MOBİL TELEFONLAR İÇİN UYGULAMA GELİŞTİRME

Özer AKKOCA

Yüksek Lisans Tezi

Bilgisayar Mühendisliği Anabilim Dalı Bilişim Programı

Ağustos – 2007

JÜRİ VE ENSTİTÜ ONAYI

Özer Akkoca'nın "**Mobil Telefonlar için Uygulama Geliştirme"** başlıklı **Bilgisayar Mühendisliği** Anabilim Dalındaki, Yüksek Lisans Tezi 16.07.2007 tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı-Soyadı	İmza
Üye (Tez Danışmanı):	**Prof. Dr. YAŞAR HOŞCAN**	**……**
Üye	**: Yard. Doç. Dr. HAKAN KAĞNICIOĞLU**	**……**
Üye	**: Yard. Doç. Dr. YUSUF OYSAL**	**……**

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun …… tarih ve …… sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

MOBİL TELEFONLAR İÇİN UYGULAMA GELİŞTİRME

Özer AKKOCA

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı - Bilişim Programı

Danışman: Prof. Dr. Yaşar HOŞCAN
2007, 177 sayfa

Günümüzde mobil iletişim sektörü hızlı bir şekilde gelişmektedir. Mobil telefonların fiyatları gün geçtikçe düşmekte ve yetenekleri artmaktadır. Bir kişisel bilgisayarın yapabildiği birçok işi mobil telefonlar da yapabilmektedir.

Bu tezde, mobil cihazlar için program geliştirmede kullanılan J2ME (Java Micro Edition) platformu incelenmiş, özellikleri anlatılmış ve bu platform kullanılarak bir uygulama programı geliştirilmiştir. Geliştirilen uygulama programı ile mobil telefonun bir el terminali olarak kullanılması amaçlanmıştır. Bu uygulama programının el terminaline göre üstünlükleri ve eksiklikleri belirtilmiş ve kodları üzerinde çalışması anlatılmıştır. Programın çalışması ve kurulumu hakkında gerekli bilgiler verilmiştir.

Uygulama programı Netbeans programı kullanılarak geliştirilmiş, bazı simülatör programları ve mobil telefonlar kullanılarak test edilmiştir. Uygulama programı CD'de MobileApplication, Musteri, Konsol ve Servletler klasörlerindedir.

Anahtar Kelimeler: J2ME, CLDC, MIDP, Sipariş Takip Programı, El terminali.

ABSTRACT

Master of Science Thesis

APPLICATION DEVELOPING FOR MOBILE PHONES

Özer AKKOCA

Anadolu University
Graduate School of Sciences
Computer Engineering - Informatics

Supervisor: Prof. Dr. Yaşar HOŞCAN
2007, 177 pages

Mobile communication business has been growing quite fast at these days. Mobile phone prices drop down from day to day and their capabilities increase. Mobile phones can do many works as personal computers do.

In this thesis, using J2 ME platform which is used improving programs for mobile devices is examined, features are expressed and using this platform an application program is improved. With the application program, mobile phone is intended as a handheld terminal. Application program is compared to handheld terminal and specified advantages, disadvantages and how to work on codes. There is necessary information about how to set up and run program.

Application program is improved using Netbeans program and it is tested using some simulator programs and mobile phones. The application program is in MobileApplication, Musteri, Konsol and Servletler folders at CD.

Keywords: J2ME, CLDC, MIDP, Order Control Program, Handheld Terminal

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Sadece 10-15 yıllık bir geçmişi olmasına rağmen mobil telefonlar hayatımızın vazgeçilmez bir parçası olmuşlardır. İlk kullanılmaya başlanıldığı dönemlerde sadece sesli iletişim için kullanılan mobil telefonlar, teknolojinin gelişimine paralel olarak günümüzde standart bir kişisel bilgisayarın yapabildiği birçok işi yapabilir hale gelmiştir. İnternete bağlanmak, multimedya uygulamalarını çalıştırmak, fotoğraf çekmek, müzik dinlemek, oyun oynamak, ofis dokümanlarını hazırlamak bunlardan sadece birkaç tanesidir. Ayrıca bir cebe sığacak kadar küçük olması bu cihazların bilgisayarlara karşı en önemli avantajıdır.

