesnelerini kullanarak gerçekleştirir. Sunduğu ara yüz Ek 1.10'da verilmiştir.

DlgListeSaglikData diyalog nesnesi ise sistemde oluşturulurken listeleme yapmaz. Öncelikle kullanıcıdan, listelenmesi istenecek sağlık bilgisi tipinin seçilmesini bekler. Seçilen tipe göre MerkezDb'den o tipteki sağlık bilgilerini temsil eden tr.edu.ege.ube.akss.merkezdb.SaglikDataBean nesneleri alınır ve bu nesne özellikleri görüntülenir. (Ek 1.11)

6.1.7 Akıllı kart işlemleri

Sistem yönetim yazılımı içerisinde hasta ve doktor kartları ile ilgili her türlü işlem ana form üzerinden açılan iki ara yüz ile gerçekleştirilir. Bu ara yüzleri kullanıcıya sunan DlgHastaKarti ve DlgDoktorKarti ara yüz bileşenleridir. Aşağıdaki alt başlıklarda hasta ve doktor kartları üzerinde hangi işlemlerin gerçekleştirildiği ve bu işlemler gerçekleştirilirken görev alan bileşenler hakkında bilgiler yer almaktadır.

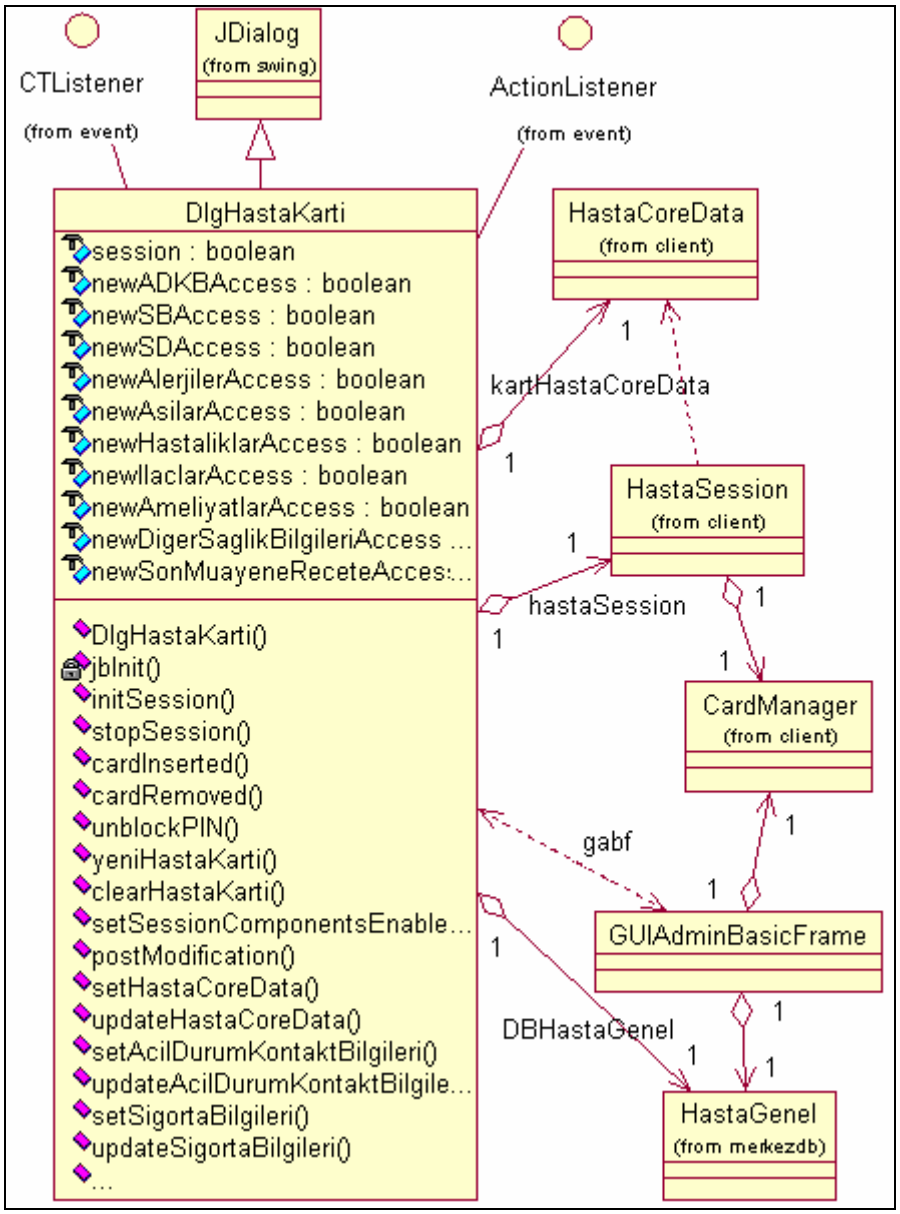
6.1.7.1 Hasta akıllı kartı işlemleri

Hasta akıllı kartları üzerinde işlem yapılmasını sağlayan ara yüzü kullanıcılara sunan DlgHastaKarti sınıfından türetilen nesnedir (Şekil 6.10). Nesne istekleri yerine getirirken hem sistem akıllı kartları hem de veritabanı ile iletişim halindedir.

DlgHastaKarti bir ara yüz bileşeni olarak, javax.swing.JDialog sınıfını uzatmakta; kullanıcı kaynaklı ara yüz olaylarını algılamak için de java.awt.ActionListener ara yüzünü uygulamaktadır. Ek 1.12'de nesnenin kullanıcılar önüne sunduğu grafik ara yüzü görülmektedir.

DlgHastaKarti nesnesi, üzerinde çalıştığı makine üzerinde gerçekleşen akıllı kart kaynaklı olayları algılamak için opencard.core.event.CTListener ara yüzünü uygular. Daha önce de belirtildiği gibi bu ara yüzü uygulayan bileşenlerin kendilerini makine

üzerine yer alan OpenCard platformundaki olay üreticisine (event generator) kart terminali dinleyicisi olarak kaydetmeleri gerekmektedir.



Şekil 6.10 DlgHastaKarti sınıfı

DlgHastaKarti nesnesi de sistemde ilk oluşturulduğunda kendini bu olay üreticisine kayıtlar. Böylelikle bundan sonra oluşacak akıllı karta dayalı olaylardan haberdar olacaktır. Nesne işlevi sona erdiğinde de üreticiden kaydını silmektedir. CTListener ara yüzü hakkında detaylı bilgi sistem altyapı çalışmalarında verilmiştir.

CTListener ara yüzü uygulayıcısı olarak DlgHastaKarti, opencard.core.event.CardTerminalEvent nesnesini parametre alan cardInserted() ve cardRemoved() metotlarını barındırmaktadır. Böylece kart okuyucuya akıllı kart yerleştirilmesi ve kartın çıkarılması olayları kontrol edilir. Bu tasarım sayesinde özellikle akıllı kart oturumların açılması ve kontrolü nesne tarafından kolayca gerçekleştirilmektedir.

DlgHastaKarti, sağlık sisteminde kullanılan akıllı kartlara özgü kart servislerini kullanmak amacıyla kendisini oluşturan GUIAdminBasicFrame'in tr.edu.ege.ube.akss.client.CardManager nesne özelliğine referans elde etmektedir. Hatırlanacağı gibi sistem yönetim yazılımının ana formu olan GUIAdminBasciFrame nesnesi ilk oluştuğunda akıllı kart ortamını başlatmakta ve kart servislerinde kullanılacak olan CardManager nesnesini hazırlamaktadır. DlgHastaKarti nesnesi de yine ilk oluşturulduğunda elde ettiği bu CardManager nesnesini kullanarak hasta kartı oturumlarında kullanacağı tr.edu.ege.ube.akss.client.HastaSession nesnesini hazırlar. CardManager ve HastaSession nesneleri kart istemci ara yazılımı paketinde yer almakta olup haklarında detaylı bilgi bu yazılımın anlatıldığı bölümde verilmiştir.

Kart okuyucuya hasta akıllı kartı yerleştirildiğinde makine üzerine yer alan akıllı kart platformu olay dinleyici DlgHastaKarti nesnesine haber vermekte; nesne de bu olayı değerlendirerek kart oturumunu başlatmaktadır. Oturum başlangıcında CardManager nesnesi vasıtasıyla güvenli iletişim kanalı kurulur, akıllı kart üzerindeki hasta uygulaması seçilir ve bu uygulama ile iletişime geçilir. Hastaya ait genel bilgiler ve acil durum iletişim bilgileri otomatik olarak karttan alınır ve kullanıcıya

görüntülenir. Kart iletişiminde `HastaSession` nesnesi görev almakta; bu da `DlgHastaKarti` ara yüz bileşenini APDU iletişiminden soyutlamaktadır. `DlgHastaKarti` nesnesinin görevi sadece kendisine gelen bilgileri uygun yapıda ekranda görüntülemektir.

Hasta kartı oturumunun açılması sırasında `DlgHastaKarti` nesnesi sistem veritabanı ile de iletişime geçerek kart sahibinin veritabanındaki genel bilgilerini `tr.edu.ege.ube.akss.merkezdb.HastaGenel` nesnesi içerisinde alır ve kendisinde bu nesneyi özellik olarak barındırır.

Hasta oturumunun asıl anlamda açılabilmesi için kart üzerine kullanıcı yetkilendirilmesinin yapılması gerekmektedir. Ek 1.13'te verilen ekran görüntüsünde kart okuyucusuna hasta akıllı kartının yerleştirilmesi sonucu oturumun açılması sırasında kullanıcıdan PIN girişi istenmektedir. Burada yazılımı kullanan yönetici şifreyi kendisi girebilir ya da kart bir hastaya ait ise bu hastaya ait şifreyi veritabanından da isteyebilir. Şifrenin akıllı kart tarafından onaylanması sonucu hasta kartı oturumu açılmış olur.

Kart üzerindeki bilgilere, alan adlarına göre isimlendirilmiş sekmeli panellerden erişilebilir. Her bölümde genel olarak iki işlem söz konusudur: Bilgilerin karttan yüklenerek ekrana getirilmesi ve veritabanından karta bilgi güncellemesi. Sekmelerde diğerlerine göre farklı özelliğe sahip olan ise *"Son Muayene / Reçete"* sekmesidir. Bu bölümde kart üzerindeki muayene ve reçete bilgilerinin ekrana getirilmesi dışında kartta yer alan muayene ve reçete bilgilerinin veritabanına kaydedilmesi işlemi yer almaktadır. Böylelikle yönetici, hasta kartına doktor tarafından yazılan bilgileri sisteme kaydetmekte ve hasta reçetesine onay vermektedir. Seçimlik olarak son muayene ve reçete bilgilerinin veritabanından karta güncellemesi de mümkündür. Ek 1.14'te bu sekmenin yer aldığı ekran görüntüsü verilmiştir.

Sistem yöneticileri aynı zamanda yeni kart hazırlama, kart iptali ve bloke kartın açılması işlemlerini de yine `DlgHastaKarti` nesnesinin sunmuş olduğu ara yüz üzerinden gerçekleştirebilirler.

Yeni bir hasta kartının çıkarılması için ön şart kart çıkarılacak hastanın sistemde kayıtlı olması ve hazırlanacak akıllı kartın başka bir hastaya ait olmaması yani hiçbir hasta bilgisi içermemesidir. Yeni kart okuyucuya yerleştirilip oturum açıldığında ara yüz üzerinden yönetici yeni kart çıkarma isteğini belirtir. Kartın uygun olup olmadığı kontrol edildikten sonra bilgileri yüklenecek olan hastanın sistem geneli numarası yöneticiden istenir. Yönetici numarayı belirtip onayladıktan sonra ilgili hastaya ait, akıllı kart üzerinde bulunması gereken bilgiler veritabanından alınarak karta aktarılır.

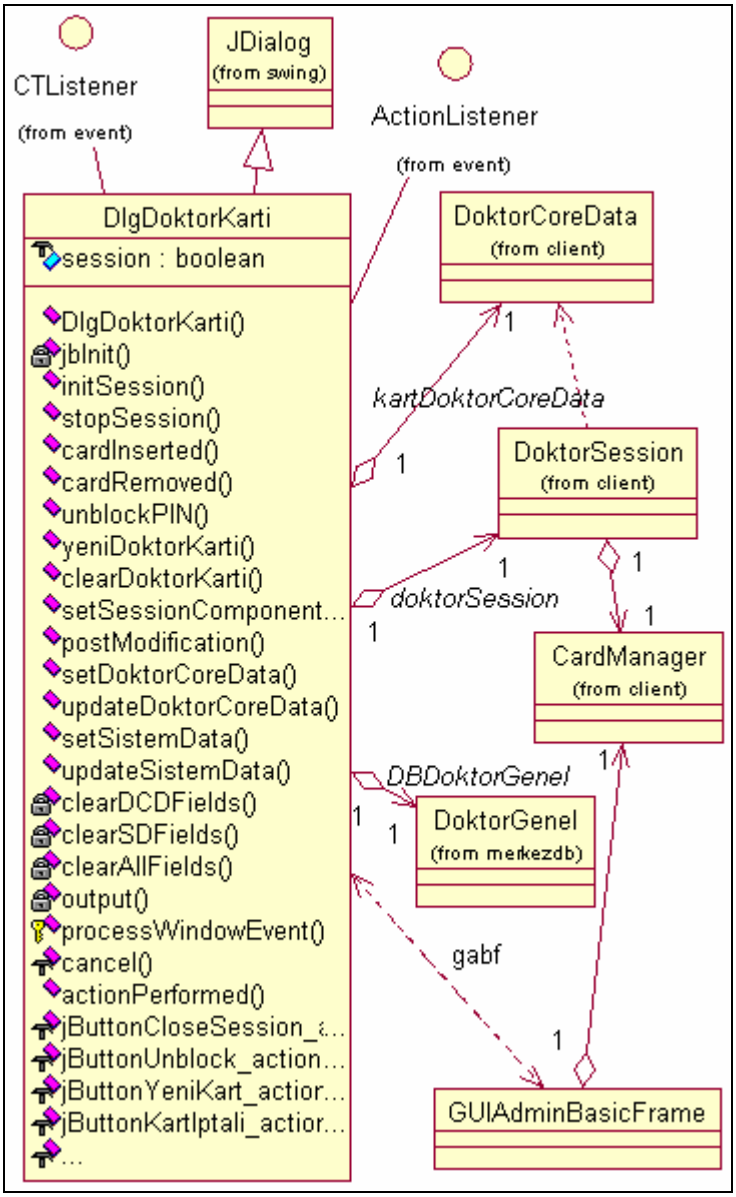
Hasta akıllı kartının iptali durumunda kart üzerindeki tüm hasta bilgileri silinmektedir. Üzerinde bilgi yer almayan kartlar yeni hastalara atanabilmektedir.

Yanlış şifre girişleri sonunda bloke olan kartların açılması için ya yeni bir şifre girilir ya da hasta sistem kayıtlarında yer alan şifre yeniden kart şifresi olarak atanır.

Hasta akıllı kartı üzerinde istenen işlemler yerine getirildikten sonra ara yüzden oturum sona erdirilir. Kartın herhangi bir zamanda okuyucudan çıkarılması da yine oturumun otomatik olarak sonlanması sonucunu doğurur. Oturum sonlandıktan sonra ara yüz bileşeni yeni bir hasta kartını beklemeye geçer.

6.1.7.2 Doktor akıllı kartı işlemleri

Doktor akıllı kartları üzerinde işlem yapılmasını sağlayan ara yüzü, kullanıcılara `DlgDoktorKarti` sınıfından türetilen nesne (Şekil 6.11) sunmaktadır.



Şekil 6.11 DlgDoktorKarti sınıfı

`DlgDoktorKarti` sınıfının tasarımı ve sistemde üstlendiği rol `DlgHastaKarti` ile aynıdır. Bu nedenle içerdiği olay dinleme ve iletişim mekanizmalarına burada değinilmemiştir.

`DlgDoktorKarti` bileşeni, akıllı kart iletişimlerini, barındırdığı `tr.edu.ege.ube.akss.client.DoktorSession` nesnesi aracılığı ile gerçekleştirmektedir. Bu nesnenin işlevi `HastaSession` nesnesi ile aynıdır. Farkı hasta akıllı kartları yerine doktor akıllı kartları ile iletişimde bulunmak için gereken hizmetleri sunmasıdır. Bu sınıf ile ilgili detaylı bilgi kart istemci ara yazılımının anlatıldığı bölümde verilmiştir.

Oturumu açılan kart sahibi doktorun bilgileri, `tr.edu.ege.ube.akss. merkezdb.DoktorGenel` nesnesi içinde veritabanından alınır ve ilgili işlemlerde kullanılır.

`DlgDoktorKarti` nesnesinin sunduğu grafik ara yüz ile, kart sahibi doktorun genel bilgileri ve sistem bilgilerinin veritabanındaki bilgiler ile güncellenmesi, kartın iptali ve yeni doktor kartının çıkarılması işlemleri mümkündür. Ek 1.15'te kullanıcılara sunulan ara yüz görülmektedir.

6.2 Klinik Yazılımı

Bu bölümde muayenehanelerde yer alan ve kullanıcılar ile etkileşimi sağlayan sistem yazılım bileşenleri hakkında bilgi verilmektedir. İlk olarak yazılımın genel tanıtımı ve tasarımı anlatılmıştır. Daha sonra yazılımın yerine getirdiği fonksiyonlar ve diğer sistem bileşenleri ile olan iletişimi ele alınmıştır.

6.2.1 Yazılımın genel tanıtımı

Klinik yazılımı, Akıllı Kart Sağlık Sistemi kapsamında doktorların muayenehanelerinden sistem ile ilgili gerekli işlemleri yerine getirmelerini sağlayacak kullanıcı ara yüzlerini içeren sistem yazılım paketidir.

Doktorlar, hastalara ait akıllı kartlar üzerindeki hasta bilgilerine bu yazılım aracılığı ile erişmekte ve muayene işlemlerini gerçekleştirmektedirler. İşlemler sırasında gereken klinik bazlı hasta bilgilerine güvenli ve yetkilendirilmiş olarak erişim de bu yazılım aracılığı ile sağlanır. Muayene sonrası akıllı karta muayene ve reçete bilgilerinin yazılması ve klinik veritabanında hasta bilgilerinin güncellenmesi ile ilgili kullanıcı ara yüzleri de yine bu yazılım içerisindedir.

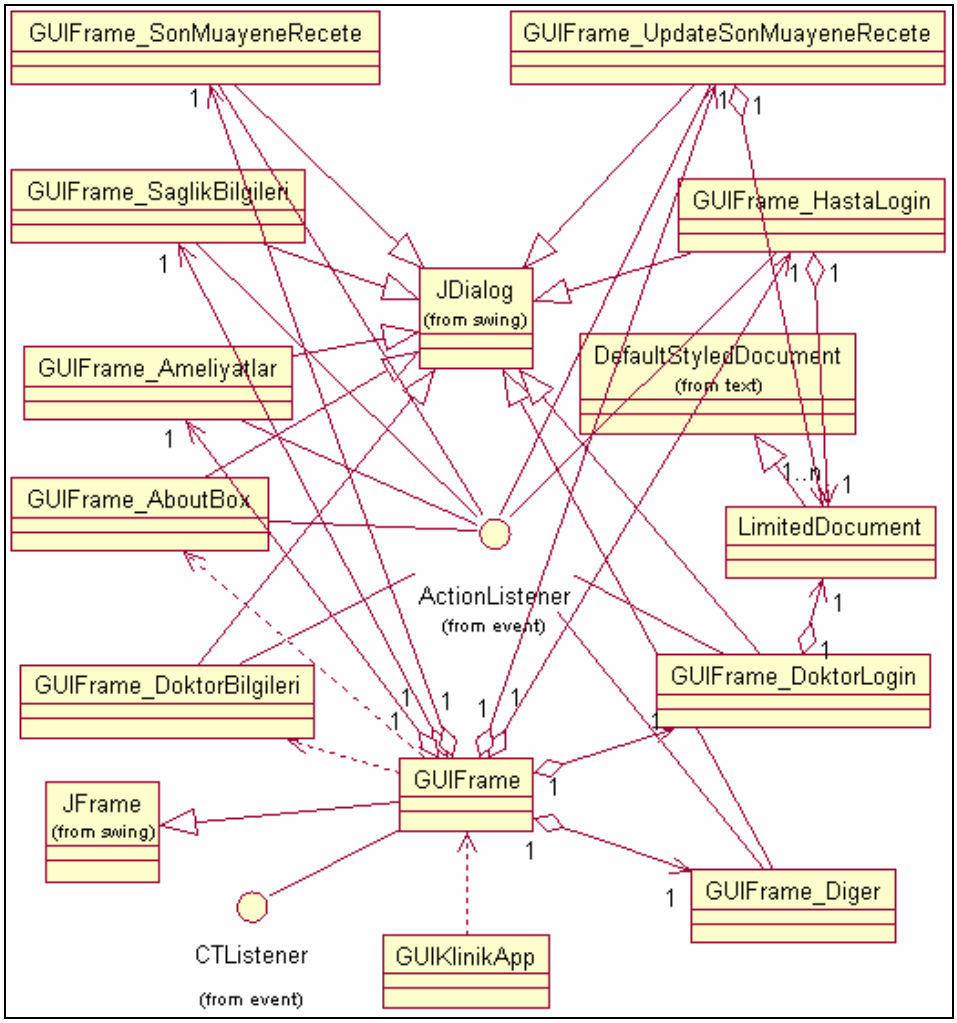
Yazılımın kullanılabilmesi için öncelikle bir doktor oturumunun açılması şarttır. Doktorlar kendilerine ait akıllı kartlar aracılığı ile doktor oturumunu başlatırlar. Bu arada gerekli olan merkez veritabanı bağlantısı gerçekleştirilir. Oturumun başarıyla açılması sonrası hasta kabullerine geçilebilir.

Her hasta kabulünde yazılım üzerinde yeni bir hasta oturumu açılır. Oturum ancak hasta akıllı kartları ile mümkündür. Oturum süresince doktor hasta akıllı kartından istediği bilgilere ulaşabilmektedir. Kliniğe özel bilgilere de yine akıllı kart aracılığı ile ulaşılmaktadır. Muayene sonrası hasta kartına ve veritabanına güncelleme işlemleri yerine getirilmekte ve hasta oturumu kapatılarak yeni bir hasta beklenmektedir.

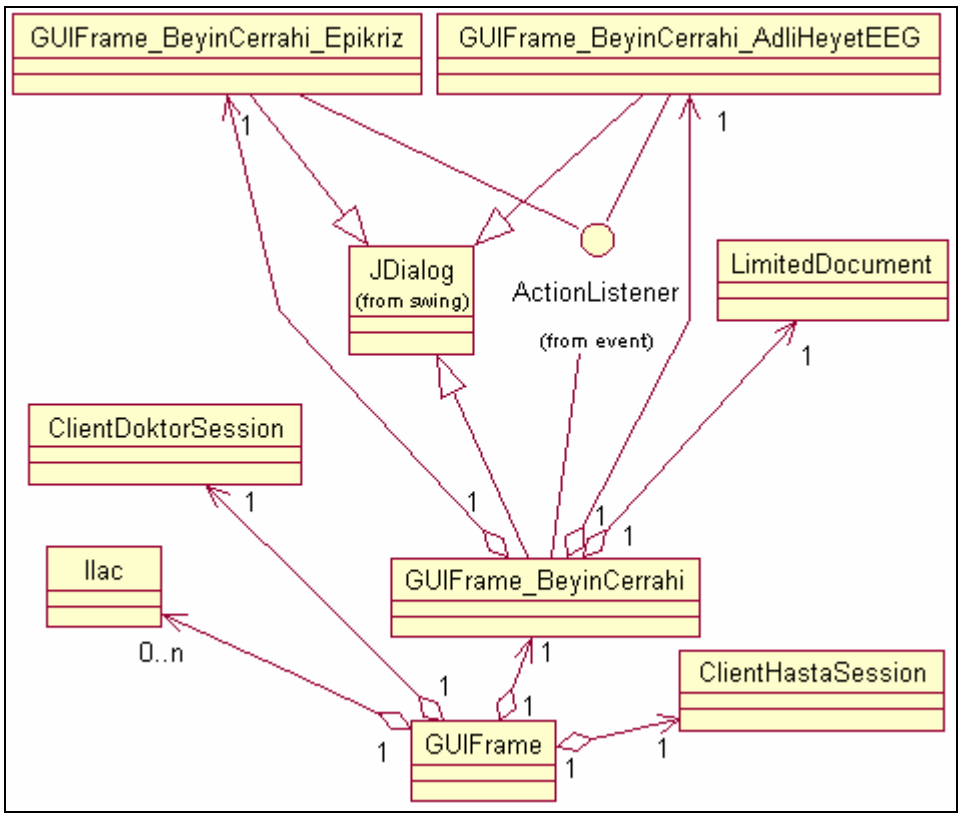
Uygulama, EÜ Beyin Cerrahi A.B.D.'nda çalışmak üzere gereken örnek bileşenleri içermektedir. Ancak yazılımın bölüme ait veritabanı ile iletişimi ve sunduğu ara yüz modülü hariç diğer bileşenleri hastanenin başka bölümlerinde de kullanılabilecek şekilde hazırlanmıştır. Yazılımın başka bir bölümde kullanılabilmesi için ilgili bölüme ait veritabanı ile iletişime geçilecek bileşenlerin ve kullanıcı ara yüzünün yer alması yeterlidir. Bu bileşenlerin de tasarımı Beyin Cerrahi Bölümü'ne benzer bir şekilde olacaktır.