Bu tez çalışmasında; ticari firmalardaki sipariş alma görevlilerinin (plasiyerlerin) kullandıkları el terminali cihazının yerine mobil telefonların kullanımı için gerekli olan yazılım hazırlanmıştır. Çalışmada, plasiyerlerin sipariş alma işlemini mobil telefon ile firmanın stok veritabanına bağlanarak sağlıklı bir şekilde gerçekleştirmesi amaçlanmıştır.

El terminali yerine mobil telefon kullanma fikri, firmalara birçok avantaj sağlamaktadır. El terminalleri mobil telefonlara göre oldukça pahalı cihazlardır. Ayrıca ülkemizdeki firmaların büyük bir kısmının kullandığı el terminalleri internet bağlantısını desteklememektedir. Bu nedenle stok kontrolü yapılmaksızın sipariş alma işlemi yapılmakta ve böylelikle istenmeyen durumlar ortaya çıkmaktadır.

Çalışmada hazırlanan yazılım, Java programlama dilinin mobil cihazlar için olan Java 2 Micro Edition (J2ME) sürümü kullanılarak hazırlanmıştır. Yazılımın hazırlanması aşamasında mobil telefon yerine NetBeans programının emülatörü, ve test aşamasında giriş seviyesinde bir mobil telefon olan Nokia 6070 kullanılmıştır.

2. J2ME ORTAMININ TANITILMASI

2.1. Java Dilinin Kısa Bir Tanıtımı

Java programlama dili bilgisayar dünyasında önemli bir yer teşkil etmektedir. Nesne tabanlı programlama özelliği, yazılan programın değişikliğe uğraması ya da yeniden derlenmesi gerekmeden farklı yapıdaki bilgisayar sistemlerinde kullanılması gibi özellikler Java dilinin popülerliğini sağlayan önemli özelliklerin sadece birkaçıdır. İlk geliştirilme amacı SUN Microsystem şirketinin içinde ortak bir dil kullanılması olan ve o zamanki adı OAK olan Java dili, kısa zamanda dünya çapında yoğun olarak kullanıma girmiştir. Javanın kullanıldığı yerler, basit bir ev bilgisayarlarından, en karmaşık sistem sunuculara kadar değişen bir yelpazede yer almıştır.

2.1.1. Java mimarisi

Java mimarisi ilk yayımlandığı günden bu yana pek değişmemiştir. Java derleyicisi Java programlama dilini "bytecode" kümelerine çevirmektedir. Bytecodelar, "sanal makina" olarak bilinen soyut hesap makinası komutlarıdır. Java Sanal Makinası (JVM), Java programını çalıştırabilmek için java kodlarını yorumlar. Java bytecode uygulamaları, Sun Microsystems tarafından verilmiş belirtimleri (Java Virtual Machine Specification (JVMS)) izlemek zorundadır.

Java dilinin doğasından gelen en güçlü yönlerinden biri taşınabilirliğidir. Java programı alınır ve tekrar derlenmesine gerek kalmadan değişik işletim sistemleri üzerinde çalıştırılabilir. Bu taşınabilirliği pek çok yolla sağlayabilir. Java belirtimleri, tiplerin ve bytecodeların altta çalışan işletim sisteminden bağımsız olduğunu garanti ederler.

Java, güvenlik altyapısıyla da bilinmektedir. Java çalışma zamanı ortamı Java programlarını "sandbox" içinde çalıştırır. Bu "sandbox"lar programın çalışacağı bilgisayarın kaynaklarının kısıtlı bir kümesine ulaşırlar. Java programları "sandbox"lar içine hapsedilirler, böylece kötü programlar programa ulaşamazlar veya programın çalışacağı bilgisayara zarar veremezler. Bu özellik

dağıtık veya web-tabanlı uygulamaların hem sunucu hem de istemci taraflarında gereklidir.

Java dilini bir diğer önemli özelliği nesneye dayalı olmasıdır. Java'daki nesne modelinin genişletilmesi oldukça kolaydır. Çoğu nesneye dayalı diller sert, yönetilmesi zor nesne hiyerarşileri seçerler veya performansı ve çok yönlülüğü arttırmak için tamamen dinamik nesne modelleri kullanırlar. Java ise dengeyi sağlamıştır, gerektiği yerde kullanılan dinamik arayüz modeli ile basit bir sınıf mekanizması sağlar.

Java dili basittir. Java dili eğer başka bir nesneye dayalı programlama diline aşinaysanız basittir. Java'nın sözdizimi öğrenilmesi kolaydır. Java dilinin sözdizimi C++ ile benzerdir ve diğer programlama dillerinde bulunan yararlı özellikleri içerir [1].

2.1.2. Java 2

Tüm programlama dilleri ve geliştirme ortamları gibi Java da ilk çıktığı günden bu yana gelişmiştir. Java'nın ilk çıkışından itibaren pek çok özellik ve yetenek Java'ya eklenmiştir. Java 1.2 deki ilerlemeler üzerine Sun Microsystems Java'yı yeniden tanımladı ve Java'da temel değişikler yapmak üzere Java'yı lisansladı. Java 1.2 Java 2 oldu ve JDK ve JRE versiyonları 1.2 de kaldı. Daha önemlisi Java platformu 3 sürüme ayrıldı.

- Java 2 Standart Edition (J2SE)
- Java 2 Enterprise Edition (J2EE)
- Java 2 Micro Edition (J2ME)

Her sürüm Java tabanlı uygulamaları geliştirmek için Java sanal makinesini ve çalışma zamanı sınıflarını içeren eksiksiz bir ortam sunar. Üç sürüm de değişik cihazlar üzerinde çalışabilen uygulamaları hedeflerler. Masaüstü uygulamaları için gerekli kullanıcı arayüz sınıflarını sunan J2SE kullanılır. Bileşen tabanlı programlamayı vurgulayan J2EE ise sunucu tabanlı uygulamalarda tercih edilir. Gömülü ve el cihazları içinse hedef J2ME'dir.

Bir sürümü diğerinden ayıran, her sürümün tanımladığı sınıf kütüphaneleridir. J2ME'yi J2SE nin alt kümesi olarak ve J2SE'yi de J2EE'nin alt

kümesi olarak düşünülebilir. Bytecode'ların gerektirdiği, üç sürümde de bulunan sınıfları sağlayarak Java bytecode'unu her sürümde çalıştırmak mümkündür. J2ME tabanlı cihazlar J2SE ve J2EE'nin sağladığı sınıflardan çok daha az sınıfa sahiptir.

2.1.2.1. Java 2 Standart Edition (J2SE)

Java 2 Standart Edition, temel Java ortamıdır. Bu ortam, çekirdek Java sınıflarını ve API'lerini web browserlar üzerinde çalışabilecek uygulamalar da dahil standart istemci - sunucu uygulamalarını geliştirme ve çalıştırma olanağı sağlar.

2.1.2.2. Java 2 Enterprise Edition (J2EE)

Java 2 Enterprise Edition farklı Java API'lerini ve Java olmayan teknolojileri birleştirir. Genellikle çok katlı ve dağıtık uygulamaları geliştirmek için kullanılır. J2EE teknolojileri, günümüzün büyük çok katlı, heterojen uygulamalarını bir arada sunar. J2EE sıklıkla orta-katman veya sunucu taraflı teknolojileri tanımlamak için kullanılır. Gerçekte, J2EE bilgi sistemlerinin tüm katmanlarında kullanılan teknolojileri içerir. Örneğin JDBC istemci Java appletlerinden, orta katmanlı Java servleti veya Enterprise Java Bean tarafından verilere ulaşmak için kullanılır.