6.2.2 Yazılım mimarisi

Klinik yazılımı, bir önceki bölümde detayları verilen sistem yönetim yazılımı ile birlikte sistem MVC mimarisinin *görünüm* katmanında yer almaktadır. İş mantığı değil de kullanıcı ile etkileşimi sağlayacak olan görsel kullanıcı ara yüzü bileşenlerini içermektedir.



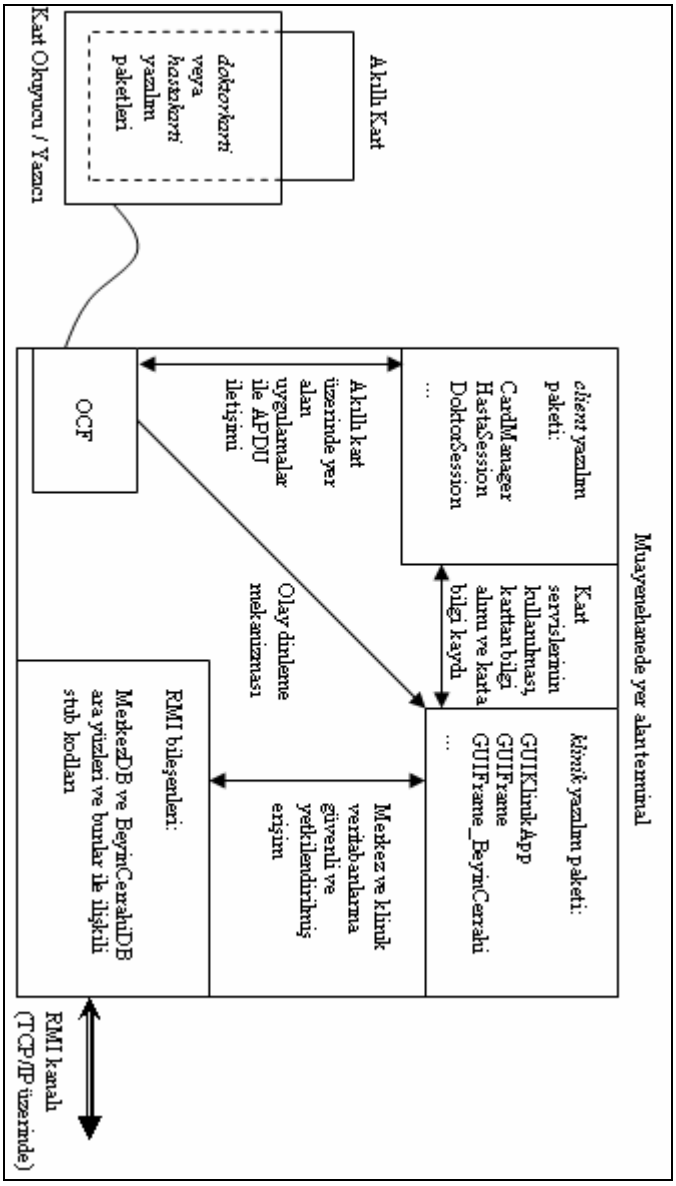
Şekil 6.12 Klinik yazılımı nesne modeli (1.Kısım)



Şekil 6.13 Klinik yazılımı nesne modeli (2.Kısım)

Yazılım paketi `tr.edu.ege.ube.akss.gui.klinik` olarak adlandırılmıştır. Şekil 6.12 ve 6.13'te yazılımın nesne modeli yer almaktadır. Model de uzantı (extend) ve uygulama (implementation) ilişkileri dışında Java Swing bileşenleri ile ilgili olan ilişkiler gösterilmemiştir.

Uygulama çalıştırıldığında `GUIKlinikApp` sınıfı tarafından bir `GUIFrame` nesnesi oluşturulur. `javax.swing.JFrame` sınıfının bir uzantısı olarak tasarlanan `GUIFrame` sınıfından türetilen nesne ekrana getirilecek olan ana form bileşenlerini içermektedir. İçerdiği mönü aracılığı ile kullanıcılar isteklerini yerine getirecek olan diyaloglara erişirler.



Şekil 6.14 Klinik ara yüz bileşenlerinin diğer sistem yazılımları ile iletişimi

`GUIFrame`, `javax.swing.JDialog`'u uzatan ve `java.awt.event.Action Listener` ara yüzünü uygulayan diyalog nesnelerini özellik olarak barındırır. Adları *"GUIFrame_"* ile başlayan bu sınıflardan türetilen nesneler kullanıcı isteklerini yerine getirmede

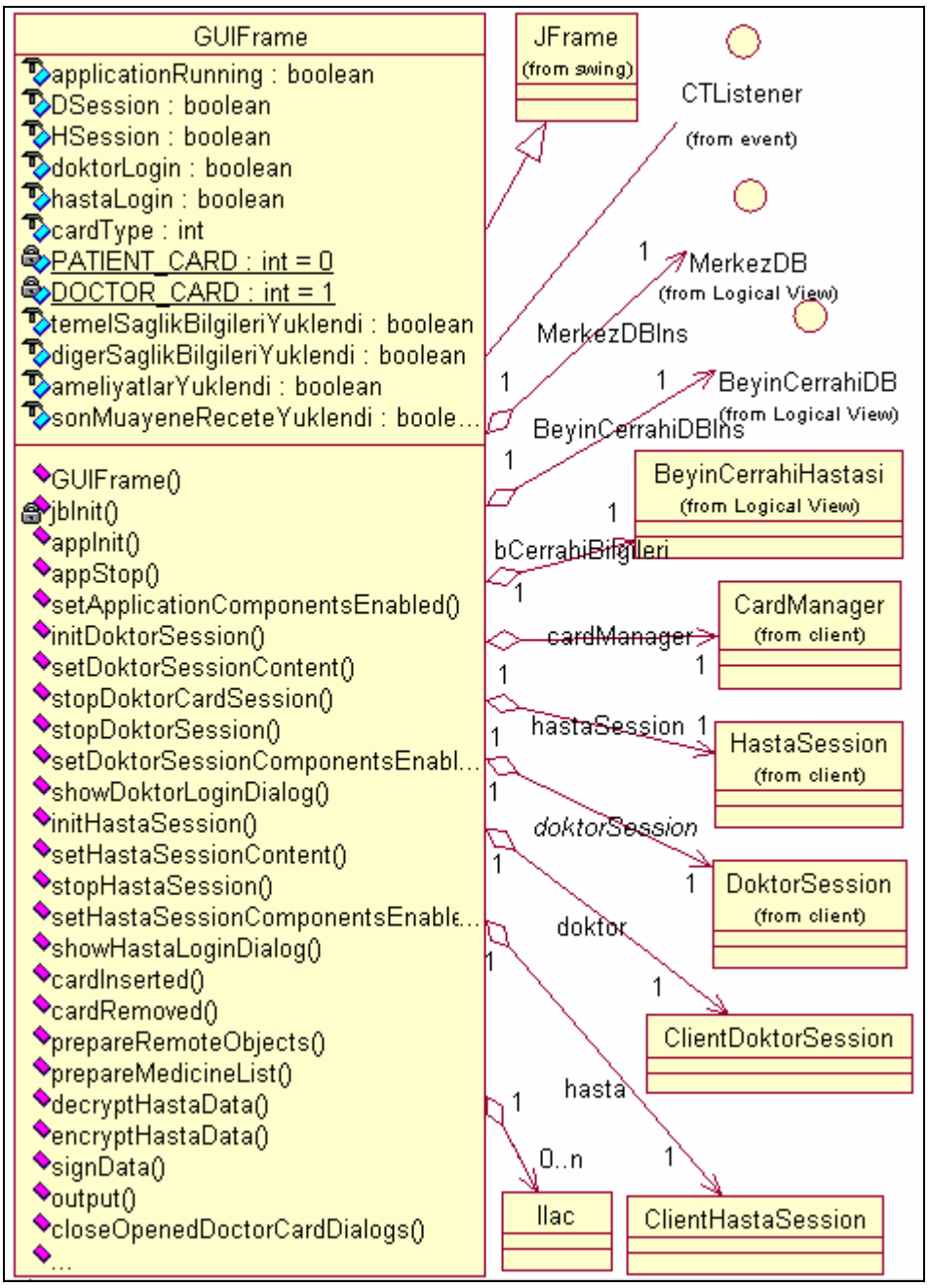
görev alacak grafik ara yüzleri oluştururlar. `GUIFrame_BeyinCerrahi`, `GUIFrame_BeyinCerrahi_Epikriz` ve `GUIFrame_BeyinCerrahi_AdliHeyetEEG` dışındaki diyalog nesneleri tüm klinikler için ortaktır.

Klinik yazılımı aynı makine üzerinde yer alan kart istemci ara yazılımı ve sistem RMI istemci bileşenleri ile iletişim halindedir (Şekil 6.14). Kart okuyucuya yerleştirilen akıllı kart ile ilgili tüm işlemler `tr.edu.ege.ube.akss.client` kart istemci ara yazılımı paketinde yer alan sınıflardan türetilen nesneler aracılığı ile gerçekleştirilir. Merkez ve klinik veritabanlarına bağlantı ise sistem dağıtık nesne protokolü üzerinden gerçekleştirilmektedir. Muayenehanelerde yer alan terminaller aynı zamanda birer sistem RMI istemcisidirler. Kart istemci ara yazılımı ve sistem RMI protokolü ile ilgili detaylı bilgi daha önceki bölümlerde yer almaktadır. Bu bölümde klinik paketinde yer alan ara yüz bileşenlerinin bu paketlerde yer alan nesneler ile iletişimi açıklanmıştır.

6.2.3 Yazılım ana formu

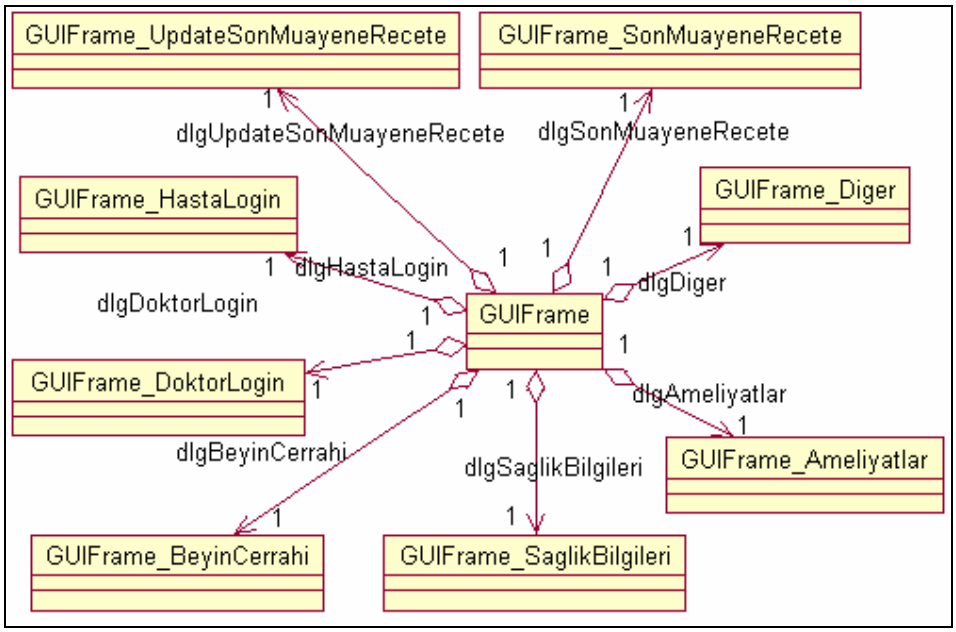
Klinik yazılımı çalıştırıldığında ilk yüklenen ara yüz `GUIFrame` sınıfından türetilen nesnedir. Şekil 6.15 ve 6.16'da görüldüğü gibi `GUIFrame` bir çok nesneyi özellik olarak barındırmaktadır. Bu nesneler içerisinde kart iletişiminde görev alan nesneler, RMI istemci bileşenleri ve kullanıcılara sunulan ara yüz diyalog nesneleri yer almaktadır.

`GUIFrame` nesnesi faaliyete geçtiğinde içerdiği ara yüz bileşenlerini hazırladıktan sonra ilk olarak kullanılacak olan akıllı kart ortamını başlatır. Bunun için `tr.edu.ege.ube.akss.client.CardManager` sınıfından bir nesne türetmesi ve özellik olarak saklaması yeterlidir. Oluşturulan `CardManager` nesnesi ilgili sistem özellik dosyasını okuyarak hangi bağlantı noktasından hangi kart okuyucu/yazıcı ünitesinin kullanılacağını belirler. Yine aynı pakette yer alan `tr.edu.ege.ube.akss.client.HastaSession` ve `tr.edu.ege.ube.akss.client.DoktorSession` nesneleri hazırlanır. Bu nesneler sırası ile doktor ve hasta akıllı kartları ile iletişimde kullanılacaktır.



Şekil 6.15 GUIFrame sınıfı (1. Kısım)

`CardManager`, `HastaSession` ve `DoktorSession` nesneleri ile ilgili bilgi kart istemci ara yazılımının anlatıldığı bölümde yer almaktadır.



Şekil 6.16 GUIFrame sınıfı (2. Kısım)

`GUIFrame` nesnesi aynı zamanda `opencard.core.event.CTListener` ara yüzünü uygulayarak kendisini sistem üzerindeki akıllı kart olay dinleyici kayıtçısına kaydetmekte böylelikle kart olaylarından haberdar olmaktadır. Daha önce de belirtildiği gibi `CTListener` uygulayıcıları bu ara yüze ait `cardInserted()` ve `cardRemoved()` metotlarını doldurarak bu olaylara karşılık hangi işlemleri yapacaklarını kontrol edebilirler. `CTListener` ara yüzü hakkında detaylı bilgi sistem alt yapı çalışmalarında verilmiştir.

`GUIFrame` nesnesi bir ara yüz bileşeni olarak ağ üzerindeki veritabanlarına bağlantıda sistem dağıtık nesne protokolünü kullanmakta olup bir RMI istemcisi olarak görev yapmaktadır. Merkez ve klinik veritabanlarına erişimde iki uzak nesneye ait ara yüzleri kullanır: `MerkezDB` ve `BeyinCerrahiDB`. Hazırlanan klinik yazılımı beyin

cerrahi bölümüne uygun olarak tasarlandığından iletişime geçilen RMI sunucusu bu bölüme ait sunucudur. Buna göre kullanılan klinik veritabanı erişim uzak nesnesi de `BeyinCerrahiDB` ara yüzüne sahiptir.

Uygulama ilk başlatıldığında kart ortamını hazırlandıktan sonra sıra uzak nesne bağlantılarını hazırlamaya gelir. `GUIFrame`, sistem özellik dosyasından hangi RMI sunucusu ile iletişime geçeceği bilgisini aldıktan sonra bu sunucunun RMI kayıtçısı ile iletişime geçerek bu sunucu üzerinde çalışan uzak nesnelere referans elde etmektedir. Böylece tüm uzak veritabanı işlemlerinden mimaride görünüm katmanında yer alan `GUIFrame` bileşeni soyutlanmış olur. Verileri uygun nesneler içerisine RMI kanalından alacak ve gönderecektir.

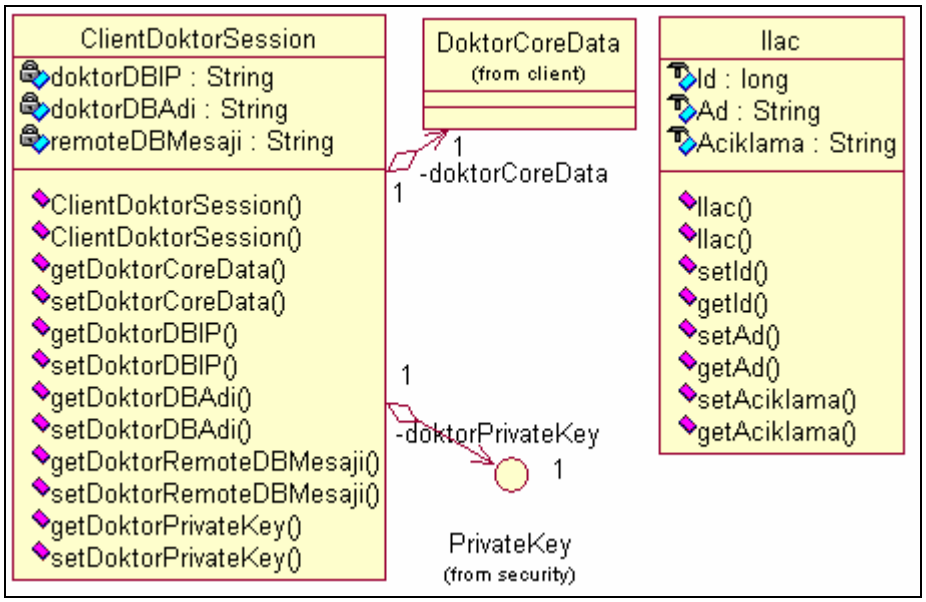
Uygulama başlangıcında yerine getirilen son işlem makine üzerinde yer alan lokal ilaç veritabanı ile bağlantıya geçilerek ilaç listesinin elde edilmesidir. Uygulamada ilaç tabloları MS Access veritabanında yer almakta; bu veritabanına erişim de JDBC-ODBC köprüsü üzerinden gerçekleştirilmektedir.

Veritabanından alınan ilaç kayıtlarının her biri yine klinik paketinde yer alan `Ilac` sınıfından türetilen nesneler içerisinde yer almakta; bu nesneleri barındıran `java.lang.Vector` nesnesi de `GUIFrame` içerisinde özellik olarak saklanmaktadır. Reçete yazma işleminde doktorun önüne bu `Ilac` nesnelerinin (Şekil 6.17) oluşturduğu liste getirilip seçim yapması istenecektir.

Tüm bu hazırlık işlemlerinden sonra uygulama doktor oturumu açmaya hazır hale gelir ve kart okuma ünitesine bir doktor kartının yerleştirilmesini bekler (Ek 2.1).

Şekil 6.16'da görüldüğü gibi `GUIFrame` nesnesi kullanıcılar ile etkileşimi sağlayacak olan diyalog nesnelerini de özellik olarak barındırmaktadır. Mönü üzerinden kullanıcı isteğini belirttiğinde bu isteği yerine getirecek olan diyalog penceresini sunacak olan nesne `GUIFrame`

tarafından oluşturulur. Kart olaylarına gereken tepkilerin verilmesi için bu diyaloglara ait referansları `GUIFrame` tutmaktadır. Örneğin okuyucu üniteden hasta akıllı kartı çıkarıldığında güvenlik nedeniyle hasta bilgilerini içeren tüm hasta diyaloglarının da kapanması ve oturumun sonlandırılması tasarlanmıştır. Böyle bir durumda `GUIFrame`, `CTListener` olarak bu olayı algılamakta ve halihazırda görev yapmakta olan diyalog nesnelerini sonlandırmaktadır.



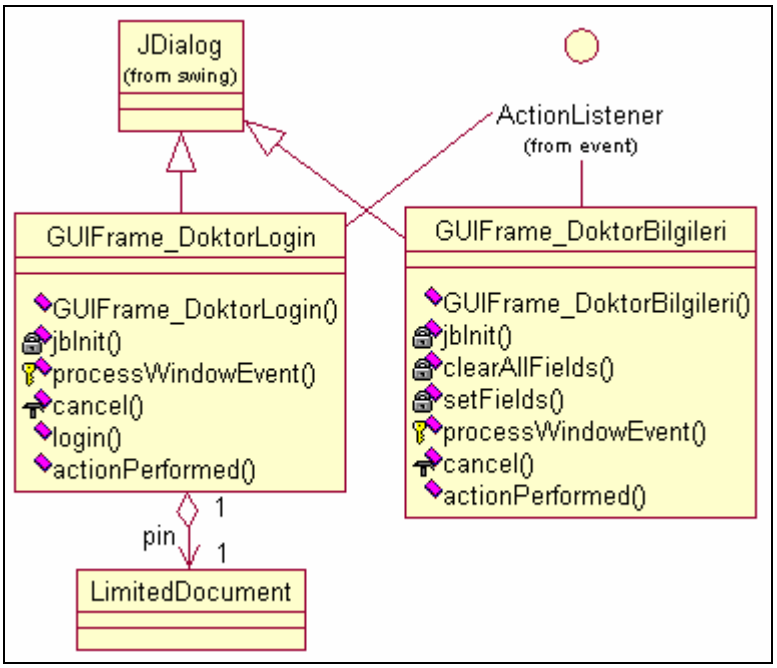
Şekil 6.17 ClientDoktorSession ve Ilac sınıfları

İzleyen bölümlerde uygulamada açılan kullanıcı oturumları ve bu oturumlar sırasında görev alan yazılım bileşenleri hakkında bilgi verilmiştir.

6.2.4 Doktor oturumu

Klinik uygulaması üzerinde herhangi bir işlem yapılabilmesi için ön şart bir doktor oturumunun açılmasıdır. Muayenehanede görevli doktor kendisine ait akıllı kart ile terminal üzerinde oturum açabilir.

Doktor kartı, kart okuyucuya yerleştirildiğinde `GUIFrame` nesnesi bundan haberdar olup kart oturumunu başlatır. Akıllı kart üzerinde doktor uygulaması kart istemci paketinde yer alan `CardManager` nesnesi aracılığı ile seçilerek iletişime geçirilir ve güvenli kanal oluşturulur. Bundan sonraki tüm doktor akıllı kartı iletişimi `GUIFrame` nesnesinin barındırdığı `tr.edu.ege.ube.akss.client.DoktorSession` nesnesinin sunduğu servisler aracılığı ile gerçekleştirilir. Bu aşamadan sonra kart sahibinden PIN girişi istenir (Ek 2.2). Giriş için ekrana getirilen diyaloğu `GUIFrame_DoktorLogin` sınıfından türetilen nesne hazırlamaktadır (Şekil 6.18).



Şekil 6.18 GUIFrame_DoktorLogin ve GUIFrame_DoktorBilgileri sınıfları

Doğru PIN girişi sonrası doktora ait kart üzerinde yer alan bilgiler uygulamaya transfer edilir. Bu bilgileri özellik olarak içeren nesne, `GUIFrame` nesnesinde saklanmakta olup `ClientDoktorSession` sınıfından (Şekil 6.17) türetilmektedir. Oturum boyunca bilgiler bu nesneden alınmaktadır.

Doktor kartı üzerinde yetkilendirme sağlanır sağlanmaz akıllı kart üzerindeki uzak veritabanı adresi alınarak RMI sunucusuna gönderilir. `MerkezDB` ara yüzü ile iletişime geçilen uzak nesneye RMI kanalı üzerinden bu adres bilgisi ve doktorun id'si gönderilir. Kanaldan dönen bilgi bu doktora ait uzak veritabanı mesajıdır (Ek 2.3)

Böyle bir mesaj alımı için bir GUI bileşeni olarak `GUIFrame`'in içerdiği kod sadece tek satırdan oluşur:

```
String mesaj = MerkezDBIns.getDoktorMesajiByIP ( doktorID, remoteDBIP);
```

`MerkezDBIns`, `MerkezDB` tipinde olup bunun üzerinden RMI sunucusunda yer alan uzak nesneye ait `getDoktorMesajiByIP()` metodu çağrılmaktadır. Uygun parametreler ile çağrılan metottan görüntülenecek mesaj döner. Görüldüğü gibi `GUIFrame` nesnesi mesaj alımında tamamı ile veritabanı erişimden ve ilgili kontrollerden bağımsızdır. Tüm bu hizmetleri `GUIFrame` için uzak nesne yerine getirmektedir.