2.2. Java 2 Micro Edition (J2ME)

Java 2 Micro Edition veya J2ME, Java yazılımlarını elektronik ve gömülü cihazlara yerleştirebilmek için tasarlanmış geliştirme ve çalıştırma ortamıdır. Diğer daha büyük Java sürümleri gibi, Java 2 Micro Edition'ın amacı ürünler arasındaki uyumluluk, kodun taşınabilirliği, güvenli ağ yapısı ve ölçeklenebilirliğin arttırılması gibi Java teknolojisinin özelliklerinin sürdürebilmektir. Hayatta pek çok şeyde olduğu gibi bir boyut her şeye uymaz. Doğal olarak mainframelerden mobil telefonlara kadar her şeye uyan bir Java

platformu pek pratik olmaz. J2ME Java dilini kişisel bilgi, iletişim ve hesap makinelerine yerleştirmiştir. Genellikle bu cihazlar geleneksel bilgisayarlardan daha küçük ve daha az güçlüdürler.

J2ME'nin geliştirilmesine Sun tarafından başlanılmıştır, fakat günümüzde dünyanın en büyük elektronik ve gömülü cihaz üreticileri tarafından desteklenmektedir. Dünyanın mobil ve kablosuz teknoloji satıcıları J2ME teknolojisini inceliyorlar veya aktif olarak katılıyorlar veya yarışan ürünler üzerinde çalışıyorlar. Bu destekleyiciler Sun tarafından geliştirilen J2ME'yi standartlaştırmak için bir topluluk oluşumu başlattılar. Bu oluşum Java Community Process olarak adlandırıldı ve J2ME'nin ilerlemesinde önemli rol oynadı.

J2ME veya diğer Java uygulamalarının çalıştığı platformlara genel olarak "cihaz" denilir. J2ME için bu cihazlar "küçük cihazlar " başlığı altında toplanır. Bu cihazları başka sözcüklerle de ifade edebiliriz: bilgi cihazları, tüketici elektronikleri, gömülü cihazlar gibi. Bu "küçük cihazların" geniş çeşitlerini tanımlamak önemlidir ve asıl önemli olan bu küçük cihazların J2ME'nin hedef cihazları olduğudur.

Java'nın J2ME ile küçük cihazlar için programlama dili ve yazılım platformu olarak yeniden doğumu, yakın gelecekte bilgisayar sistemlerinin sayısını aşacaktır. Genç bir teknoloji olarak J2ME hala gelişmektedir ve J2ME için temel destek hala büyümektedir.

J2ME kablosuz ve mobil cihazlar için Java olarak bilinir. J2ME teknolojisi pek çok kablosuz ve mobil cihaz için kullanılsa da J2ME sadece bu ortamlarda kullanılmaz. J2ME mobil platformlar için önemli bir Java platformu olmasına rağmen mobil platformlar için tek Java platformu değildir. Ve sadece Java da bir çözüm değildir.

"Mobil" bir cihazın kapasite veya durumunu tanımlar. Mobil cihazlar daha küçüktür ve bunların kaynakları kısıtlıdır. J2ME mobil cihazlar için önemli bir rol oynasa da "mobil" deyimi tüm J2ME uygulamalarını kapsamaz. Ayrıca J2ME kablosuz cihazlar üzerinde de çalışabilir, fakat "kablosuz" deyimi de "mobil" deyimi gibi tüm J2ME uygulamalarını kapsamaz.

J2ME'nin arkasında yatan yüksek seviyeli düşünce elektronik ve gömülü cihazlar pazarı için dinamik olarak genişletilebilen, ağ özellikleri fazla cihazlar ve uygulamalar yaratmak için etraflı bir uygulama geliştirme platformu sağlamaktır.

J2ME'in sınırlı özelliklerin bulunmasının temel sebebi mobil telefonların ve mesaj cihazı gibi bilgisayar sistemlerin kapasitelerin kısıtlı olmasıdır. İlk olarak bu sistemlerin kısıtlı hafızaları bulunmaktadır. Ayrıca bu sistemlerin işlemcileri basit bir bilgisayarlarla bile kıyaslanmayacak kadar azdır. Kablosuz cihazların diğer bir farkı basit klavye özellikleridir. Birçoğunun kullandığı klavye keypad olarak bilinen ve birkaç tuştan oluşan bir teknolojidir. Bunlara ilaveten, kablosuz cihazların ekranları küçük ve birçoğu ekranı yanlızca siyah beyazdır.
Kablosuz cihazların bahsi geçen sınırlamaları göz önüne alındığında J2ME'nin neden diğer sürümlere oranla kısıtlı özellikler taşıdığı daha iyi anlaşılmaktadır. Java'nın kablosuz cihazlar için ürettiği J2ME sürümü üretilmeden önce Wireless Access Protocol (WAP) ve i-mode adında iki programlama platformu geliştirilmiştir. Bu teknolojilerde hali hazırda var olan programlama platformları tamamen yeniden düzenlenmiştir. Örneğin WAP, HTML adındaki normal bilgisayarlarda kullanılan dil yerine, Wireless Markup Language (WML) adında kendi uygulama dilini hazırlamıştır. Benzer çalışmalar "i-mode" teknolojisinde de yapılmıştır. Tamamen yeniden düzenlenen programlama yapısı, farklı makinelerde uyum problemine yol açmaktadır. Farklı yapıdaki programlama yapısı ile aynı programın değişik makinelerde kullanılması mümkün olmamaktadır. Örneğin, WAP destekli cihazlar normal bilgisayarlar için hazırlanan web sayfalarını ziyaret edememektedirler. Bu yüzden WAP kullanıcılarının bu sayfaları görebilmeleri için tamamen yeniden bir sayfanın hazırlanması gerekmektedir. Benzer bir şekilde, i-mode kullanıcıları sitelere basit socket bağlantıları yapamamakta ve ancak bağlantı kurucu ek protokoller ile bağlantı işlemlerini yerine getirebilmektedirler. WAP ve i-mode sistemleri hakkında bahsi geçen farklı programlama yapıları J2ME için geçerli değildir. Java dilinde yazılan bir program ile socket bağlantısında, internet sitelerine giriş direkt olarak sağlanabilmektedir [2].

Değişik mimarisinin sonucu olarak J2ME, J2SE ve J2EE'ye kıyasla değişik amaçlara sahiptir. Aşağıda J2ME mimarisinin ana hedefleri özetlenmeye çalışılmıştır:

- Değişik yetenekteki cihazlara destek sağlar. Bu cihazlar kullanıcı arayüzü, veri depolama, ağ bağlantısı ve bant genişliği, bellek büyüklüğü, güç tüketimi, güvenlik gereksinimleri duyarlar.
- Çok fazla kişiselleştirilmiş hatta sadece tek bir kişi tarafından kullanılan cihazlara odaklanmıştır.
- Değişen aralıklardaki ağ kapasitelerinde ve servislerinde ağ bağlantısı sağlar. Ağ bağlantısı J2ME uzayında yer alan cihazlar için çoğunlukla hayati önem taşır ve yetenekleri, düşük bant genişliği, kablosuz ve aralıklı bağlantıdan daha sık ve daha yüksek bant genişliğindeki bağlantılar aralığında değişir.
- Ağ bağlantısı üzerinden uygulamaların ve verinin alınmasını sağlar. Ağ bağlantısı J2ME uygulamalarının cihazlara taşınmasında tercih edilen bir yoldur. Uygulamaların cihaz üzerinde hazırlanabilmesi veya belleğe doğrudan yüklenebilmesi ve çalıştıktan sonra bellekten atılabilmesi gerekir.
- Java dilinin çapraz-platform özelliklerini her cihazın eşsiz özelliklerini ve kısıtlarını alarak genişletir.
- Hızlıca değişen pazara esneklik ve var olan ve henüz çıkmamış uygulamalara uyum sağlar.
- Değişik yetenekteki, özellikteki ve işlem gücündeki cihazlarda uygulamaların ölçeklenmesini sağlar.
- Original Equipment Manufacturer (OEM)'dan bağımsız olarak J2ME destekli cihazlarda uygulama geliştirmeyi sağlar.