Uzak mesajın da alınmasından sonra doktor akıllı kartına ait oturum sona ermektedir. Ancak uygulama üzerinde ilgili doktora ait oturum devam eder. Halihazırda bulunan doktor oturumu ile ilgili bilgiler uygulamadaki mönüden *"Doktor Bilgileri"* öğesi seçilerek elde edilebilir (Ek 2.4). Ekrana gelen diyaloğu `GUIFrame_DoktorBilgileri` sınıfından türetilen nesne hazırlar (Şekil 6.18). `GUIFrame_DoktorBilgileri` nesnesi kendisini oluşturan `GUIFrame` nesnesinin `ClientDoktorSession` nesnesi özelliğine erişerek bu nesneye ait özellikleri görüntülemektedir.

Doktor oturumu uygulama üzerinde hazırlandıktan sonra hasta kabullerine geçilebilir. Doktor oturumu sonlandırıldığında uygulama üzerindeki aktif doktor oturumu ve varsa hasta oturumları otomatik

olarak kapanır. Bu durumda hasta kabulleri için doktor akıllı kartı kullanılarak yeni bir oturumun açılması gerekmektedir.

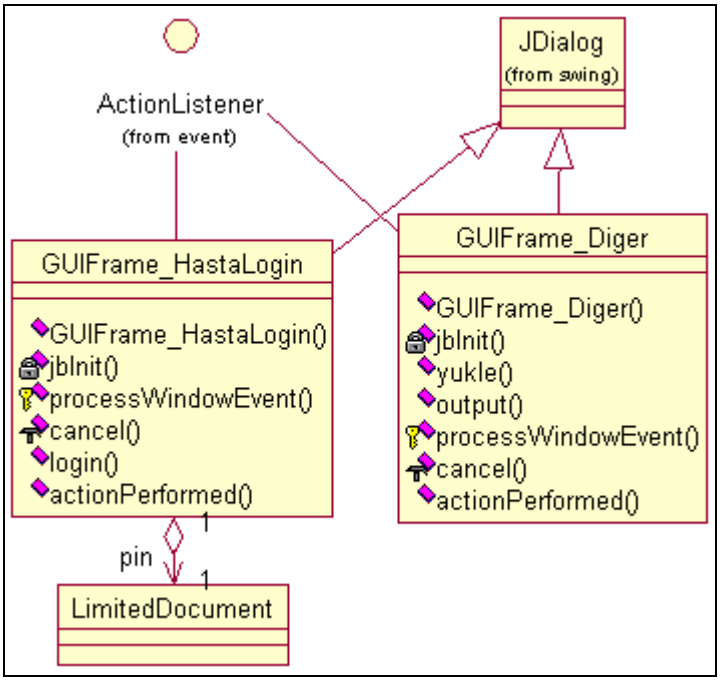
6.2.5 Hasta oturumu

Klinik yazılımı üzerinde bir doktor oturumu açıldıktan sonra hasta kabullerine geçilebilmektedir. Bu durum sağlandıktan sonra hastaya ait akıllı kart, kart okuyucuya yerleştirildiğinde yine kart uygulaması ile yazılım arasında iletişim kanalı kurulur. Bunu sağlayan `GUIFrame`'in `CardManager` nesnesidir. Bundan sonraki tüm hasta akıllı kartı iletişimi `GUIFrame` nesnesinin barındırdığı `tr.edu.ege.ube.akss.client.HastaSession` nesnesinin sunduğu servisler aracılığı ile gerçekleştirilir.

Doktor oturumuna benzer şekilde hasta oturumunda da PIN doğrulaması mevcuttur (Ek 2.5). PIN girişi için gerekli diyaloğu `GUIFrame_HastaLogin` sınıfından (Şekil 6.19) türetilen nesne ekrana getirir.

Doğru PIN girişi gerçekleştirildikten sonra hasta oturum bilgilerini hazırlamak üzere hasta akıllı kartından gerekli içerik alınmaktadır. Karttan sırasıyla hasta genel bilgileri, acil durum iletişim bilgileri, sigorta bilgileri ve DES anahtarı alınır ve `ClientHastaSession` sınıfından (Şekil 6.20) türetilen nesneye özellik olarak atanarak ekranda görüntülenir. Yine RMI istemci bileşenleri kullanılarak uzak veritabanından hastaya ait mesaj alınır ve ekranda görüntülenir (Ek 2.6). Dağıtık nesne iletişimi doktor oturumunda anlatıldığı gibidir.

Hasta oturumu hazırlanırken doktor oturumundan farklı olarak kliniğe ait veritabanından hastaya ait klinik bilgilerinin alınması da gerçekleşmektedir. Bu bilgi klinikten kliniğe değişmekle birlikte `GUIFrame` içerisinde bu bilgileri barındıracak bir nesnenin olması ve uzak metot çağırımları her klinik için yeterli olacaktır. Daha önce de belirtildiği gibi uygulamada Beyin Cerrahi bölümü ele alınmış ve bu bölüme ait RMI istemci ve sunucu bileşenleri hazırlanmıştır.

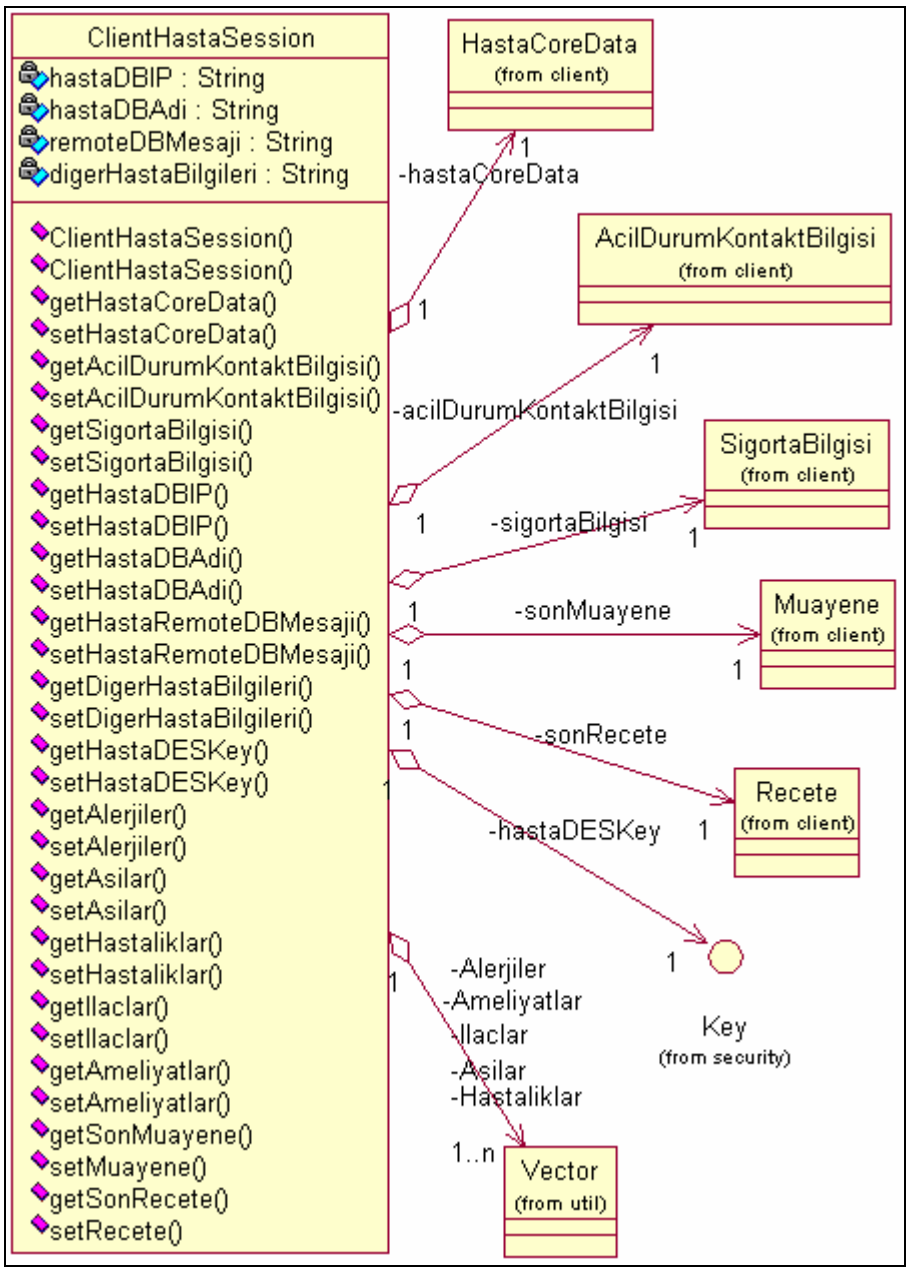


Şekil 6.19 GUIFrame_HastaLogin ve GUIFrame_Diger sınıfları

Beyin cerrahi veritabanından bilgi alımı ve gönderiminden sorumlu uzak nesne `BeyinCerrahiDB` ara yüzünü uygulamakta olup Beyin Cerrahi RMI sunucusu üzerinde çalışmaktadır. Klinik yazılımından `BeyinCerrahiDB` ara yüzü kullanılarak bu nesnenin ilgili metodu çağrılmaktadır:

```
byte[] encryptedBeyinCerrahiBilgileri = BeyinCerrahi
DBIns.getEncryptedHastaData(hastaID,        hastaDBIP,
doktorID);
```

`GUIFrame` içerisinde bu kodun işletilmesi ile uzak nesneden hastaya ait klinik bilgileri RMI kanalı üzerinden şifreli olarak alınmaktadır. Gelen bilgiler hasta akıllı kartında yer alan DES anahtarı ile çözülür ve ortaya çıkan `BeyinCerrahiHastasi` nesnesi `GUIFrame` içerisinde özellik olarak yer almaya başlar.



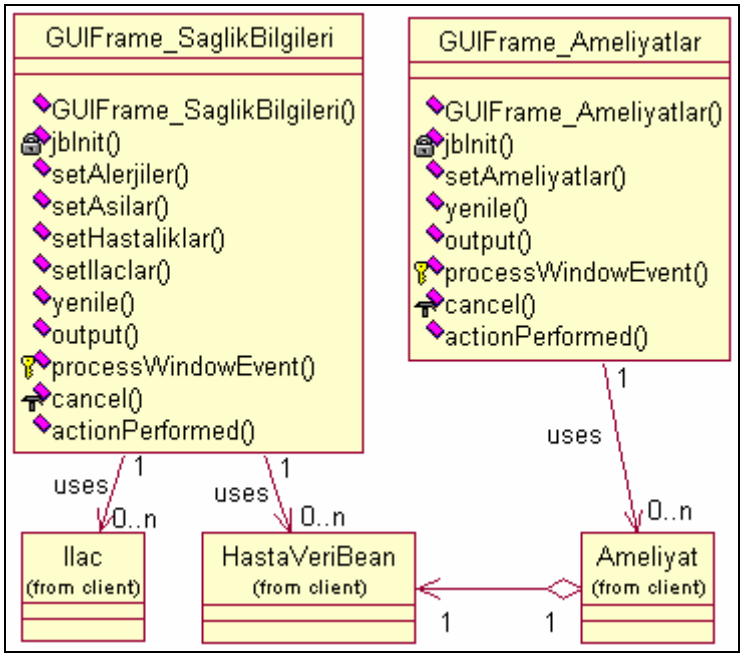
Şekil 6.20 ClientHastaSession sınıfı

Hatırlanacağı üzere `BeyinCerrahiDB`'yi uygulayan nesne RMI sunucusunda ilgili hastaya ait klinik bilgilerini, oluşturduğu bir

`BeyinCerrahiHastasi` nesnesine özellik olarak aktardıktan sonra bu nesneyi bir akışa çevirip hasta DES anahtarı ile şifrelemektedir. Alıcı tarafta da bunun tersi bir işlem yapılarak şifrelenen nesne elde edilmektedir. `BeyinCerrahi` RMI bileşenleri ile ilgili detaylı bilgi sistem sunucu katmanı yazılımının anlatıldığı bölümde yer almaktadır.

Kliniğe ait hasta bilgilerinin alınması ile birlikte hasta oturumu kullanıma hazır hale gelir. Doktor, hastasına ait kart üzerindeki sağlık bilgilerine ilgili diyaloglar aracılığı ile erişebilecektir.

Hasta mönüsünden *"Temel Sağlık Bilgileri"* öğesi seçildiğine doktorun önüne hastaya ait alerji ,aşı, kronik hastalık ve sürekli kullandığı ilaçlara ait bilgiler gelmektedir (Ek 2.7). Bu bilgilerin görüntülenmesini sağlayan nesne `GUIFrame_SaglikBilgileri` sınıfından türetilmiştir (Şekil 6.21).



Şekil 6.21 GUIFrame_SaglikBilgileri ve GUIFrame_Ameliyatlar sınıfları ve bunların kart istemci ara yazılımında yer alan nesneler ile olan ilişkileri

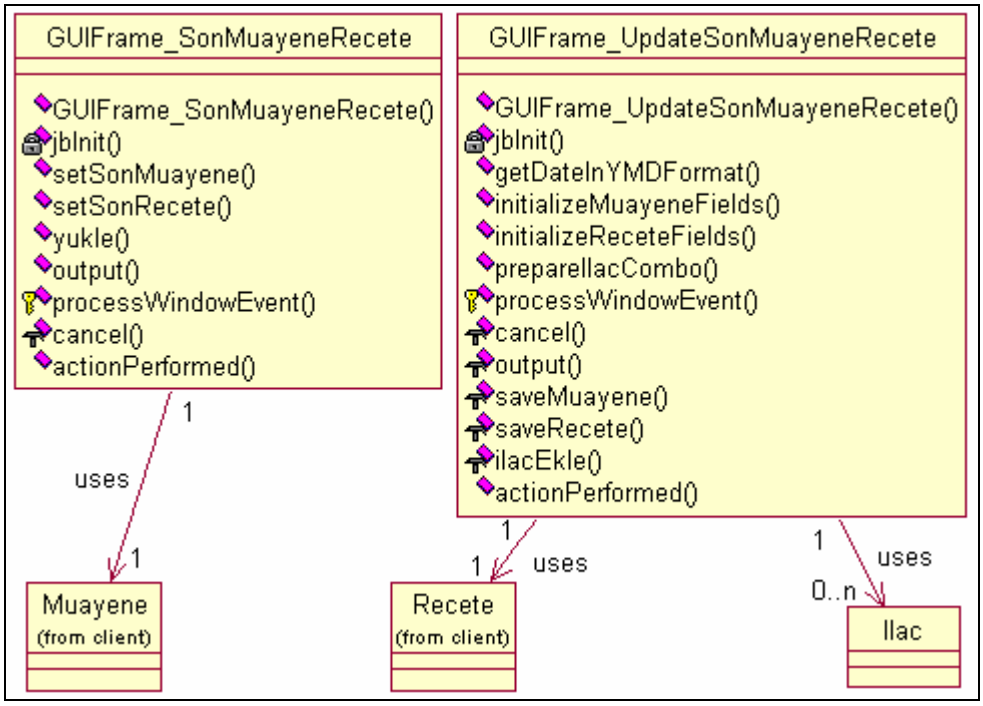
`GUIFrame_SaglikBilgileri` nesnesi `GUIFrame`'de özellik olarak yer alan `HastaSession` nesnesini kullanarak akıllı kart ile iletişimde bulunmakta ve hastaya ait alerji, aşı ve hastalık bilgilerini `tr.edu.ege.ube.akss.client.HastaVeriBean` nesneleri içerisinde elde etmektedir. Sürekli kullanılan ilaç bilgileri de `tr.edu.ege.ube.akss.client.Ilac` nesnelerinin oluşturduğu bir dizi olarak elde edilmektedir. Ara yüz bileşeni nesnenin yapması gereken tek iş bu `HastaVeriBean` ve `Ilac` nesnelerinin *getX* metotlarını kullanarak içerdikleri verilere erişmek ve bunları tablolar için de görüntülemektir. Elde edilen temel sağlık bilgileri ilgili `java.util.Vector` nesneleri içerisinde `ClientHastaSession` nesnesine de özellik olarak atanır. Böylelikle oturum süresince ilgili bilgileri elde etmek için tekrar hasta kartı ile iletişime geçmeye gerek yoktur.

Hastaya ait ameliyat bilgilerini ise `GUIFrame_Ameliyatlar` sınıfından (Şekil 6.21) türetilen diyalog nesnesi ekrana getirir. `GUIFrame_SaglikBilgileri` nesnesine benzer şekilde kart ile iletişime geçilerek hasta ameliyat bilgileri, `tr.edu.ege.ube.akss.client.Ameliyat` nesneleri içerisinde elde edilmektedir. `GUIFrame_Ameliyatlar` nesnesinin kullanıcıya sunduğu ara yüz Ek 2.8'de verilmiştir. Ameliyat nesnelerinin oluşturduğu vektör yine `ClientHastaSession` nesnesine özellik olarak atanmaktadır.

Yukarıda yer alan alanlara girmeyen hasta sağlık bilgileri bir metin alanı içerisinde doktora sunulmaktadır (Ek 2.9). Şekil 6.19'da bu bilgileri sunan ara yüzü hazırlayan nesnenin türetildiği `GUIFrame_Diger` sınıfı yer almaktadır.

Hasta kartında yer alan son muayene ve son reçete bilgilerine yine ara yüz üzerinden ulaşılabilmektedir. Bu bilgileri kullanıcıya sunan nesne `GUIFrame_SonMuayeneRecete` sınıfından (Şekil 6.22) türetilmektedir. Bu diyalog nesnesinin sunduğu ara yüz Ek 2.10'da yer almaktadır.

Muayene ve reçete bilgileri `HastaSession` nesnesinin sunduğu servisler aracılığıyla karttan sırası ile `tr.edu.ege.ube.akss.client.Muayene` ve `tr.edu.ege.ube.akss.client.Recete` nesneleri olarak alınmaktadır. `GUIFrame_SonMuayeneRecete` nesnesi bu nesnelere ait içerikleri ilgili *getX* metotları ile alarak görüntüler. Bu bilgiler karttan ilk kez alındığında ilgili `Muayene` ve `Recete` nesneleri hasta oturum bilgisini tutan `ClientHastaSession` nesnesine de özellik olarak atanır. Böylelikle oturum içinde tekrar bu bilgilere ulaşılmak istendiğinde kart iletişimine ihtiyaç duyulmayacaktır. Yalnız burada unutulmaması gereken `ClientHastaSession` nesnesine de ancak hasta akıllı kartı okuyucu üzerinde iken ve kart oturumu açıkken erişilebilir. Kart, okuyucudan çıkarıldığı anda `ClientHastaSession` nesnesinin içeriği de otomatik olarak silinir. Böylelikle akıllı kart olmadan bilgi erişimine izin verilmemektedir.



Şekil 6.22 GUIFrame_SonMuayeneRecete ve GUIFrame_UpdateSonMuayeneRecete sınıfları

Doktorların, muayene sonrası muayene bilgilerini ve varsa reçete bilgilerini hasta akıllı kartlarına yazmasını sağlayacak olan ara yüz `GUIFrame_UpdateSonMuayeneRecete` (Şekil 6.22) nesnesi tarafından sağlanmaktadır. Bu diyalog nesnesinin sunduğu ara yüz Ek 2.11'de görülmektedir.

`GUIFrame_UpdateSonMuayeneRecete`, uygulama üzerinde oturumu bulunan doktora ait olan ve muayene ve reçetede bulunması gereken bilgileri `ClientDoktorSession` nesnesinden alarak ekrana getirir. Sistem tarihi de alınarak formlara aktarılır. Doktorlar sadece muayene açıklamasını ve reçeteyi yazmakla yükümlüdürler. Reçetede yer alacak ilaçların seçimi için bir liste hazırlanmakta ve doktora sunulmaktadır. Hatırlanacağı üzere uygulama ilk başlatıldığında `GUIFrame` nesnesi yerel ilaç veritabanı ile iletişime geçerek ilaç listesini elde etmekte ve her bir ilaca ait özellikleri de aynı pakette yer alan `Ilac` nesnelerinde saklamaktadır. `GUIFrame_UpdateSonMuayeneRecete` nesnesi, listeyi oluşturmak için bu `Ilac` nesnelerinin oluşturduğu diziyi kullanır.

Doktor muayene açıklama bilgisini ve reçete içeriğini yazdıktan sonra ara yüzden karta kaydetme işlemini seçebilir. Onayı alındıktan sonra hasta akıllı kartına bu bilgiler yazılır. Muayene sonrası hasta akıllı kartı ile sistem yönetim birimine başvurarak reçetesine onay alır ve ilgili muayene ve reçete bilgileri hastane veritabanına aktarılır.

Buraya kadar anlatılan bileşenler tüm hastane bölümlerinde yer alacak klinik yazılımlarında ortak olacak şekilde tasarlanmıştır. Kliniğe özel hasta bilgilerinin doktor tarafından elde edilmesi ve güncellenmesi kliniklere özel olacaktır. Uygulamada Beyin cerrahi için örnek bir ara yüz oluşturulmuş ve kullanılmıştır.

"Hasta" mönüsünden "Beyin Cerrahi Bilgileri" öğesi seçildiğinde doktorun önüne hastaya ait beyin cerrahi bilgileri gelmektedir. Daha önce de belirtildiği gibi bu bilgiler oturum başlangıcında RMI sunucusunda görev alan uzak nesne aracılığı ile klinik veritabanından alınmakta ve

`GUIFrame` nesnesinin özelliği olan `BeyinCerrahiHastasi` nesnesinde saklanmaktadır.

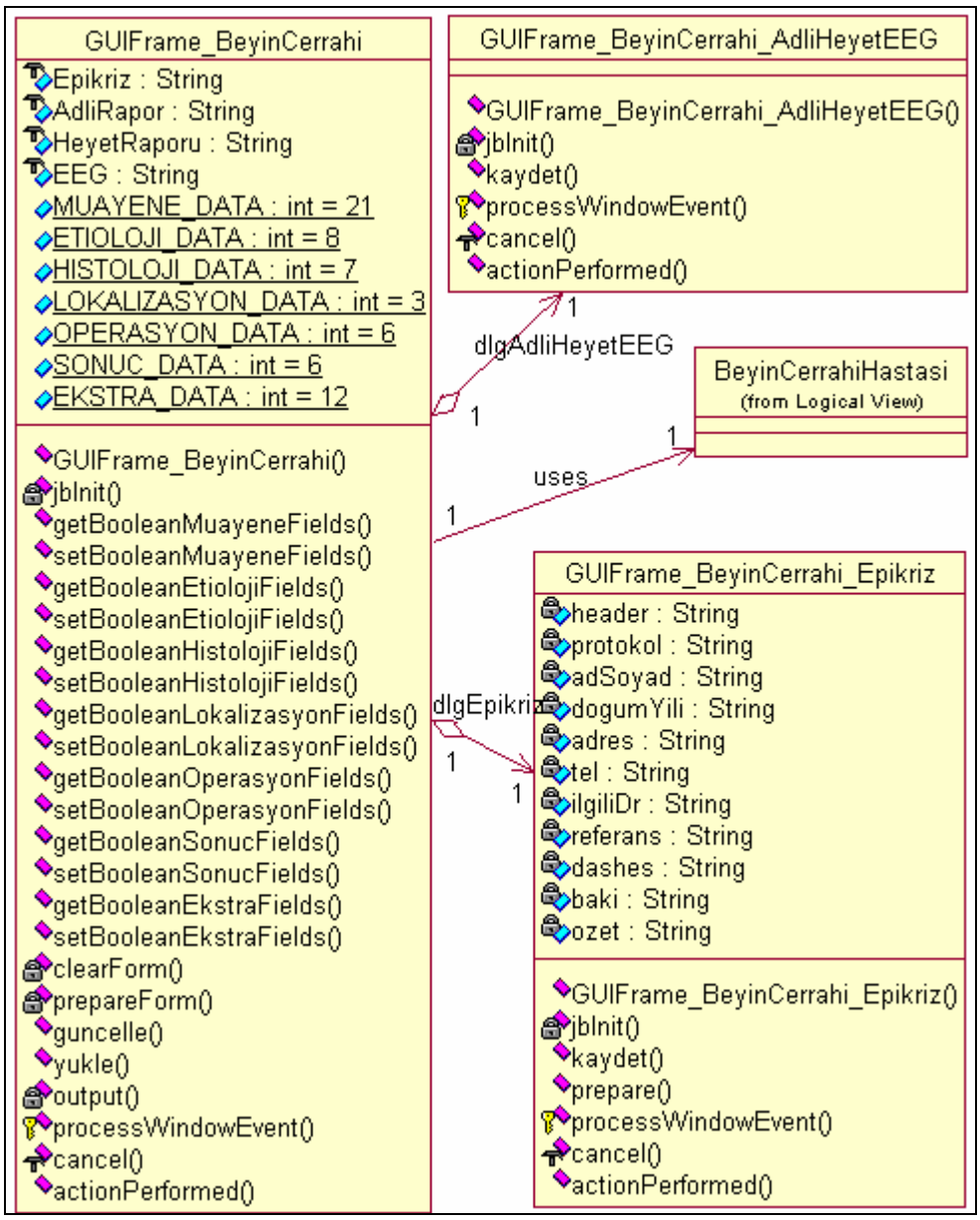
Hasta beyin cerrahi bilgileri, `GUIFrame_BeyinCerrahi` sınıfından (Şekil 6.23) türetilen bir nesnenin sunduğu ara yüzde kullanıcıya sunulmaktadır (Ek 2.12). `GUIFrame_BeyinCerrahi_Epikriz` ve `GUIFrame_BeyinCerrahi_AdliHeyetEEG` nesneleri ise hastaya ait raporları sunan diyaloglardır (Ek 2.13 ve Ek 2.14).