2.2.1. Neden J2ME'ye gerek duyuluyor?

Java ilk olarak elektronik cihazlar için tasarlandığından doğal olarak akla neden yeni bir sürüme gerek duyulduğu sorusu geliyor. Neden standart Java küçük cihazlar için de kullanılmıyor? Daha iyi bir organizasyon için API'leri üç

ayrı sürüme ayırma fikrinin arkasında Sun firmasının gelirlerini arttırma isteği olabilir. Ayrıca Java'yı değişik sürüme ayırmanın bir gerekliliği de J2ME'nin hedef cihazlarının özel gerekliliklerinin olduğudur. Bu cihazların geniş uygulama yazılım ortamından farklı yazılım gereksinimleri var. Genelde yazılımın küçük bir ayak izi olması gerekir. Bazı durumlarda Java uygulamaları, Java sınıfları ve sanal makine için gereken toplam bellek yüzlerce kilobaytı bulabiliyor. Ayrıca, elektronik ve gömülü cihazlar için geliştirilmiş yazılım uygulamaları genellikle tekil yayılma mekanizmasına sahiptirler. Örneğin, PDA cihazlarının masaüstü bilgisayarlarından uygulamaları ve veriyi indirmek için "cradle" adı verilen cihazları vardır. Sonuç olarak, bu cihazların kullanıcı arayüzü, ağ bağlantıları var ve diğer pek çok gereklilikleri Java API'leriyle karşılanamaz. Mobil telefonların küçük ekranları için grafik kullanıcı arayüzü geliştirirken Java'nın kullanıcı arayüzü geliştirmek için kullanılan Swing paketinin içerdiği bileşenler genişletilmelidir. Fakat bu paket mobil telefonun belleğine sığmaz. Bir Java ortamının tüm cihazlara uymayacağı gerçeğini J2ME göstermiştir. Platform bağımsızlığının gerektirdiği aynı prensipler, dil sözdizimi, güvenlik ve güvenilirlik J2ME de dahil tüm Java sürümlerinde bulunmaktadır.

2.3. Konfigürasyon ve Profiller

J2ME mimarisinin çözmeye çalıştığı en önemli problem farklı kısıtlara, özelliklere, becerilere sahip çeşitli cihazları nasıl destekleyeceği sorunudur. Birinci çözüm yolu; her cihaz için gerekli olacak tüm mimari elemanları katarak Java'yı genişletmekti. Fakat bu bellek sıkıntısı çeken küçük cihazların kullanılamamasına yol açtı. Başka bir çözüm yolu, bu cihazların ortak özelliklerini içerecek şekilde J2ME'yi kısıtlamaktı. Bu çözüm yoluyla da güçlü cihazların becerilerini özellikleri, becerileri daha az olan araçlarmış gibi sınırlandırmak zorunda kalınmıştır.

İki çözümde ihtiyaçları karşılamadığı için J2ME çeşitli konfigürasyon ve profillere bölünmüştür. Konfigürasyon ve profiller J2ME'nin modüler yapısını oluşturan temel elemanlardır. Bu iki eleman J2ME destekli pek çok cihaz için uygundurlar [3].

J2ME konfigürasyonu küçük cihazlar ailesi için minimum Java platformunu tanımlar. Bu ailenin üyelerinin hepsi aynı bellek ve işlemci gücü gereksinimlerine sahiptir. Konfigürasyon, uygun sistem seviyesindeki özellikleri, Java dili özelliklerini, mevcut sanal makine karakteristik ve özelliklerini ve minimum Java kütüphanelerini tanımlar. Yazılım geliştiriciler belirli bir konfigürasyonu kullanan cihaz ailesi için belirli bir düzeyde sistem desteğinin olabileceğini umarlar. Ayrıca bir konfigürasyon belirli bir cihaz kategorisi için minimum özellikler kümesini belirler. Cihaz üreticileri, gerçek bir platform sağlamak amacıyla belirtilen konfigürasyondaki yetenekleri mevcut, belirli bir cihaz ailesi için profiller gerçekleştirirler.