Doktor, hastaya ait klinik bilgilerini veritabanında güncellemek istediğinde devreye veri şifreleme ve imzalama işlemleri girmektedir. Hasta bilgilerini içeren `BeyinCerrahiHastasi` nesnesi öncelikle bir bayt dizisine dönüştürülür. DES şifrelemesinde kullanılacak olan `java.crypto.Cipher` nesnesi hastaya ait DES anahtarı ile ilklenir ve bayt dizisini şifreler.

Elde edilen şifrelenmiş bayt dizisininin imzalanması ise şu şekilde gerçekleşmektedir: `java.security.Signature` sınıfından oluşturulan nesne, oturumu açık olan doktora ait DSA özel anahtarı ile ilklenir. Bu özel anahtar `ClientDoktorSession` nesnesinin barındırdığı `java.security.PrivateKey` nesnesi içerisinde yer almaktadır. Şifrelenmiş hasta bilgilerine ait bayt dizini bu anahtar ile işlem gördüğünde imzalanmış bayt dizini çıktı olarak alınmaktadır.

Şifrelenmiş ve imzalanmış verilerin klinik veritabanında güncellenmesi için ara yüz bileşeninin, RMI sunucusu üzerinde bulunan bu işle ilgili uzak nesnenin – ki bu nesne `BeyinCerrahiDB` ara yüzünün uygulayan nesnedir – metodunu çağırması yeterlidir.

```
BeyinCerrahiDBIns.updateSignedAndEncryptedHastaData(
hastaID, hastaDBIP, doktorID, doktorDBIP,
encryptedData, signatureBytes);
```



Şekil 6.23 GUIFrame_BeyinCerrahi, GUIFrame_BeyinCerrahi_Epikriz ve GUIFrame_BeyinCerrahi_AdliHeyetEEG sınıfları

RMI kanalı üzerinde gelen parametreleri kullanan uzak nesne önce imza doğrulamasını gerçekleştirerek doktorun yetkili olup olmadığını kontrol eder. Daha sonra imzalanmış şifreli verileri hastaya ait DES anahtarı ile çözer ve klinik veritabanında günceller. Bu işlemlerin RMI sunucu üzerinde nasıl gerçekleştirildiği hakkında detaylı bilgi sistem sunucu katmanı yazılımının anlatıldığı bölümde verilmiştir.

Uzak metot işlem sonucu istemci tarafına yine kanal üzerinden dönmekte; bu sonuç da ara yüz bileşeni aracılığı ile kullanıcıya aktarılmaktadır.

Tüm muayene işlemleri tamamlandığında kullanıcı yine ara yüz üzerinden hasta oturumunu sonlandırır. Oturum sona erdiğinde `ClientHastaSession` ve `BeyinCerrahiHastasi` nesnelerinin içerikleri boşaltılmakta ve hasta işlemlerine ait mönü de etkisiz hale getirilmektedir. Hasta kartının okuyucudan çıkarılması sonucunda uygulama tekrar yeni bir hasta kartının takılması olayını dinlemeye geçer (Ek 2.15).

7 SONUÇ

Bu çalışmada hazırlanan Akıllı Kart Sağlık Sistemi ile bir hastane bünyesinde akıllı kartlara dayalı bir otomasyonun nasıl gerçekleştirilebileceği ortaya konmuştur. İlk olarak ihtiyaçları karşılayacak akıllı kartlara ait tasarımlar belirlenmiş daha sonra sistem dağıtık protokolü ve bu protokolde yer alacak sistem bileşenleri hazırlanmıştır.

Sistemde akıllı kartlar mobil bilgi taşıma dışında güvenlik ve yetkilendirme prosedürlerinde de kullanılacak şekilde hazırlanmıştır. Tasarımda savunulan; akıllı kartların sahibini tanımlayıcı bilgiler dışında tüm hastane birimlerince kullanılabilecek genel bilgileri de içermesidir. Özellikle hasta kartları sadece belli bir hastane bölümüne yönelik bir yapıda olmayıp bölümlere özelleşmiş bilgilere erişimde kullanılmaktadır. İçerdikleri genel bilgiler ise herhangi bir veritabanına erişme imkanı olmayan kart terminallerinden (örneğin ambulansta yer alan bir bilgisayar) uygun yazılım aracılığı ile erişilebilir.

Bu çalışma kapsamında akıllı kart iletişim bileşenleri dışında üç katmanlı mimariye uygun yazılımlar da hazırlanmıştır. Özellikle veri, iş ve ara yüz bileşenlerinin mümkün olduğunca modüler tasarımı yazılım mimarisinin esnek olması sonucunu doğurmuştur. İstemci / sunucu iletişimleri için hazırlanan dağıtık nesne protokolü ise sistemde planlanan güvenlik ve yetkilendirme özelliklerinin kullanılmasına ve sistemin genel ağlar üzerinde bile çalışabilmesine imkan vermiştir.

Sistemin uygulamasında muayenehanelerde çalışmak üzere tasarlanmış olan istemci yazılım paketleri Intel PIII 650 MHz istemciye sahip üzerinde Microsoft Windows 2000 işletim sistemi çalışan bir bilgisayarda çalıştırılmıştır. Aynı zamanda kart terminali olan bu bilgisayar üzerinde Java 1.3.1 çalışma ortamı ve OCF 1.2 kütüphanesi de yer almaktadır.

Sistem dağıtık nesne protokolünde yer alan sunucu bileşenlerinin üzerinde çalıştığı bilgisayar ise Intel P4 1.4 GHz işlemcili olup üzerinde Microsoft Windows XP işletim sistemi yer almıştır. Java çalışma ortamı olarak da JRE 1.4.0 bulunmaktadır.

Sistem bileşenlerinin tamamı Java programla dili kullanılarak geliştirildiğinden tüm sistem platform bağımsızdır. Üzerinde Java sanal makinesi yer alan her platformda çalışabilecek özelliktedirler.

Hazırlanan sistem, tüm bileşenleri ile çalıştırılmış ve protokolün işleyişi test edilmiştir. Gerek kart iletişimlerinde gerek ağ üzeri nesne iletişimlerinde olağandışı bir durumla karşılaşılmamış, önceki bölümlerde açıklanan tüm bileşenler görevlerini başarı ile yerine getirmişlerdir.

Testler sırasında akıllı kart ve bilgisayar üzerinde çalışan istemci uygulamalar arasındaki veri alışverişi sırasında geçen süreler de ölçülmüştür. 9600 baud'luk standart akıllı kart – bilgisayar veri yolu üzerinden karta *"Durum 2 İletişim Modeli"*ne uygun bir komut APDU'sunun gönderilmesi ve 255 baytlık veri içeren bir cevap APDU'sunun alınıp ara yüzde içeriğinin görüntülenmesi yaklaşık 1.5 saniyede gerçekleşmektedir. Akıllı karta muayene bilgisinin yazılması gibi, *"Durum 3 İletişim Modeli"*ne uygun bir komut APDU'nun gönderilip cevap APDU'nun alınması ise yaklaşık 2 saniyede gerçekleşmektedir. Tabi bu sürelere bilgisayar üzerindeki sanal makinede ara yüz ve kart istemci ara yazılımına ait nesnelerinin birbirleri ile olan iletişimleri için geçen süreler de dahildir. Okuyucuya yerleştirilen bir hasta veya doktor kartının algılanıp bu karta uygun oturumun açılmasına ve PIN girişi için gerekli diyalogun görüntülenmesine kadar geçen süre ise yaklaşık 9 saniyedir.

Sistem hazırlanırken karşılaşılan en önemli problem sağlık sektöründe bilgi saklama ve geri getirmeye dair bir standardizasyonun olmamasıdır. Özellikle sağlık bilgilerinin dünya geneli bir kodlamaya sahip olmasının bu tip otomasyon sistemlerinin tasarımlarında büyük kolaylıklar sağlayacağı açıktır. Bu nedenle sağlık bilgilerinin saklanacağı

veritabanının tasarımında akıllı kart kullanımı ihtiyacına yönelik bu çalışmaya özel bir kodlamaya gidilmiştir.

Yukarıda bahsedilen bir kod sisteminin ve hasta protokollerinin daha net belirlenmesi ile birlikte tez kapsamında geliştirilen sistemin gerçek bir hastane otomasyon sistemi ile bütünleşik olarak çalışması mümkündür.

Akıllı kartların özellikle mobil hasta bilgisi taşıması ve güvenlik özellikleri ile varolan sağlık sistemlerinde büyük fayda sağlayacağına inanılmaktadır. Hatta kart kapasitelerinin artması ve aynı zamanda maliyetlerinin daha ucuz olması ilgili süreci hızlandıracaktır. Örneğin sistemdeki hasta kartlarında yer kısıtlarından dolayı hastaya ait sadece son reçete ve muayene bilgileri yer almıştır. Ancak ileriye yönelik bu tez temel alınarak yapılacak başka bir çalışmada daha büyük kapasiteye sahip akıllı kartların kullanılması ile birden fazla muayene ve reçete bilgisi ya da başka diğer bilgilerin de (örneğin röntgen filmleri, test dokümanları, vb.) kartlar üzerinde taşınması mümkündür. Bunun yanı sıra sistem örneğin eczaneleri de kapsayacak bir biçimde genişletilebilir (Hasta kartlarında yer alan reçetelerin eczanelerde işlem görmesi için gerekli bileşenlerin hazırlanması, vb.).

Sistemin akıllı kart olanakları dışında ortaya konan dağıtık protokolü ise özellikle MVC sistem yazılım mimarisine dayalı bilgi sistemlerinin tasarımına örnek gösterilebilir. Devam çalışmalarında bu nesne protokolünde gelişmiş RMI olanaklarının ve J2EE platformunda yer alan EJB mimarisinin kullanılması iş süreçlerinin daha kolay yönetildiği, daha etkin bir sistemin ortaya çıkmasına neden olabilir. Bunların yanı sıra güvenlik ve yetkilendirme de kullanılan nispeten kolay teknolojiler yerine gerçek sistem uygulamasında daha gelişmiş teknolojilere başvurulabilir (örneğin DES yerine RSA uygulanması gibi) – ki bu şekildeki bir modifikasyon sistemin katmanlı mimarisi sayesinde oldukça basit olarak gerçekleştirilecektir.

Ortaya konan sistemin akıllı kart tabanlı sağlık sistemlerinin Türkiye'de de geliştirilmesinde genel bir çatı sunması mümkündür. Nispeten yeni olan akıllı kart teknolojilerinin kullanılmasına dayanan daha sonraki çalışmalar için de kaynak özelliğindedir.

KAYNAKLAR DİZİNİ

Anderson K., Marshall N., Melnyk M., Schaefer L., 1997, *Smart Cards In Health Care Industry*, Management of Technology I, ISyE/MGT/PubP 6771, 12p.

Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M., 1996, *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*, John Wiley & Son Ltd, New York - USA, 476p

Chen Z., 2000, *Java CardTM Technology for Smart Cards Architecture and Programmer's Guide*, Addison-Wesley, Massachusetts - USA, 368p

Danny A., Li S., Houle P., Wilcox M., Phillips R., Mohseni P., Zeiger S., Bergsten H., Ferris M., Diamond J., Bogovich M., Fleury M., Vedati K., Halberstadt A., Patzer A., 1999, *Professional Java Server Programming: with Servlets, JavaServer Pages (JSP), XML, Enterprise JavaBeans (EJB), JNDI, CORBA, Jini and Javaspaces*, Wrox Press Inc., USA, 1121p

Gamma E., Helm R., Johnson R., Vlissides J., 1994, *Design Patterns - Elements Of Reusable Object-Oriented Software*, Addison-Wesley, Massachusetts - USA, 395p

Hansmann U., Nicklous M. S., Schack T., Seliger F., 2000, *Smart Card Application Development Using Java*, Springer, Berlin - Germany, 293p

Health Smart Card, Health Smart Card, URL: *www.healthsmartcard.net*, last update: 2000

KAYNAKLAR DİZİNİ (devam)

Horstmann C. S., Cornell G., 2000, *Core Java 2 Volume II - Advanced Features*, Sun Microsystems Press, California - USA, 920p

Netcards, GIE Sesam Vitale, *Trans-European Healthcare Facility Service for Mobile Citizens*, URL: *http://www.sesam-vitale.fr/html/projets/netcards/index_eng.htm*, last update: November 2002

Netcards Project, Netcards Consortium, *Trans-European Healthcare Facility Service for Mobile Citizens*, URL: *http://www.netcards-project.com/*, last update: 2003

Netlink, GIE Sesam Vitale, *Netlink Project*, URL: *http://www.sesam-vitale.fr/html/projets/netlink/index.htm*, last update: July 2002

Novak R., Kandus G., Trcek D., 1999, *Further Development of a Smart-card Based Health Care Information System in Slovenia*, Presented at the Fifth International Congress on Conference and Exhibition on Cards Applications in Health Care: Health Cards'99, Milan (Italy)

Novak R., Kandus G., Trcek D., 2001, *Slovene Smart-card and IP Based Health-Care Information System Infrastructure*, International journal of medical informatics, vol. 61, Elsevier, pages 33-43, 11p.

OpenCard Consortium, 1999, *OpenCard Framework 1.2 Programmer's Guide*, IBM Deutschland Entwicklung GmbH, Boeblingen - Germany, 82p

KAYNAKLAR DİZİNİ (devam)

Pagetti C., Mazini C., Pierantoni M., Gualandi G., Schepel H., 2001, *A European Health Card Final Report*, European Parliament, Directorate General for Research, Document for STOA Panel, pages 16-29, 14p.

Rankl W., Effing W., 2000, Smart *Card Handbook Second Edition*, John Wiley & Sons, West Sussex - England, 746p

Transcards, GIE Sesam Vitale, *Transcards Project*, URL: *http://www.sesam-vitale.fr/html/projets/transcards/tcd_accueil_eng.htm*, last update: 2002

Tunalı T., Yıldırım Ş., Dalbastı T., 2002, *The Use of Smart Cards in Health Care*, IST-2000-26429 Hermes Project Workshop, June 10-12, Co-sponsored by TUBITAK and METU, pages 1-6, 6p.

EKLER

Ek 1 Sistem Yönetim Yazılımı Ekran Görüntüleri

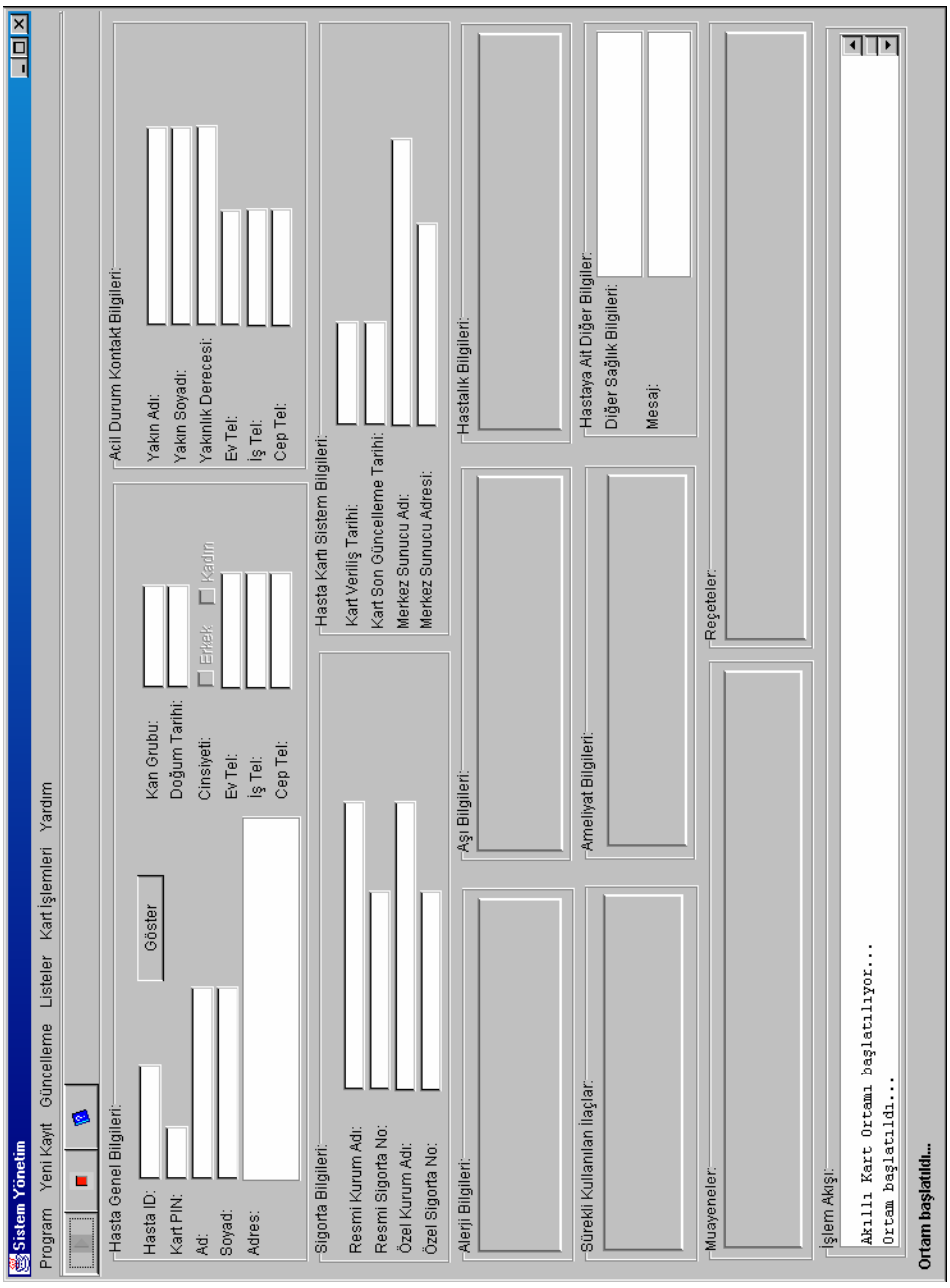
Ek 2 Klinik Yazılımı Ekran Görüntüleri

Ek 3 Sunucu Katmanı Yazılımı Ekran Görüntüleri

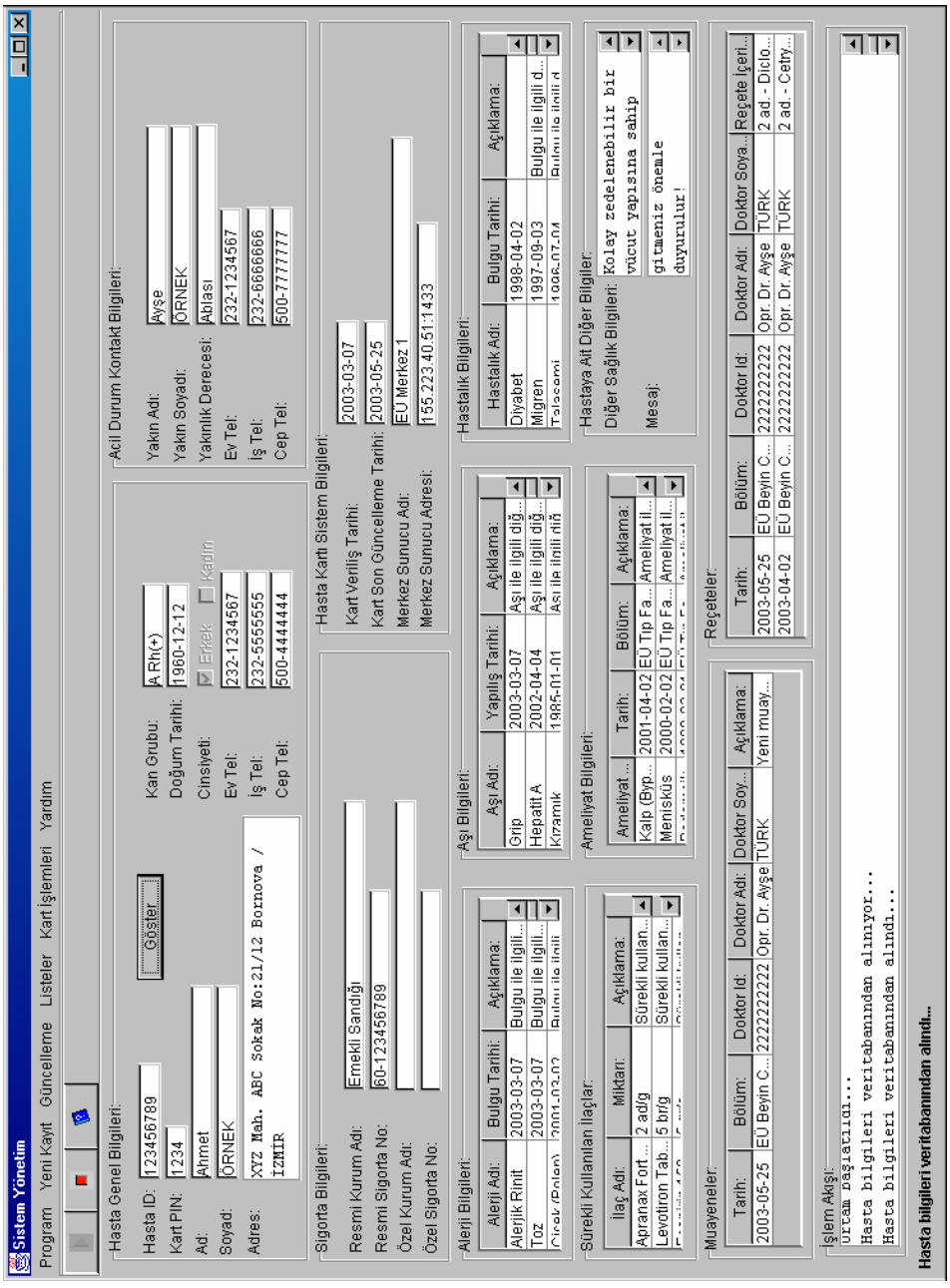
Ek 4 İngilizce Terimler Sözlüğü

Ek 1 Sistem Yönetim Yazılımı Ekran Görüntüleri

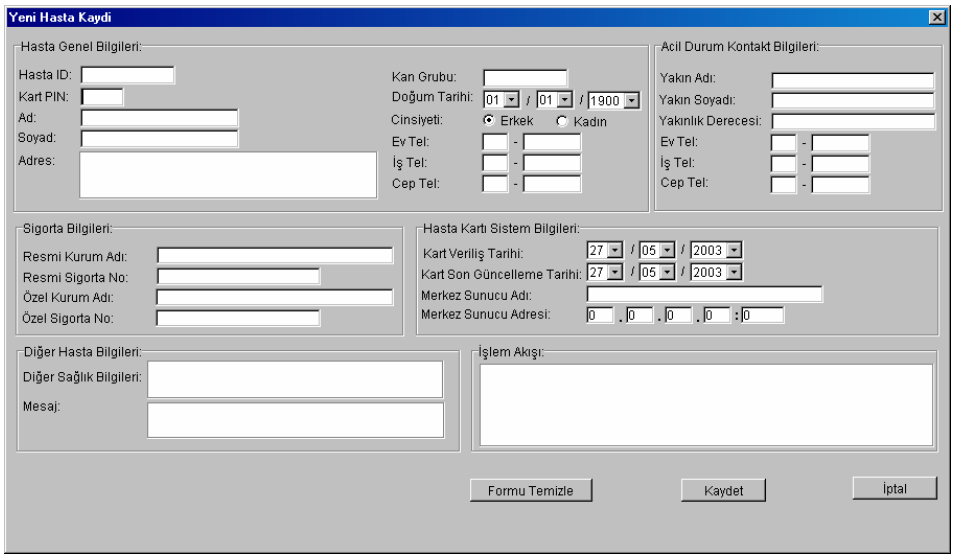
1.1 Sistem Yönetim Uygulamasının Başlatılması



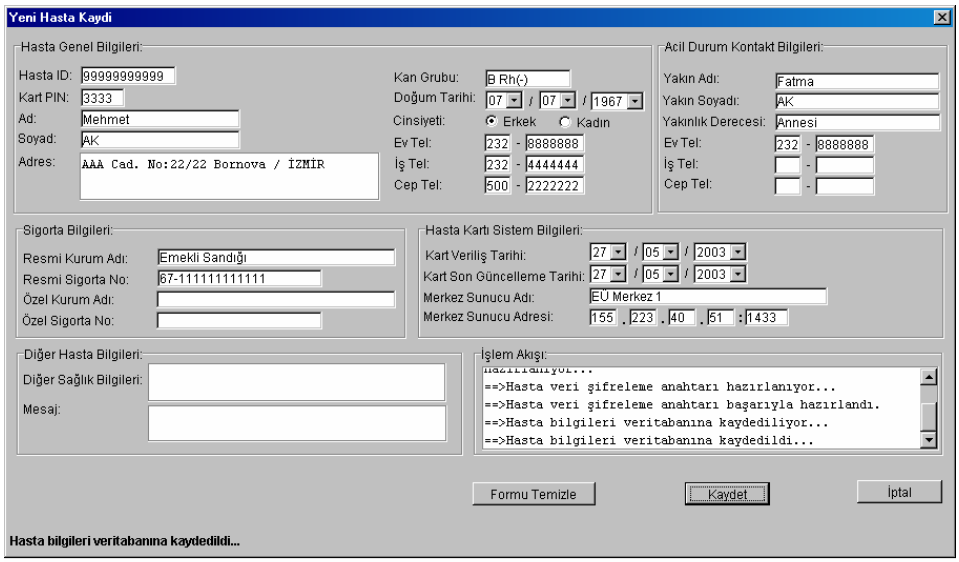
1.2 Hasta Kayıtlarının Elde Edilmesi



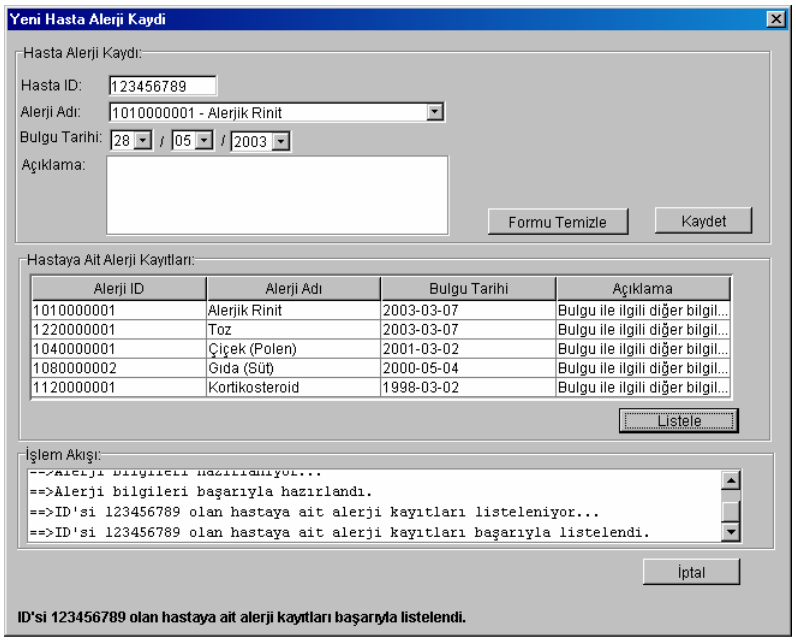
1.3 Yeni Hasta Kayıt Formu



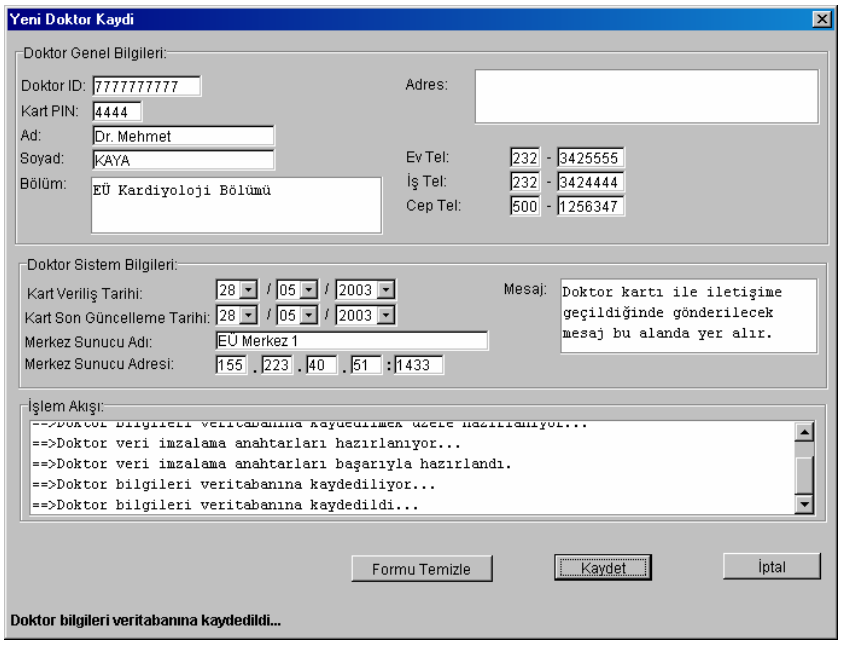
1.4 Yeni Hasta Bilgilerinin Kaydedilmesi



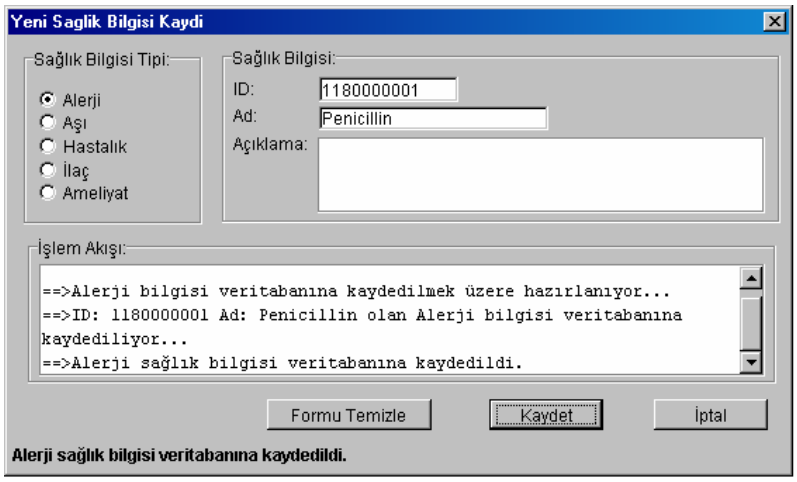
1.5 Yeni Hasta Alerji Bilgisi Kaydı



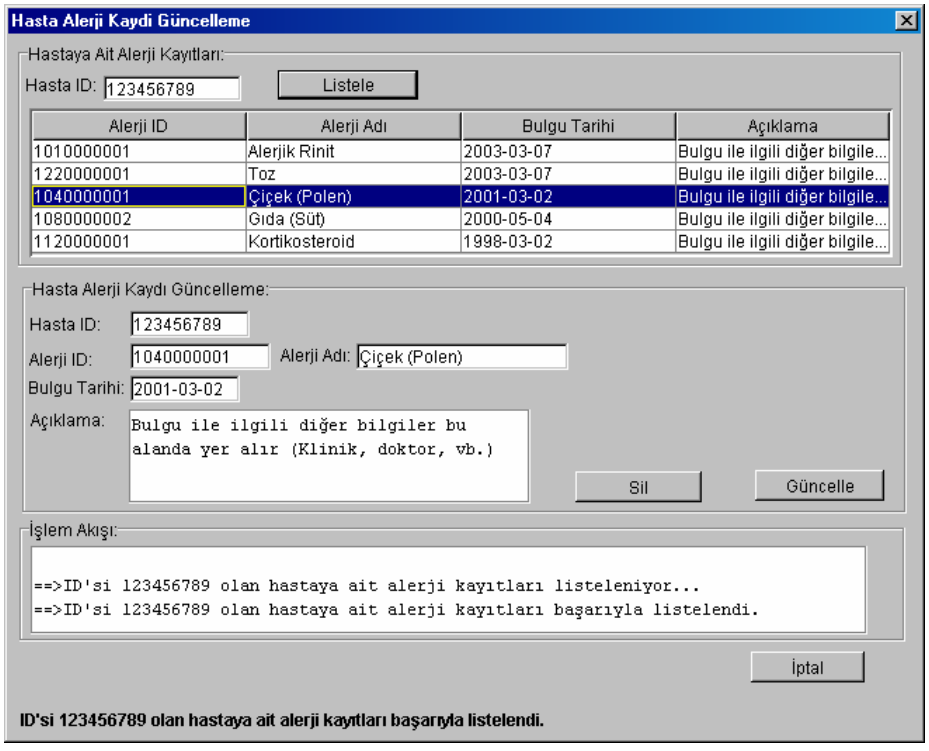
1.6 Yeni Doktor Kaydı



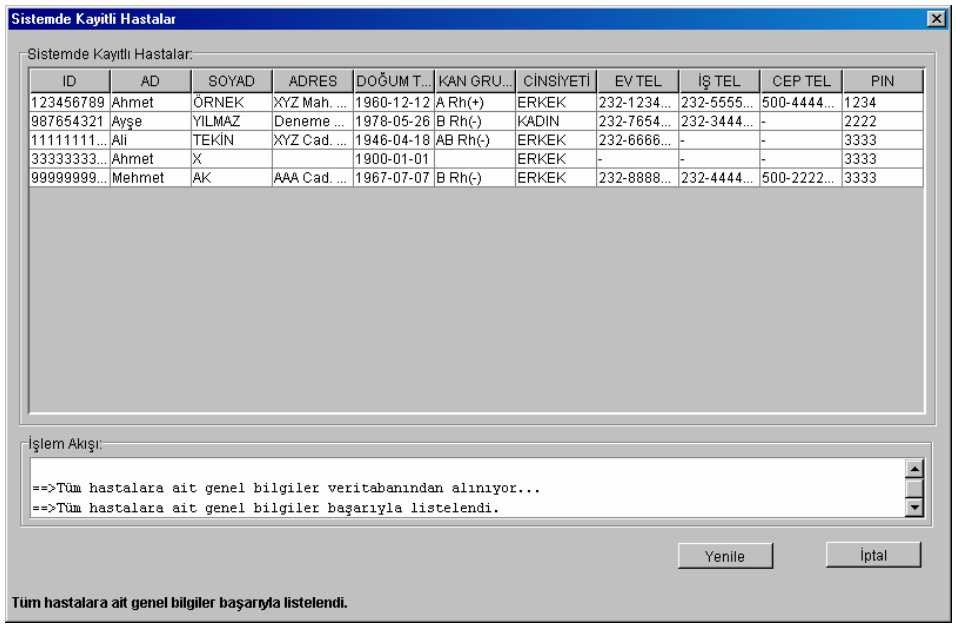
1.7 Yeni Sağlık Bilgisi Kaydı



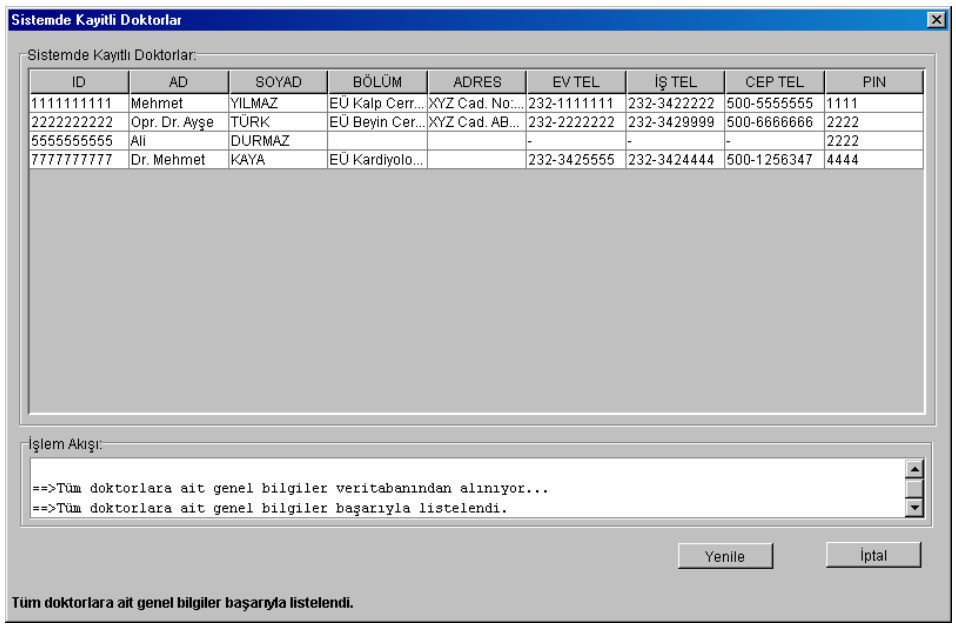
1.8 Hasta Alerji Kaydı Güncelleme



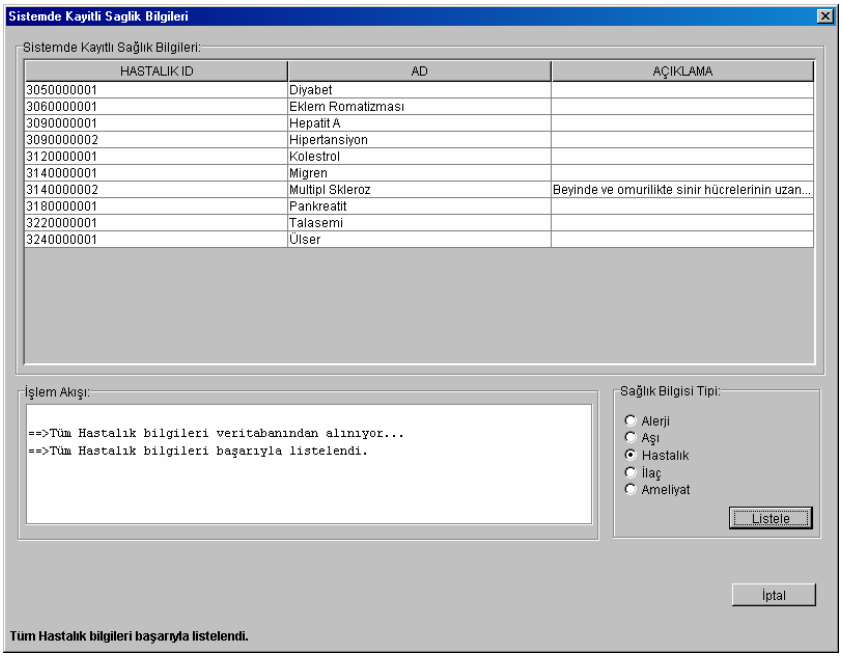
1.9 Kayıtlı Hastalar Listesi



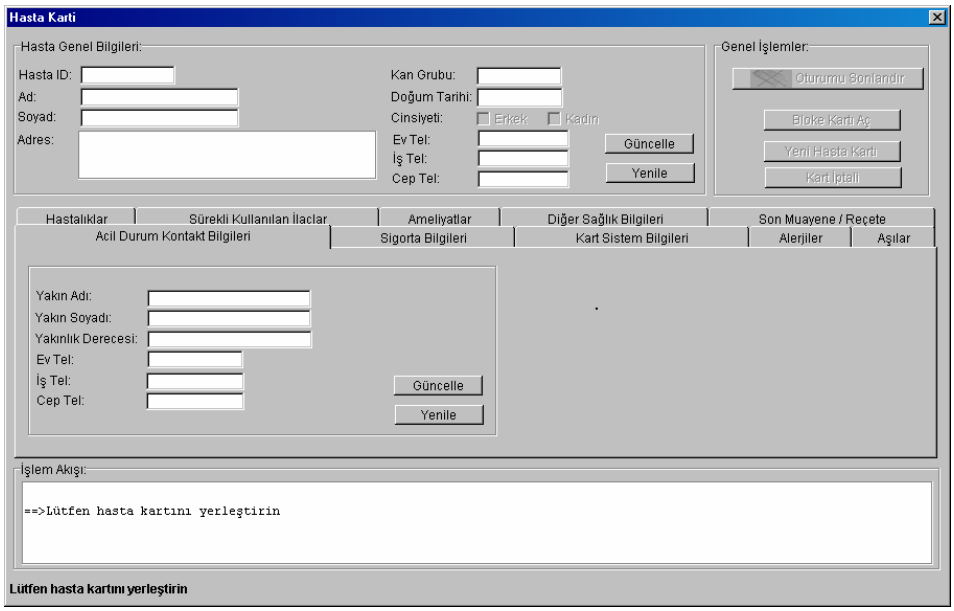
1.10 Kayıtlı Doktorlar Listesi



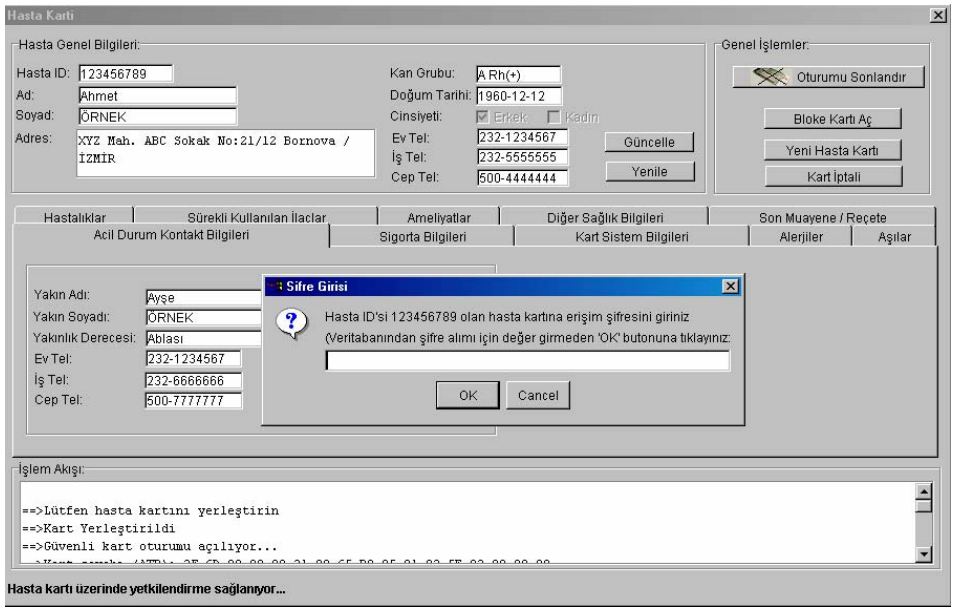
1.11 Kayıtlı Sağlık Bilgileri Listesi



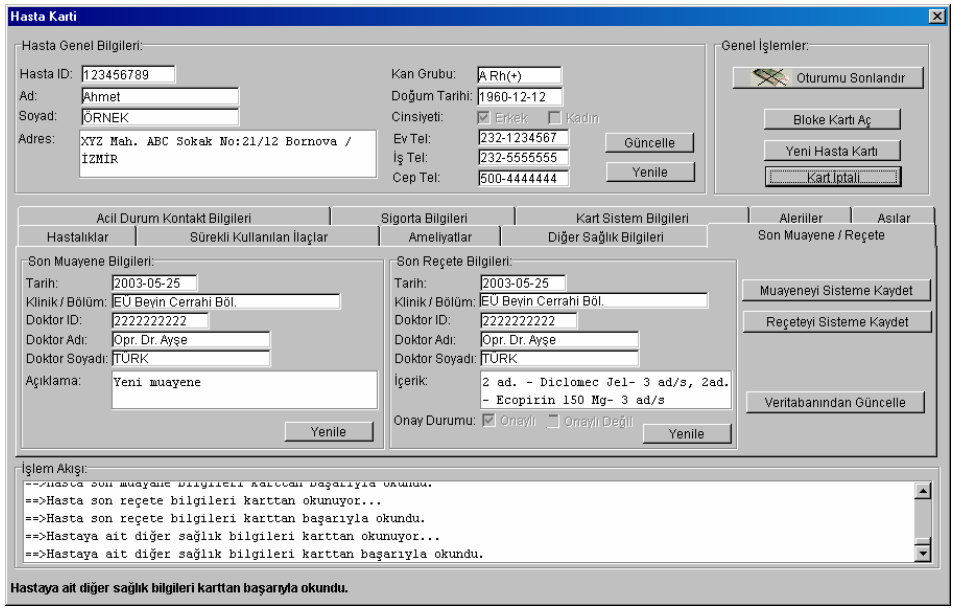
1.12 Hasta Akıllı Kartı İşlemleri Ara Yüzü



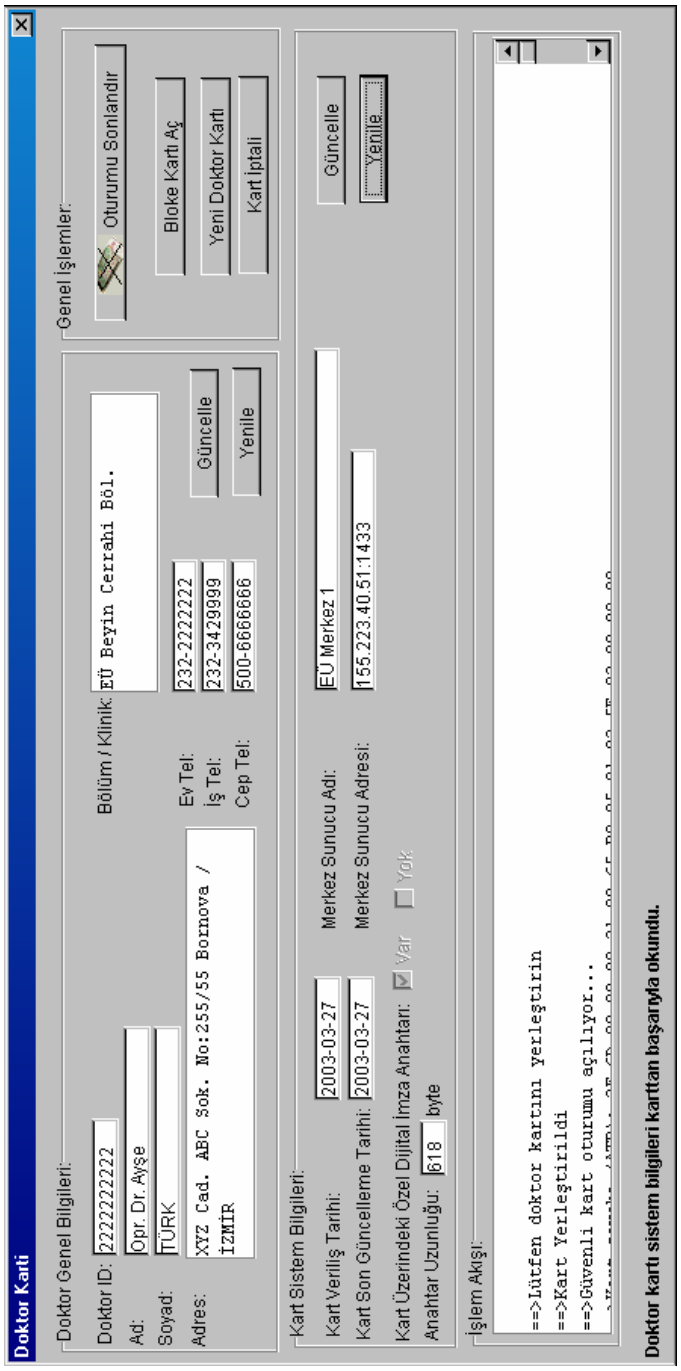
1.13 Hasta Akıllı Kartı Üzerinde Kullanıcı Yetkilendirmesi