Diğer bir J2ME yapı taşı olan profil, belirli sınıftaki cihazlar için uygulama seviyesinde arayüz tanımlarlar. Profil gerçekleştirimi, bu uygulama seviyesindeki arayüzleri sağlayan Java sınıf kütüphanelerini içerir. Böylece profil, teorik olarak tüm işlevsellik ve servisleri belirler. Aslında yaratıcıların niyeti tam olarak bu değildi. J2ME yaratıcıları profilin belirli bir cihaz kategorisinin ihtiyaçlarını göstermesi niyetindeydiler. Düşünce, profilin içinde çok sayıdaki alakasız uygulamayı toplamak değildi.

Asıl amaç, aynı kategorideki tüm cihazlar arasında Java uygulaması geliştirmek için standart bir platform tanımlayarak "interoperability"i–değişik üreticilerin gerçekleştirimleri arasında uyumluluk– garanti etmektir. Örneğin, bir profil mobil telefonlarca kullanılan Kısa Mesaj Servis (SMS) standardı için ağ iletişimini destekleyebilir. Çünkü SMS standardı mobil telefonların her yerde olan bir özelliğidir, bu yüzden mobil telefonları hedef alan bu özelliği konfigürasyonun içine yerleştirmek yerine profilin içinde tanımlamak gerekir.

Profil konfigürasyonun üzerinde gerçekleşir, yani gerçek dünya uygulamalarına bir adım daha yakındır. Tipik olarak, profiller konfigürasyonların oluştuğu kütüphanelerden daha cihaz karakteristiklerine özel kütüphaneler içerirler. Uygulamalar, konfigürasyon ve profilin üzerine yerleşirler; bu uygulamalar sadece bu iki düşük seviyeli spesifikasyonların desteklediği sınıf kütüphanelerini kullanırlar. Profillerden biri bir diğerinin üzerine yerleşebilir. J2ME platformu sadece bir konfigürasyon içerebilir.

Şekil 2.1: J2ME platformu

J2ME platformu, temel Java kütüphaneleri ve Sanal Makineyle(VM) temel çalıştırma ortamından, konfigürasyondaki sistem seviyesinde uygulama programlama arayüzleri kümesinden ve profillerdeki uygulama seviyesi API'lerden oluşur. (Şekil 2.1)

Konfigürasyon üç temel elemanı tanımlar:

- Java programlama dili özellikleri kümesi
- Java sanal makine özellikleri kümesi
- Java kütüphanelerinin ve uygulama programaları arayüzlerinin (APIs) kümesini

J2ME yaratıcıları uyumsuz platformlar arasındaki parçalanmışlığı önlemek için sadece iki konfigürasyon tanımlamışlardır.

Konfigürasyonlar arasında iç içe bir ilişki vardır. J2ME mimarisindeki tüm konfigürasyonlar süper küme altküme sıralanışı içindedirler. Bu kısıtlanmış bir konfigürasyondan daha zengin özellikli bir konfigürasyona doğru ilerlerken taşınabilirliği arttırır.

Teorik olarak, bir konfigürasyon J2SE platformu kütüphanelerinin benzeri bir destek sağlar. Fakat gerçek dünyada bu mümkün olmaz çünkü J2ME masaüstü bilgisayarlarından daha az güçlü cihazları hedefler. Konfigürasyon spesifikasyonları, J2SE den uyarlanmış tüm Java sınıflarının aynı veya altkümesi şeklinde orijinal J2SE sınıflarını gerektirir. Yani bir sınıf J2SE sürümünde bulunmayan bir methodu ekleyemez. Konfigürasyonlar, kendi spesifikasyonlarına sınıflar ekleyebilirler, yine de konfigürasyonlar mutlaka J2SE'nin bir altkümesi

değildirler. Örneğin, her iki konfigürasyon da cihaz özelliklerine ve kısıtlarına bağlı olarak J2SE'de bulunmayan tarih sınıflarını tanımlamışlardır.

Konfigürasyonlar JVM(Java Virtual Machine) detaylarını ve belli sınıftaki cihazlarla kullanılabilecek temel kütüphaneleri tanımlayan belirtimlerdir. Örneğin 512KB bellekten az hafıza alanı ve sınırlı bir ağ bağlantısı olan cihazların konfigürasyonu CLDC olarak adlandırılır. CLDC lere örnek olarak bazı mobil telefonlar ve avuç içi bilgisayarlar (PDAs) verilebilir. Daha fazla belleğe ve işlemci gücüne sahip, sürekli ağ bağlantısı olan diğer konfigürasyon ise CDC olarak adlandırılır (Connected Device Configuration). Bunlara örnek olarak ise Sharp Zaurus avuç içi bilgisayarları verilebilir.