1.14 Hasta Akıllı Kartı İçeriğinin Görüntülenmesi

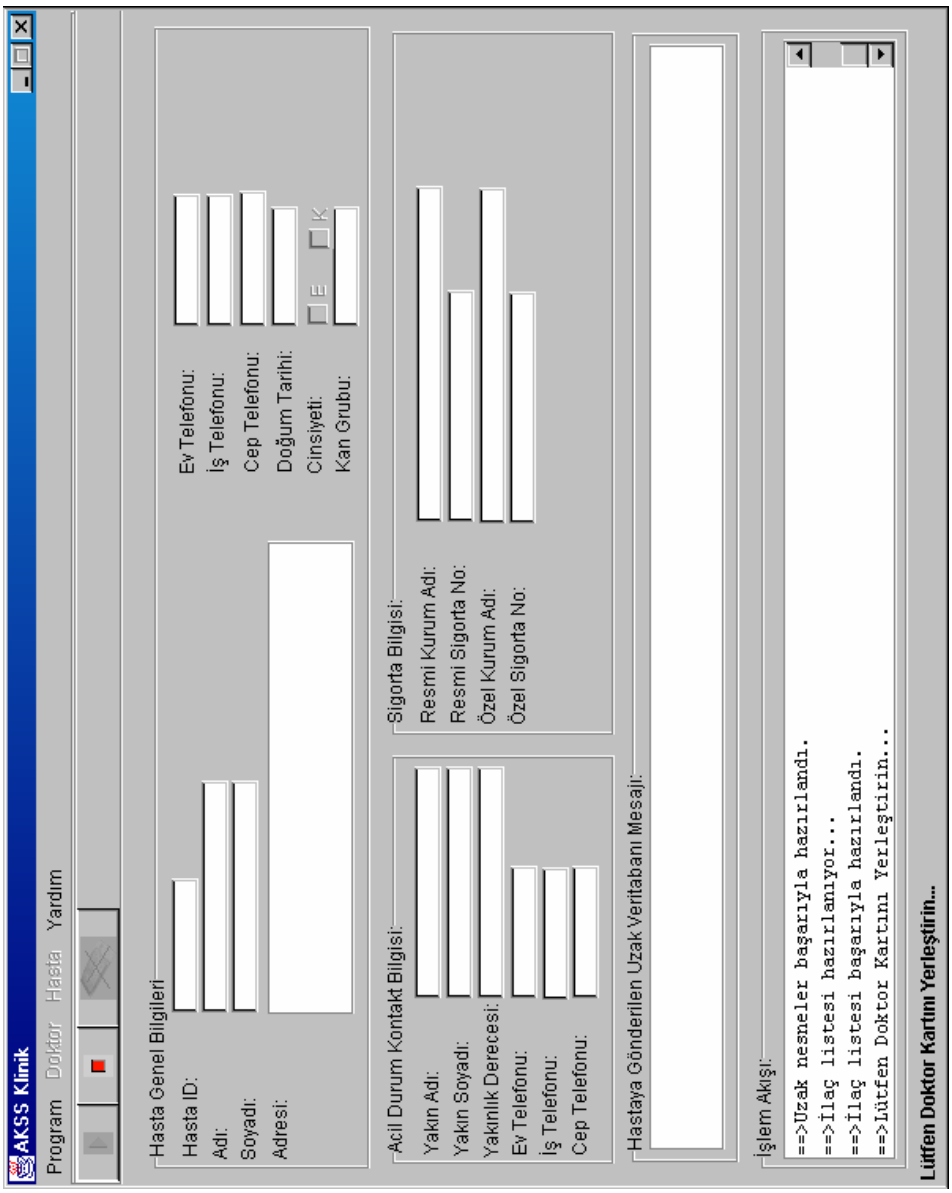


1.15 Doktor Akıllı Kartı İşlemleri Ara Yüzü

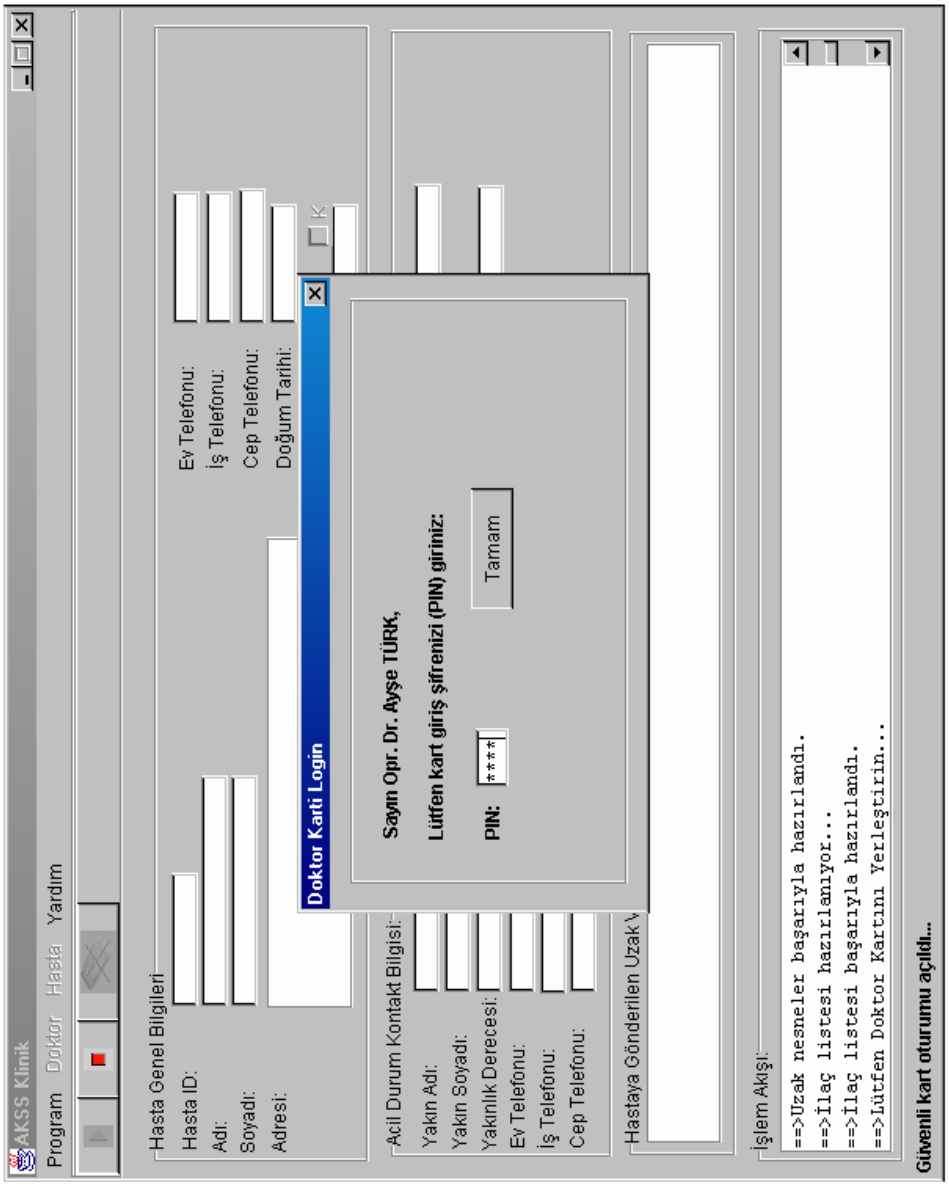


Ek 2 Klinik Yazılımı Ekran Görüntüleri

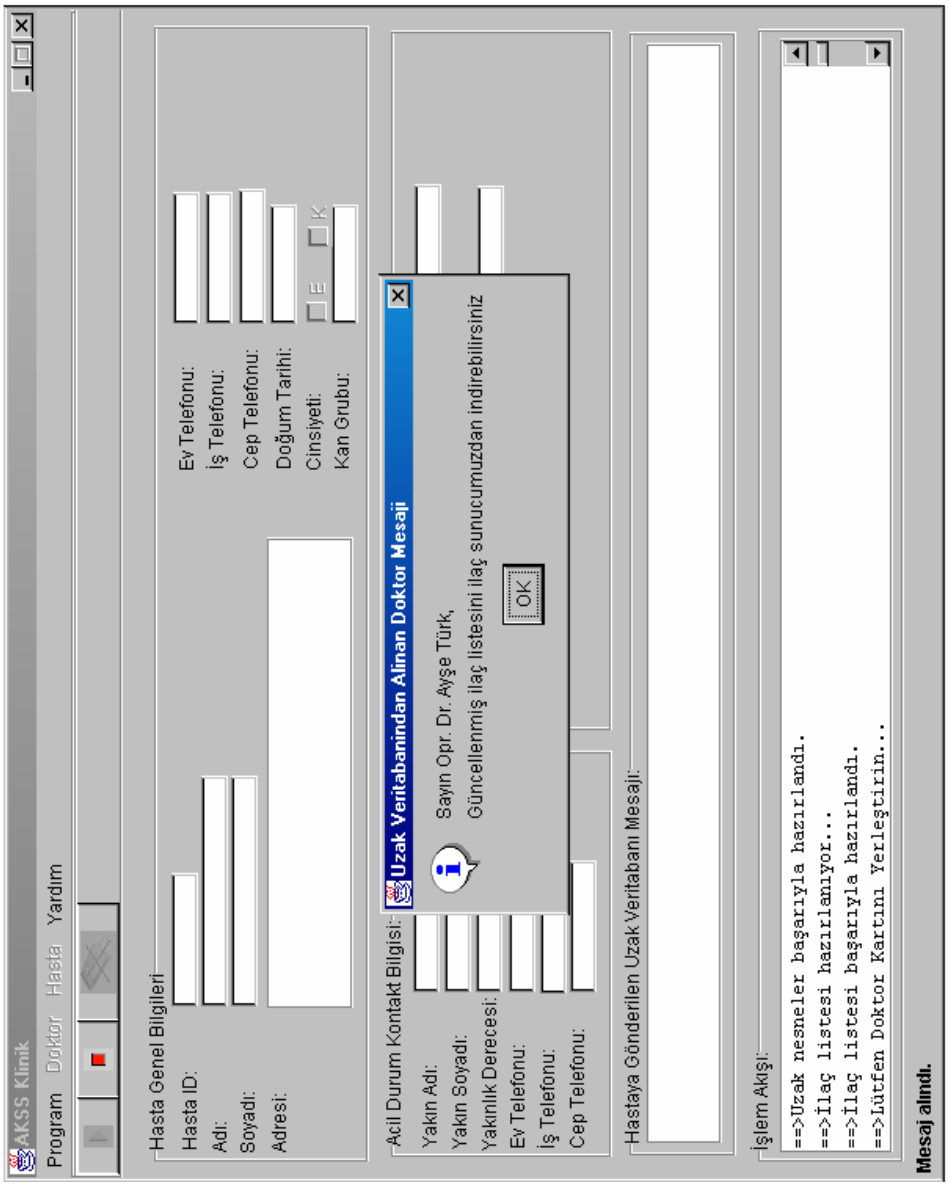
2.1 Klinik Uygulamasının Başlatılması



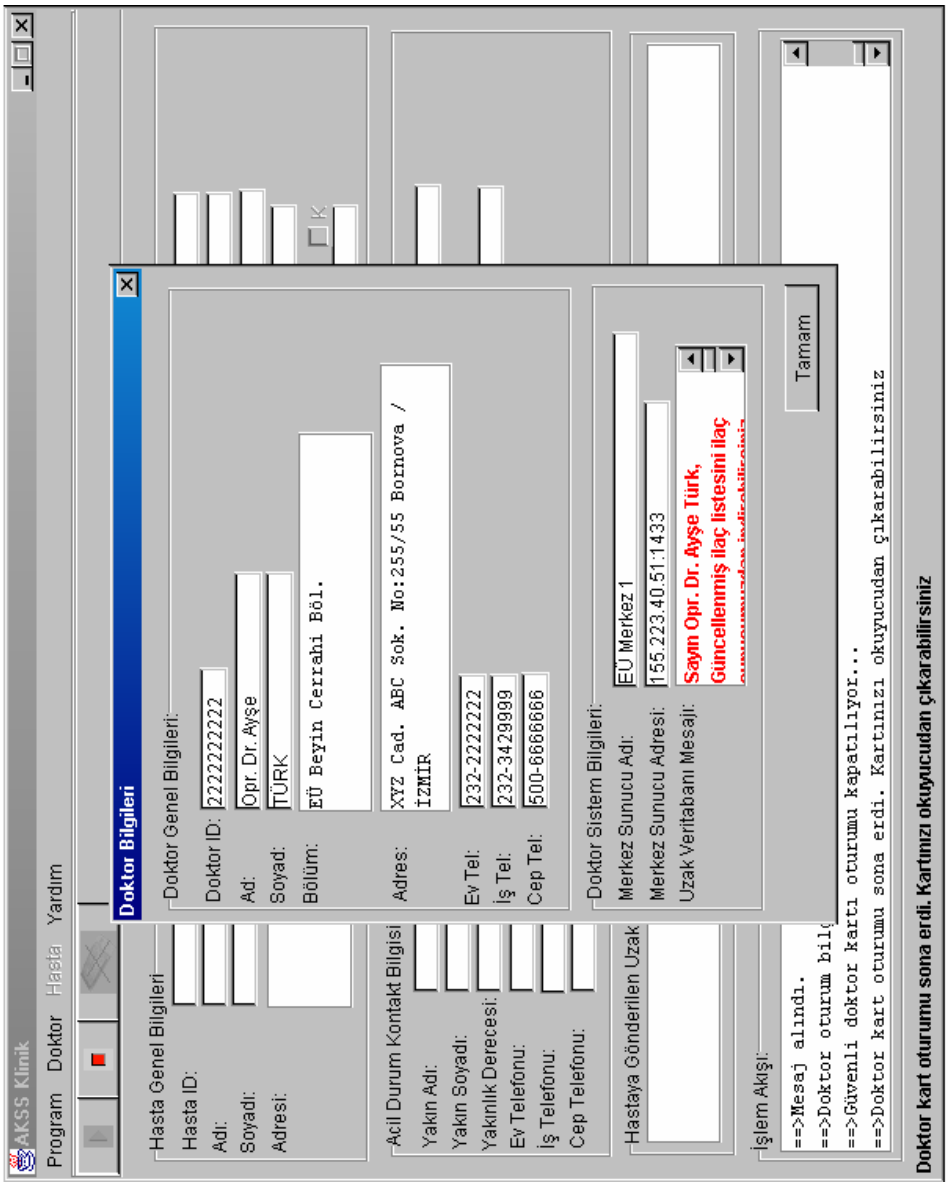
2.2 Uygulama Üzerinde Doktor Oturumunun Başlatılması



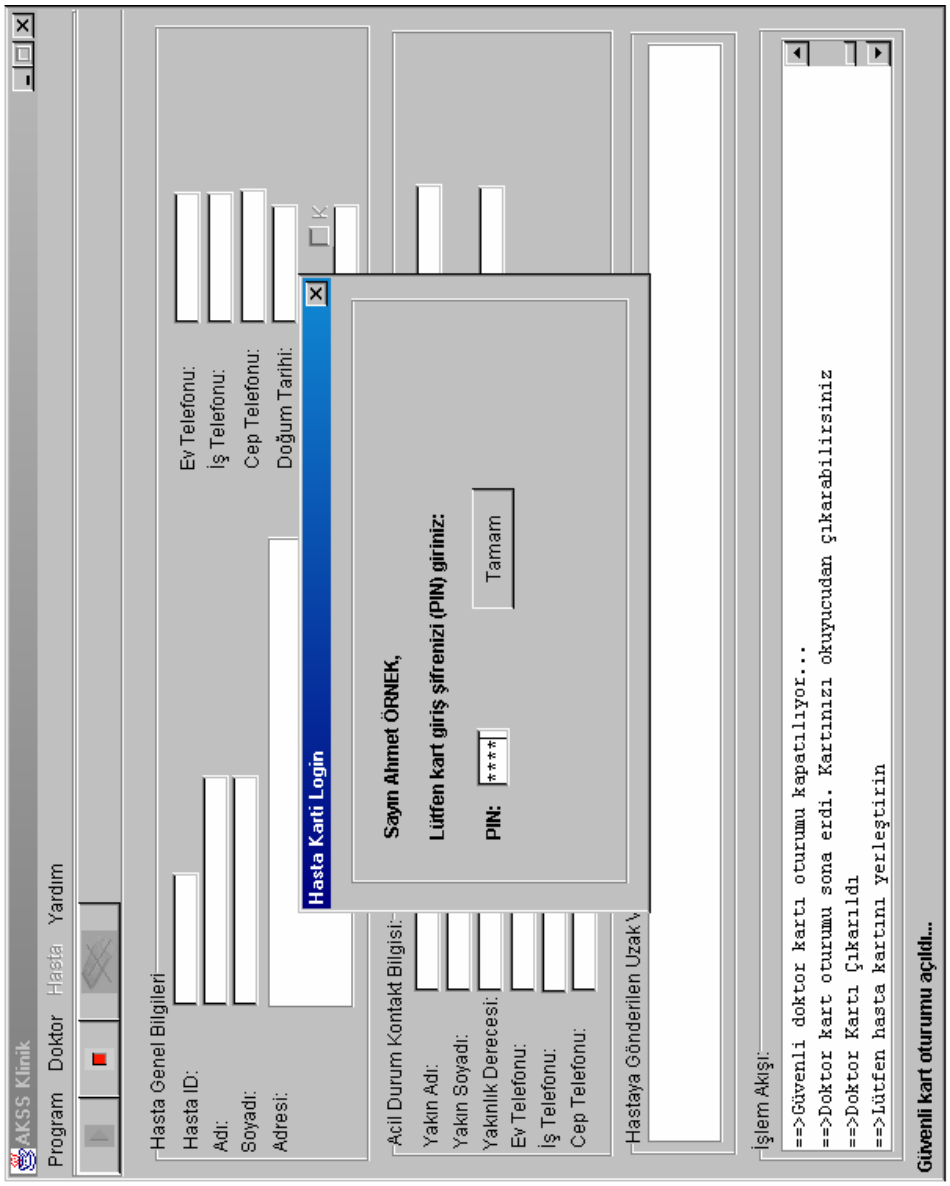
2.3 Doktor Oturumunun Açılması Sonrası Uzak Veritabanından Mesaj Alımı



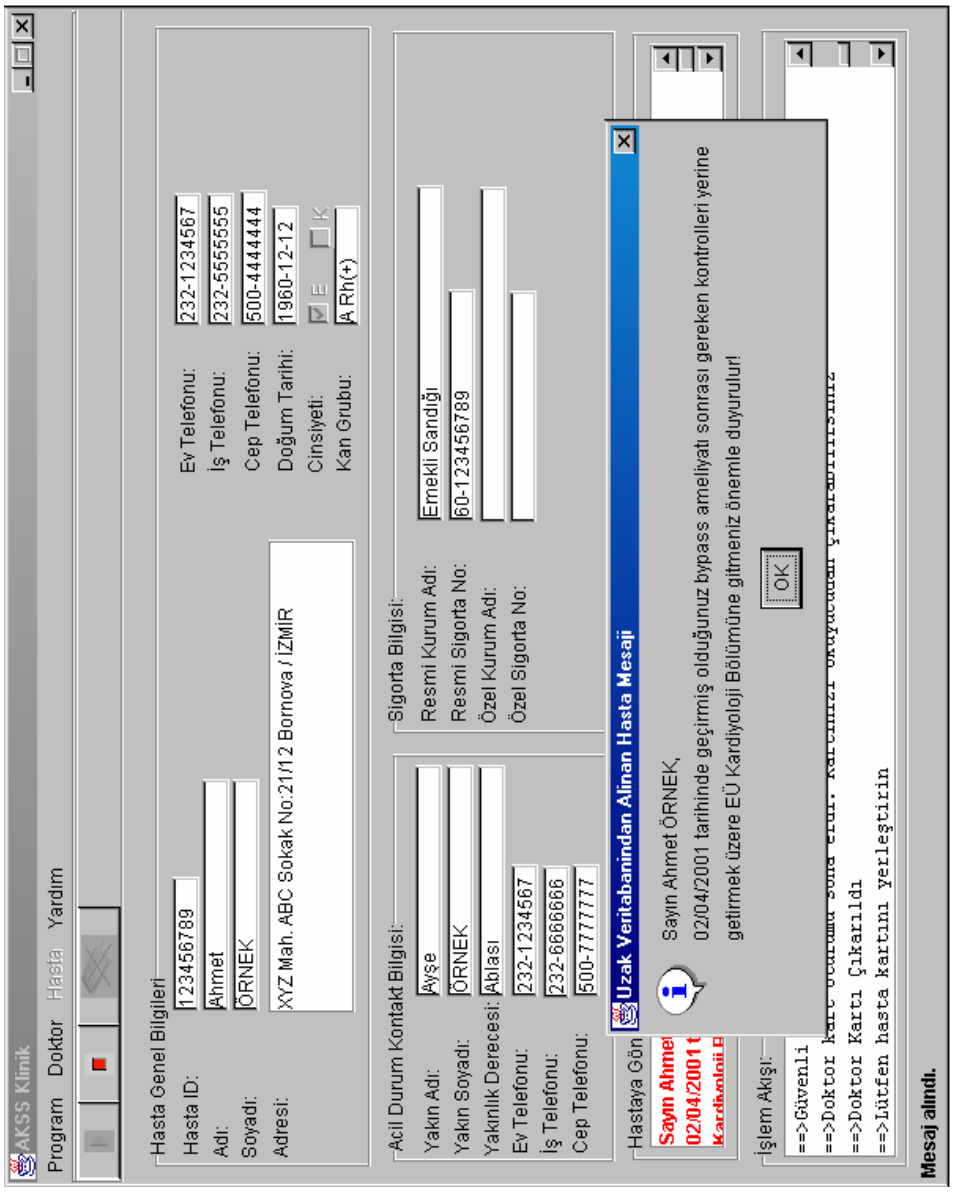
2.4 Doktor Oturum Bilgilerinin Görüntülenmesi



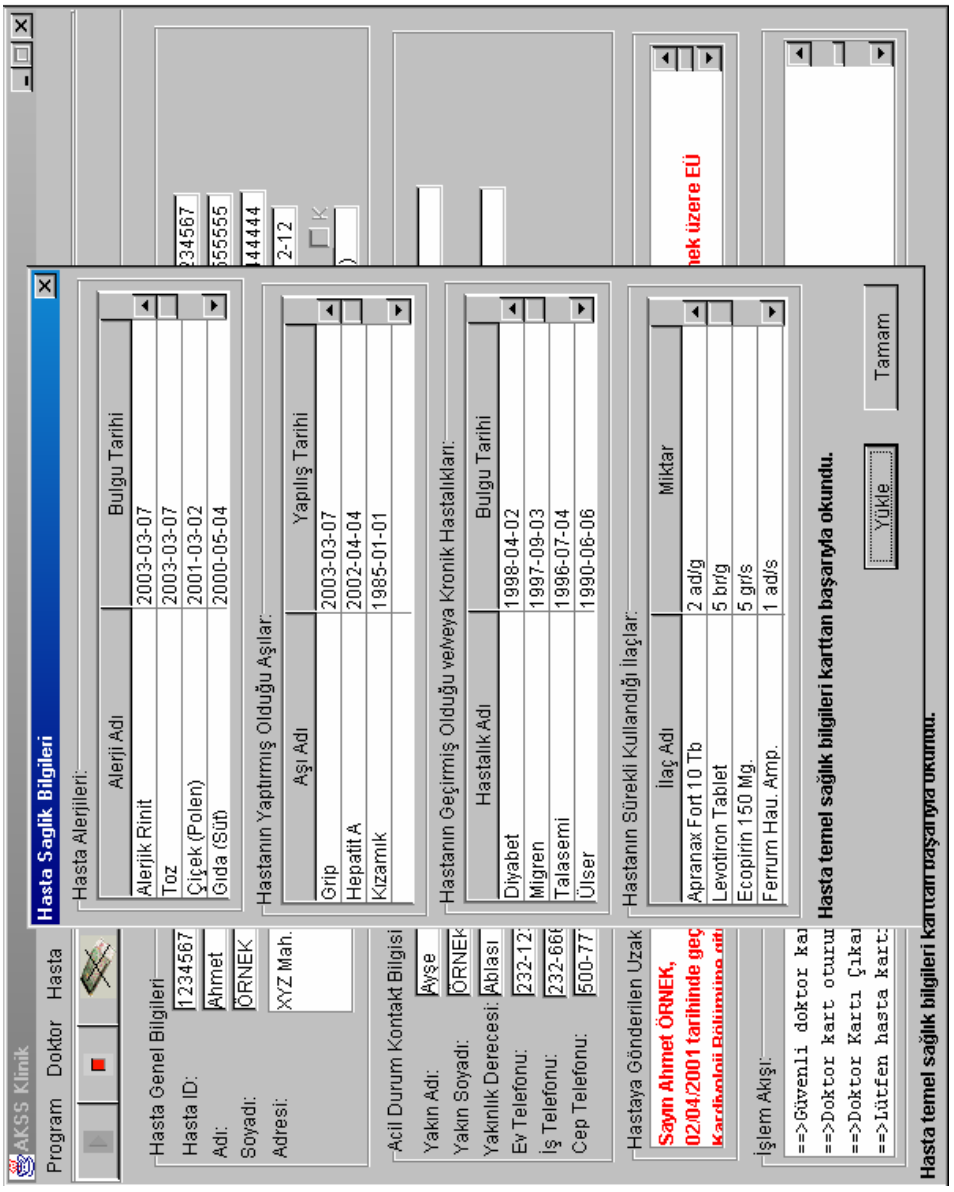
2.5 Uygulama Üzerinde Hasta Oturumunun Başlatılması



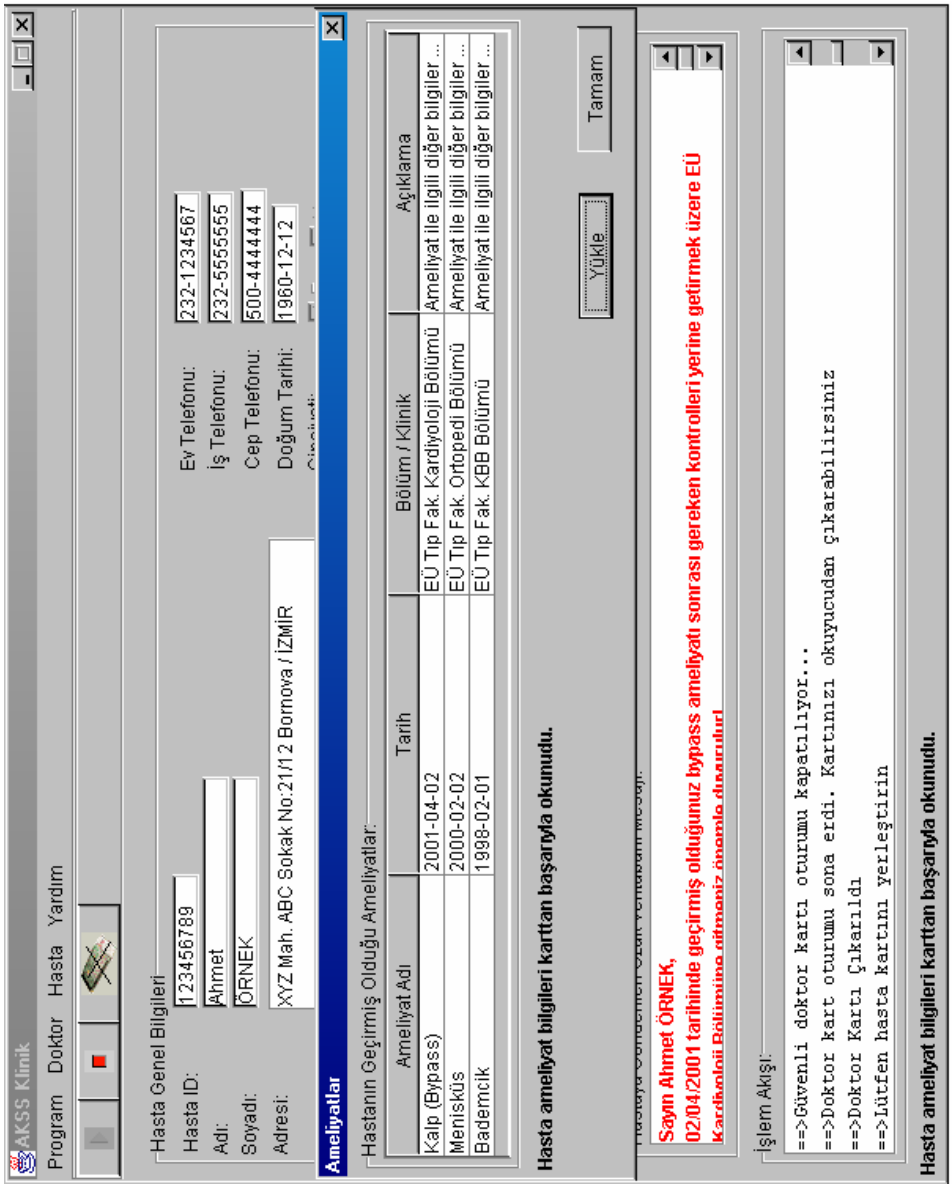
2.6 Hasta Oturumunun Açılması Sonrası Uzak Veritabanından Mesaj Alımı



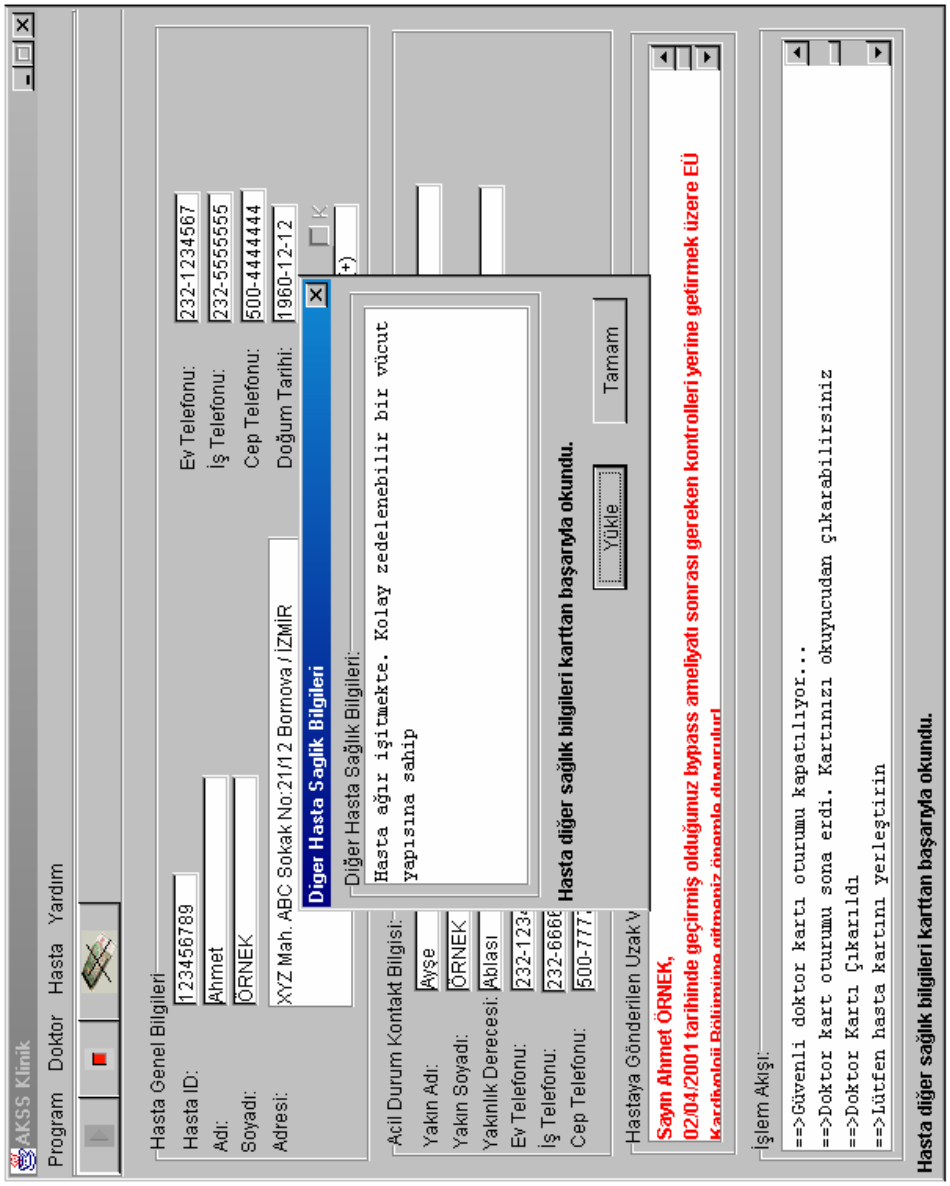
2.7 Hasta Kartında Yer Alan Temel Sağlık Bilgilerinin Görüntülenmesi



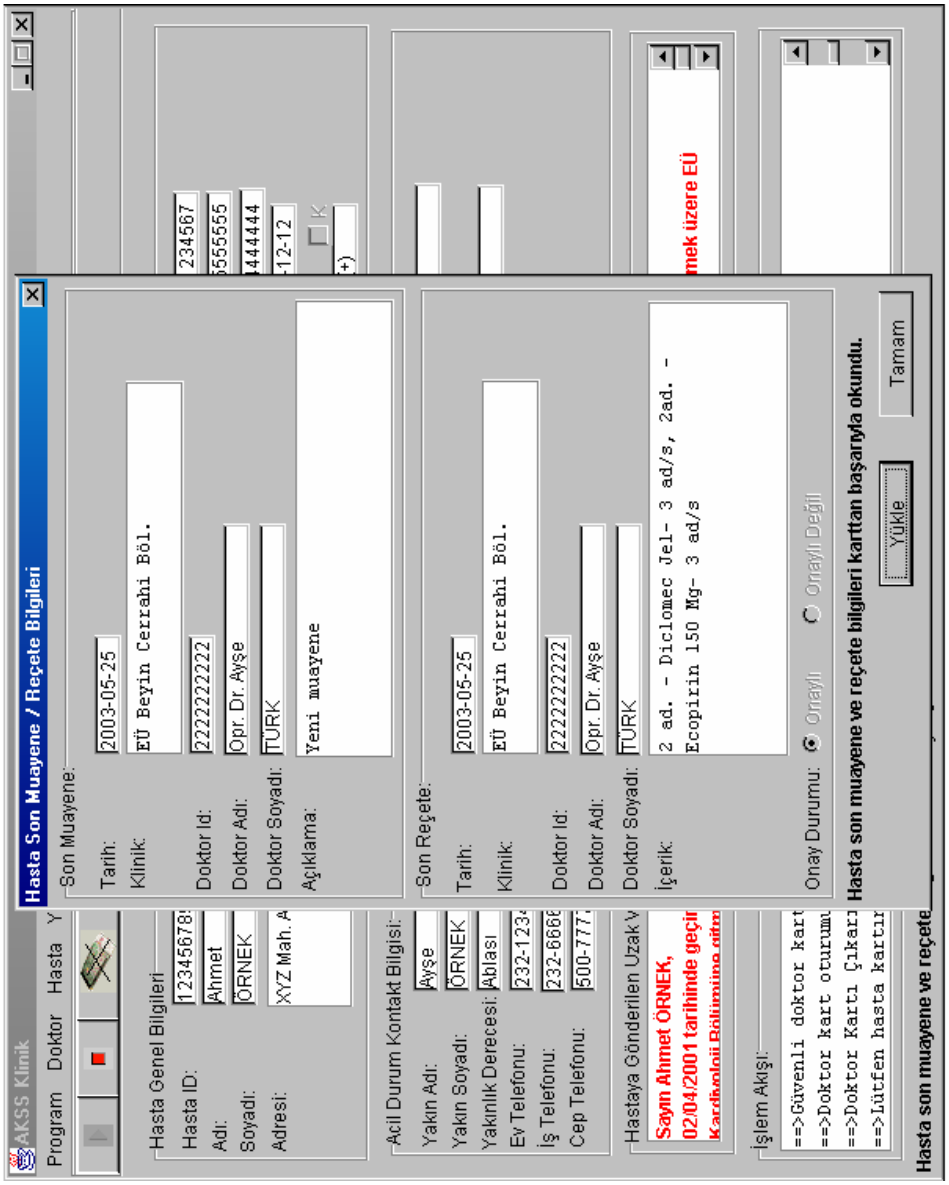
2.8 Hasta Kartında Yer Alan Ameliyat Bilgilerinin Görüntülenmesi



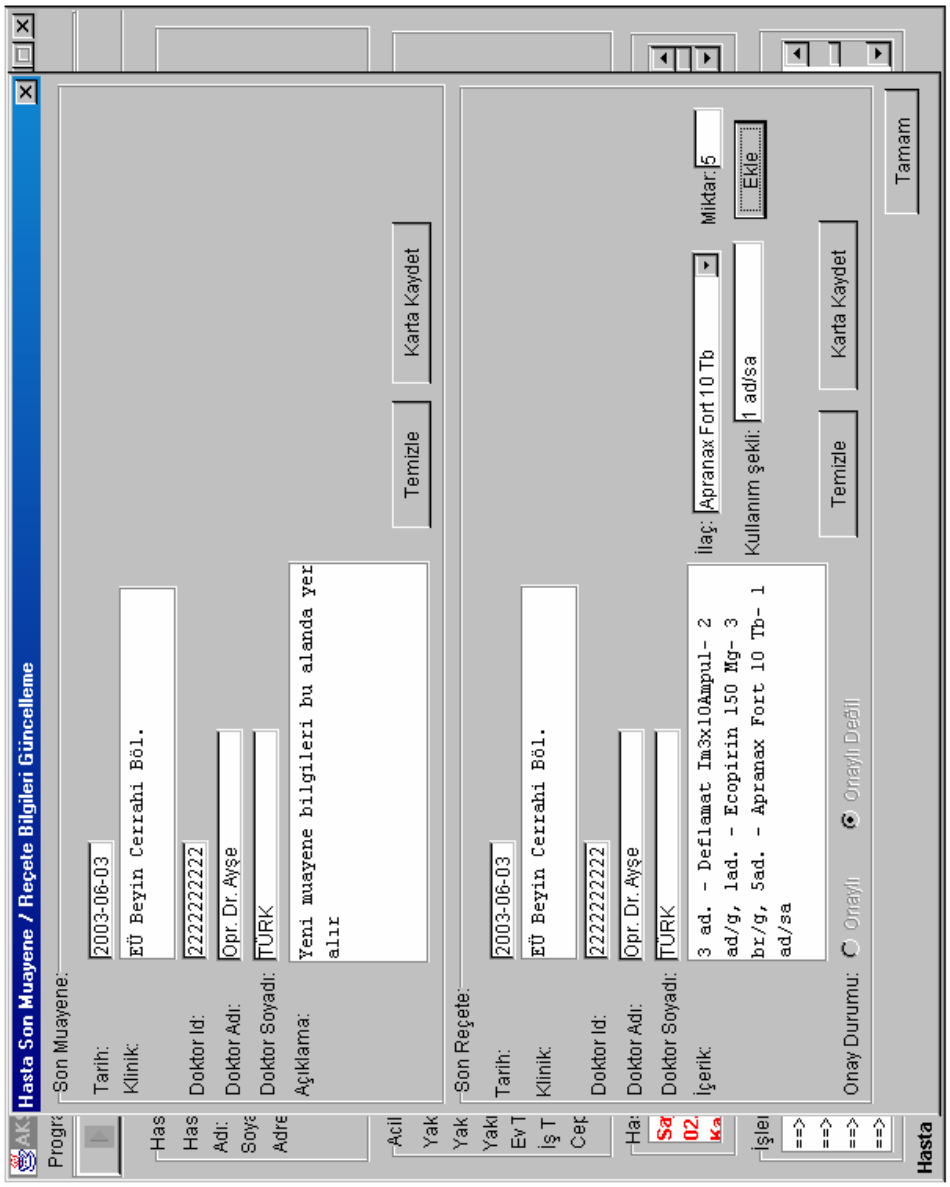
2.9 Diğer Hasta Bilgilerinin Görüntülenmesi



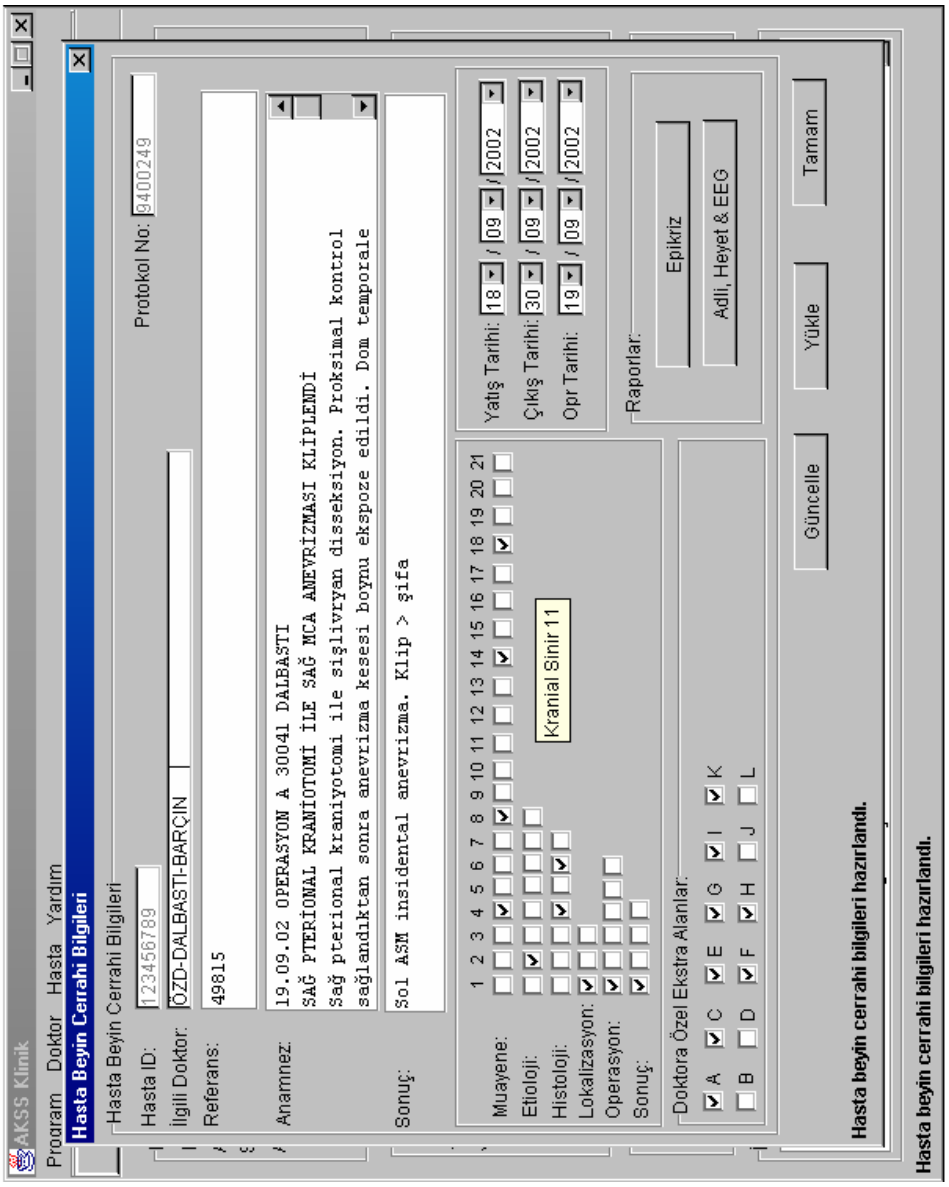
2.10 Hasta Kartında Yer Alan Son Muayene ve Reçete Bilgilerinin Görüntülenmesi



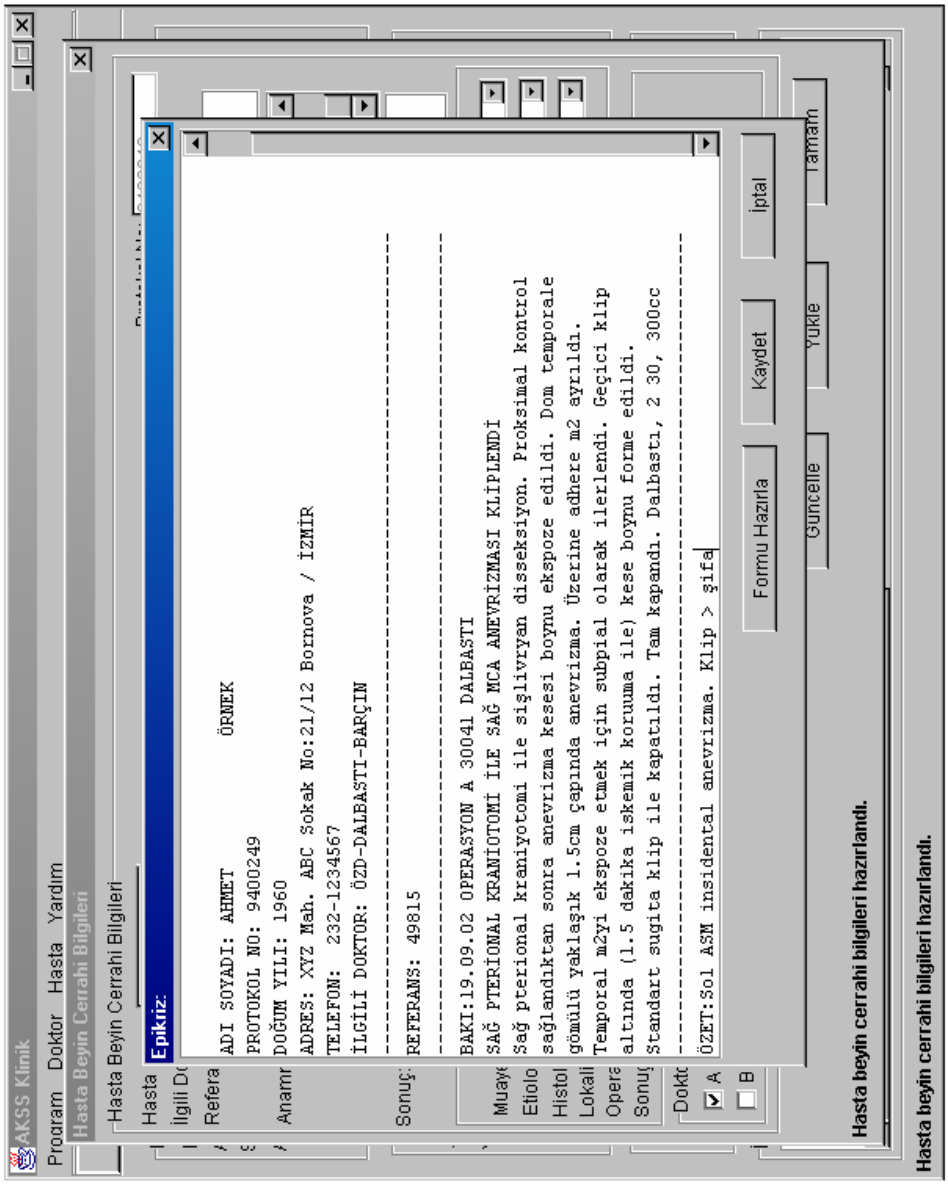
2.11 Hasta Kartına Son Muayene ve Reçete Bilgilerinin Kaydedilmesi



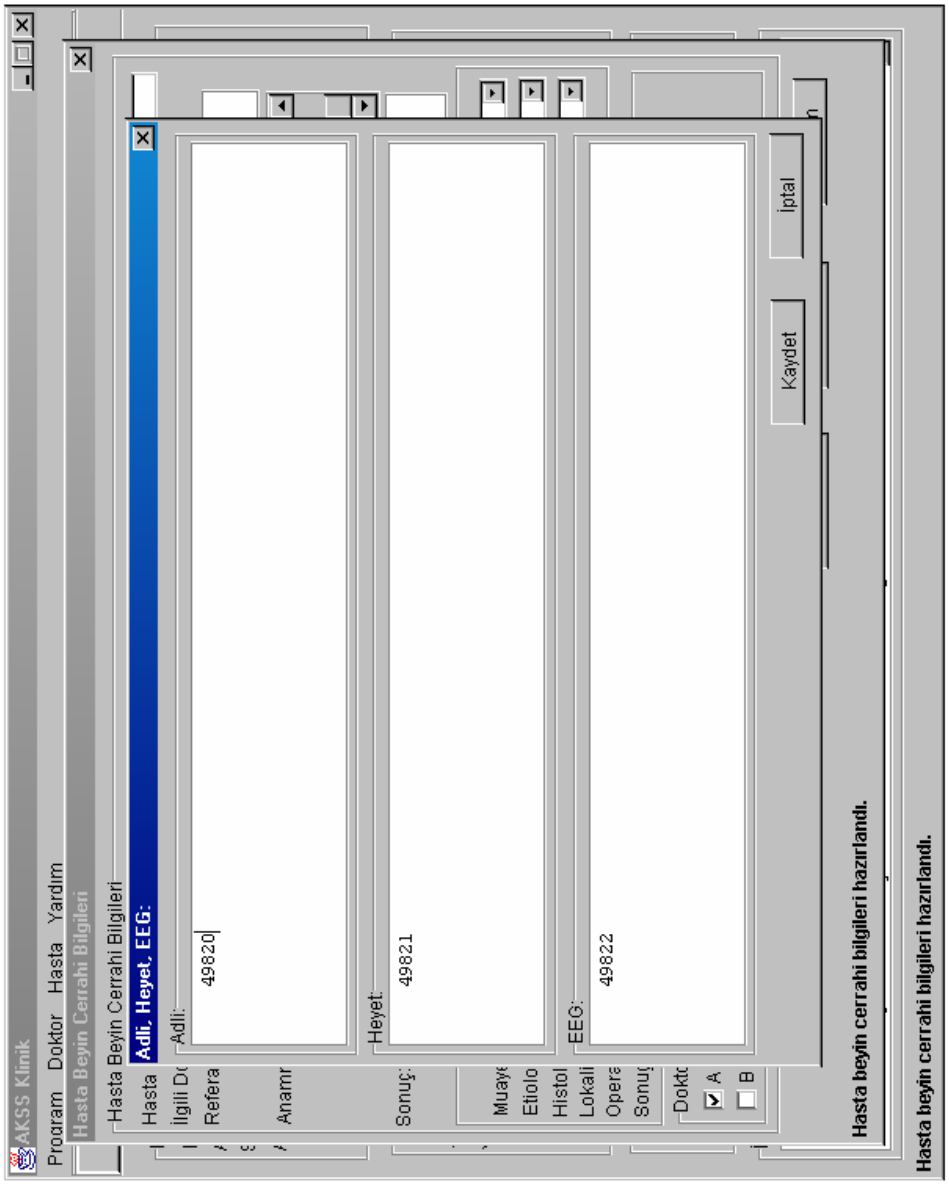
2.12 Kliniğe Özel Hasta Bilgilerinin Görüntülenmesi (1)



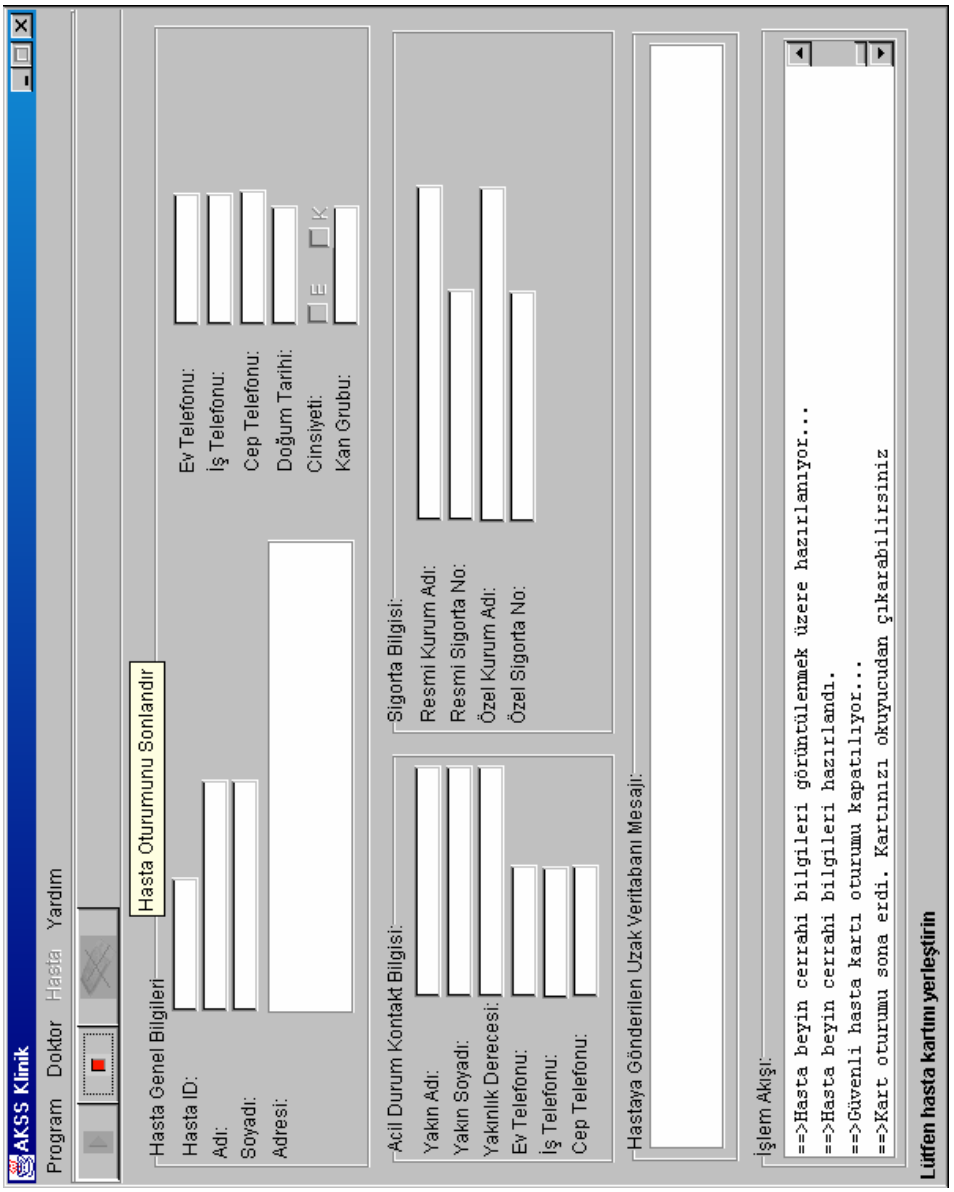
2.13 Kliniğe Özel Hasta Bilgilerinin Görüntülenmesi (2)



2.14 Kliniğe Özel Hasta Bilgilerinin Görüntülenmesi (3)



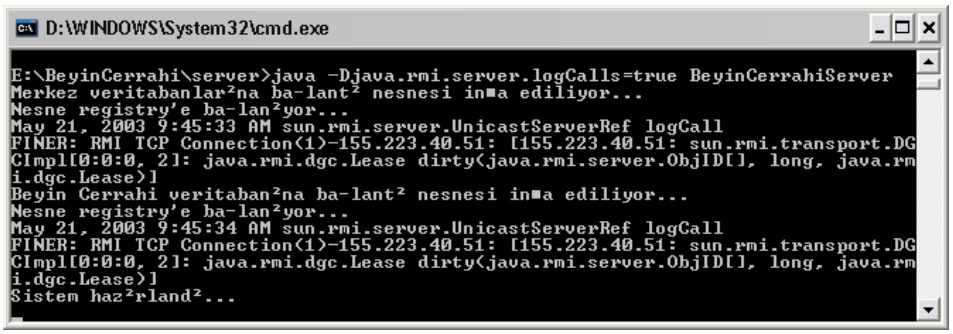
2.15 Hasta Oturumunun Sonlandırılması



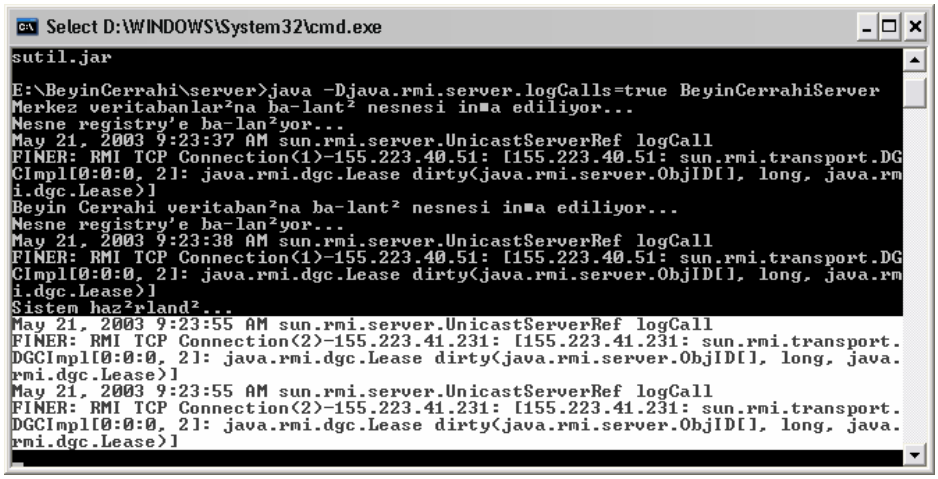
Ek 3 Sunucu Katmanı Yazılımı Ekran Görüntüleri

3.1 RMI Kayıtçısının RMI Sunucusu Üzerinde Çalıştırılması

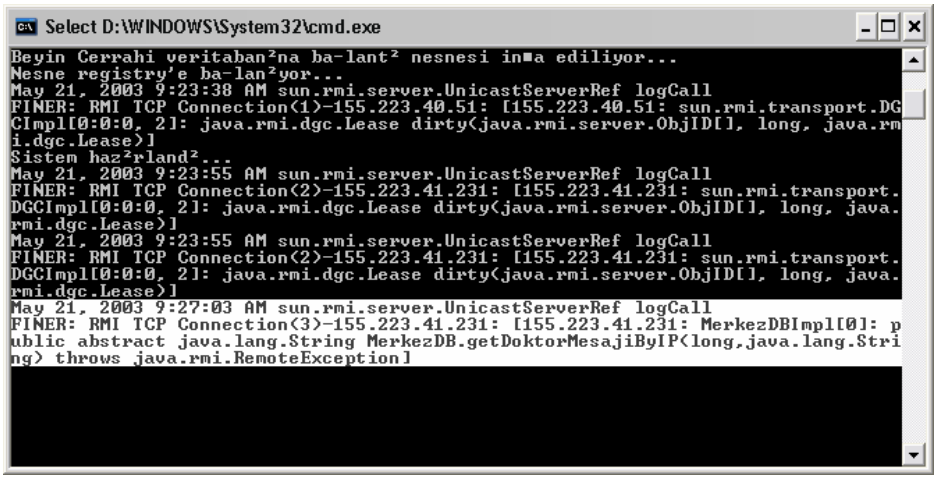
3.2 RMI Sunucu Uygulamasının Çalıştırılması



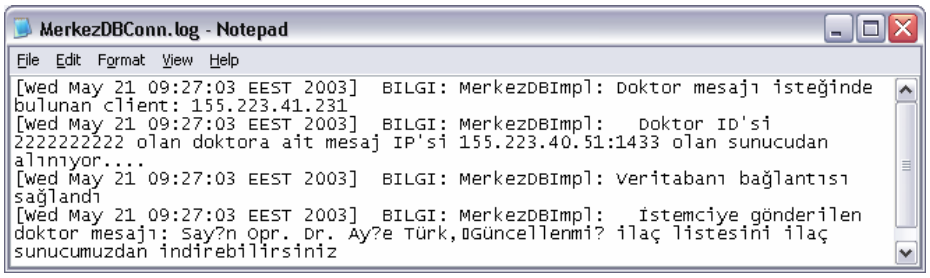
3.3 Bir RMI İstemcisinin Sunucu İle Bağlantıya Geçmesi Ve Uzak Nesnelere Referans Elde Etmesi



3.4 RMI İstemcisi Tarafından MerkezDBImpl Nesnesine Ait Metodun Çağrılması

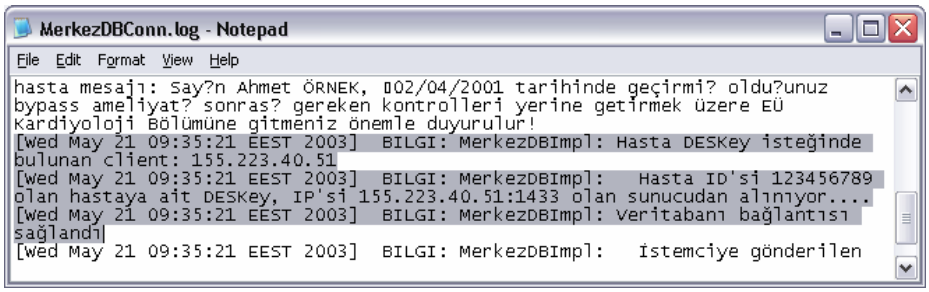


3.5 MerkezDBImpl Uzak Nesnesine Gelen İsteğin Ve Bu İsteği Karşılarken Yerine Getirilen İşlemlerin Kayıtları

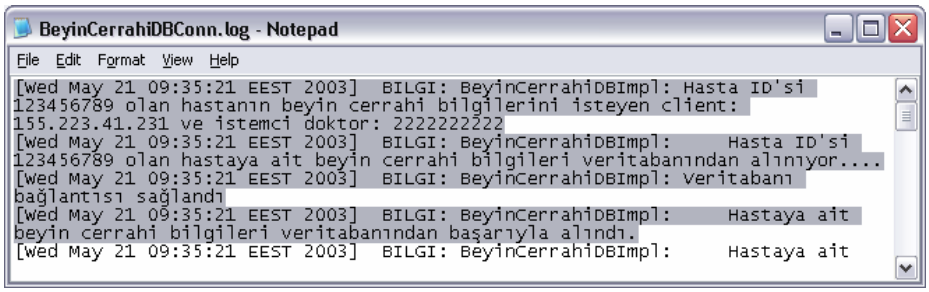


3.6 RMI İstemcisinin Hastaya Ait Şifreli Bilgileri İstemesi

3.7 Hasta Bilgilerinin İstemciye Gönderilmesi Sırasında MerkezDBImpl Tarafından Yerine Getirilen İşlemlere Ait Kayıtlar

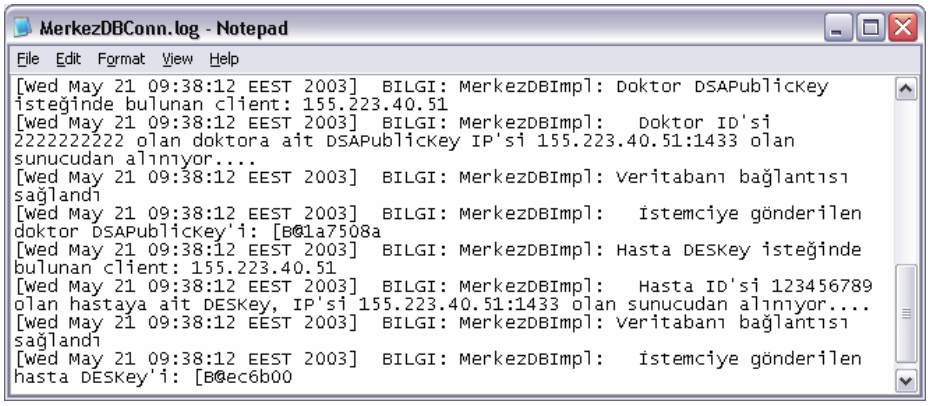


3.8 Hasta Bilgilerinin İstemciye Gönderilmesi Sırasında BeyinCerrahiDBImpl Tarafından Yerine Getirilen İşlemlere Ait Kayıtlar



3.9 RMI İstemcisinin Hastaya Ait Şifreli Bilgileri Güncelleme İsteği

3.10 Hasta Bilgilerinin Güncellenmesi Sırasında MerkezDBImpl Tarafından Yerine Getirilen İşlemlere Ait Kayıtlar

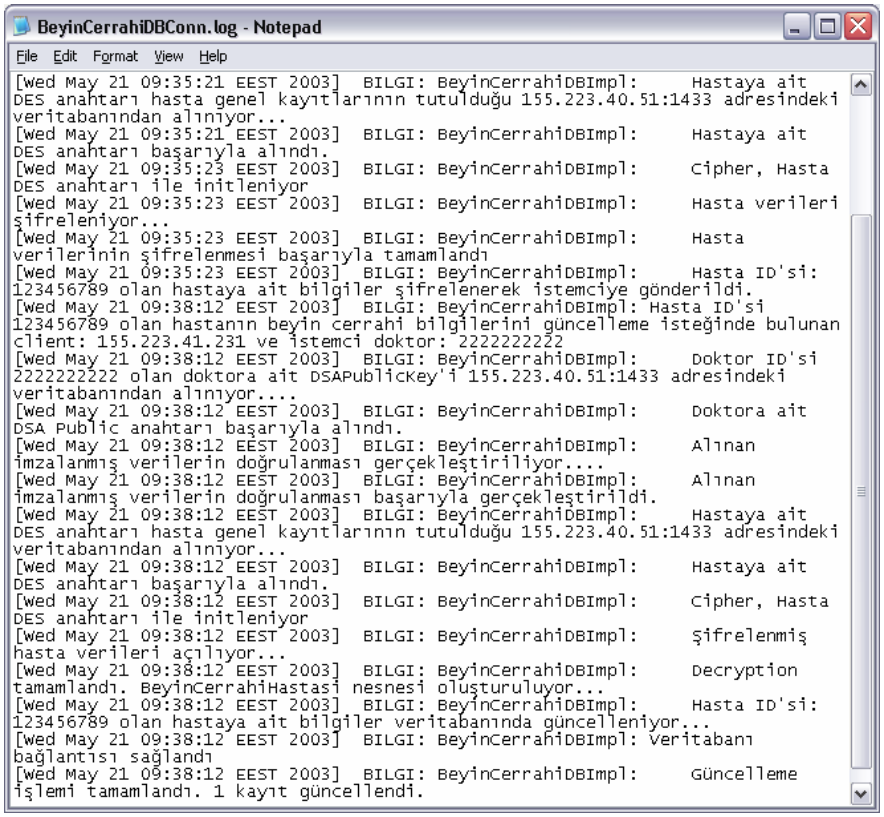


MerkezDBConn.log - Notepad

File Edit Format View Help

```
[Wed May 21 09:38:12 EEST 2003]  BILGI: MerkezDBImpl: Doktor DSAPublicKey
isteğinde bulunan client: 155.223.40.51
[Wed May 21 09:38:12 EEST 2003]  BILGI: MerkezDBImpl:   Doktor ID'si
2222222222 olan doktora ait DSAPublicKey IP'si 155.223.40.51:1433 olan
sunucudan alınıyor....
[Wed May 21 09:38:12 EEST 2003]  BILGI: MerkezDBImpl: Veritabanı bağlantısı
sağlandı
[Wed May 21 09:38:12 EEST 2003]  BILGI: MerkezDBImpl:   İstemciye gönderilen
doktor DSAPublicKey'i: [B@1a7508a
[Wed May 21 09:38:12 EEST 2003]  BILGI: MerkezDBImpl: Hasta DESKey isteğinde
bulunan client: 155.223.40.51
[Wed May 21 09:38:12 EEST 2003]  BILGI: MerkezDBImpl:   Hasta ID'si 123456789
olan hastaya ait DESKey, IP'si 155.223.40.51:1433 olan sunucudan alınıyor....
[Wed May 21 09:38:12 EEST 2003]  BILGI: MerkezDBImpl: Veritabanı bağlantısı
sağlandı
[Wed May 21 09:38:12 EEST 2003]  BILGI: MerkezDBImpl:   İstemciye gönderilen
hasta DESKey'i: [B@ec6b00
```

3.11 Hasta Bilgilerinin Güncellenmesi Sırasında BeyinCerrahiDBImpl Tarafından Yerine Getirilen İşlemlere Ait Kayıtlar



BeyinCerrahiDBConn.log - Notepad

File Edit Format View Help

```
[Wed May 21 09:35:21 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Hastaya ait
DES anahtarı hasta genel kayıtlarının tutulduğu 155.223.40.51:1433 adresindeki
veritabanından alınıyor...
[Wed May 21 09:35:21 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Hastaya ait
DES anahtarı başarıyla alındı.
[Wed May 21 09:35:23 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Cipher, Hasta
DES anahtarı ile initleniyor
[Wed May 21 09:35:23 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Hasta verileri
şifreleniyor...
[Wed May 21 09:35:23 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Hasta
verilerinin şifrelenmesi başarıyla tamamlandı
[Wed May 21 09:35:23 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Hasta ID'si:
123456789 olan hastaya ait bilgiler şifrelenerek istemciye gönderildi.
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl: Hasta ID'si
123456789 olan hastanın beyin cerrahi bilgilerini güncelleme isteğinde bulunan
client: 155.223.41.231 ve istemci doktor: 2222222222
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Doktor ID'si
2222222222 olan doktora ait DSAPublicKey'i 155.223.40.51:1433 adresindeki
veritabanından alınıyor....
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Doktora ait
DSA Public anahtarı başarıyla alındı.
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Alınan
imzalanmış verilerin doğrulanması gerçekleştiriliyor....
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Alınan
imzalanmış verilerin doğrulanması başarıyla gerçekleştirildi.
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Hastaya ait
DES anahtarı hasta genel kayıtlarının tutulduğu 155.223.40.51:1433 adresindeki
veritabanından alınıyor...
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Hastaya ait
DES anahtarı başarıyla alındı.
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Cipher, Hasta
DES anahtarı ile initleniyor
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Şifrelenmiş
hasta verileri açılıyor...
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Decryption
tamamlandı. BeyinCerrahiHastası nesnesi oluşturuluyor...
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Hasta ID'si:
123456789 olan hastaya ait bilgiler veritabanında güncelleniyor...
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl: Veritabanı
bağlantısı sağlandı
[Wed May 21 09:38:12 EEST 2003]  BILGI: BeyinCerrahiDBImpl:     Güncelleme
işlemi tamamlandı. 1 kayıt güncellendi.
```

Ek 4 İngilizce Terimler Sözlüğü

aracı	: agent
araç	: tool
artık toplama	: garbage collection
ayrık fabrika	: abstract factory
bayt akışı	: byte stream
baytkodu	: bytecode
bağlama	: link
bağlantı noktası	: port
belirteç	: identifier
birincil anahtar	: primary key
birleşik giriş kutusu	: combo box
birlikte işlerlik	: interoperability
cevap	: response
çekirdek	: core
çevirici	: converter
çevirici dili	: assembly language
çevrimiçi	: online
denetleyici	: controller
dizi	: array
durum sözcüğü	: status word
emir	: command
efendi / köle	: master / slave
ev sahibi	: host
görünüm	: view
gösterge	: pointer

ilklendirme	: initialization
isim sunucu	: name server
isim kayıtları	: naming registry
iskelet	: skeleton
iş parçacığı	: thread
kalıcı	: persistent
kalkan	: firewall
kaydetmek	: register
kodlama	: encoding
kurucu	: installer
küçük uygulama	: applet
küçük uygulama kalkanı	: applet firewall
miras	: inheritance
nesne serileştirme	: object serialization
olaya dayalı	: event driven
olay üreticisi	: event generator
on altılı	: hex
örnek	: instance
özel	: private
özel yazılım	: middleware
özellik	: attribute
parametre çözümleme	: parameter unmarshalling
parametre kodlama	: parameter marshalling
saklamak	: encapsulation
serileştirme	: serialization
soyut	: abstract
tek	: unique

tek iş parçacıklı	: single-threaded
tek örnek	: singleton
toplama	: aggregation
uygulama	: implementation
uygulamaya dayalı	: application driven
uzak doğrulama	: remote authentication
uzantı	: extend
yarı çift yönlü	: half-duplex
yeniden başlatma	: reset
yeniden kullanılabilirlik	: reusability
yerli	: native
yorumlayıcı	: interpreter
yuva	: slot

ÖZGEÇMİŞ

Geylani KARDAŞ

adres: Zafer Cad. No:24/18 Bornova / İZMİR
tel: (0232) 3425991
e-mail: geylani@bornova.ege.edu.tr

Kişisel Bilgiler

Milliyeti	: T.C.
Doğum Yeri / Tarihi	: Diyarbakır / 18.06.1979

Eğitim

2001 -	Ege Üniversitesi Uluslararası Bilgisayar Enstitüsü (UBE) *Bilgi Teknolojileri Ana Bilim Dalı*
1997 - 2001	Ege Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü
1993 - 1997	İzmir Bornova Anadolu Lisesi

Mesleki İlgi Alanları

Nesne tabanlı yazılım tasarımı
Nesneye dayalı yazılım geliştirme metodolojileri
Yazılım sistem mimarisi ve tasarım desenleri
Dağıtık sistemler
Gömülü sistemler