Profiller uygulama geliştirmeye daha yetkin bir ortam sunabilmek için konfigürasyonun üzerine tanımlanan kütüphanelerdir. Konfigürasyon java sanal makinesi (JVM) ve temel kütüphaneleri tanımlarken, bir uygulama geliştirilirken ihtiyaç duyulabilecek yetkinlikte kütüphaneleri tanımlamaz. Profiller bu alanda uygulamanın yaşam döngüsü, kullanıcı arabirimi ve kalıcı bellek kütüphanelerini içerirler.

J2ME ortamı sanal makineden, bir konfigürasyondan ve bir veya daha fazla profilden oluşur. Sanal makine işletim sistemiyle konfigürasyon arasındaki bağlantıyı kurar. Profiller ise uygulama ile J2ME ortamı arasındaki bağlantıyı sağlar.

2.4. Konfigürasyonlar

2.4.1. CLDC Konfigürasyonu

CLCD (Connected, Limited Device Configuration) daha kısıtlı kaynaklara sahip cihazlara yönelik bir J2ME konfigürasyonudur. Bu profilin çekirdeğinde bir Java sanal makinesi olan KVM yatmaktadır. KVM, JVM nin bazı kısımlarının çıkartılarak JVM'nin daha önemli ve gerekli parçalarını içerecek şekilde tasarlanmış halidir. Bu konfigürasyona ait Java paketleri:

- java.io
- java.lang

- java.lang.ref
- java.util
- javax.microedition.io

Kaynakları kısıtlı cihazların (CLDC) aşağıdaki karakteristikleri olmalıdır:

- Java ortamı için 160Kb tan 512Kb'a kadar toplam bellek.
- 16-bit veya 32-bit işlemci.
- Düşük güç tüketimi. Genellikle bu cihazlar pil gücüyle çalışırlar.
- Belirli bir çeşit ağ bağlantısını desteklemelidirler. Kesikli ve düşük bant genişliğindeki bağlantılar.

CLDC J2SE'yi baz alır fakat bazı özellikleri dahil etmez. CLDC bir temel üzerine aşağıdaki özelliklerden gerekli olanları eklenerek oluşturulur:

- Bu tip cihazlar için fonksiyonellik uygun mudur?
- Bu fonksiyonellik büyük miktarda binary koda gerek duyar mı veya bellek ve CPU zamanından çok fazla tüketir mi?
- Gerektiğinde fonksiyonellik kolaylıkla sağlanabiliyor mu?
- Bu cihazlar genellikle fonksiyonelliği destekliyorlar mı?
- Bu kaynakları kısıtlı cihazlarda fonksiyonelliğe bağlı bir güvenlik riskleri var mı?

CLDC küçük cihazların ihtiyaçlarını karşılamak için J2SE ortamında bulunan bazı özellikleri çıkarmıştır [4].

2.4.2. Kilobayt Sanal Makine (KVM)

KVM olabildiğince Java Sanal Makine Spesifikasyonlarına benzerdir. KVM'nin yetenekleri CLDC spesifikasyonlarıyla tanımlanmıştır. KVM, Java Sanal Makine Spesifikasyonlarından iyileştirme veya API desteği nedenleriyle CLDC gerektirdiğinde veya izin verdiğinde farklılaşır. Örneğin, kayan noktalı veya double veri tipileri CLDC uzayındaki cihazlar tarafından desteklenmez. Sonuç olarak, bu cihazlar üzerinde bu veri tiplerini gerçekleştirmek çok pahalı olacağından üreticiler tarafından desteklenmemektedir. Kayan noktalı ve double tipi CLDC tarafından desteklenmediğinden KVM tarafından da tanınmamaktadır.

KVM cihazlar üzerinde küçük bir yer gerektirir, derleme tercihine ve hedef platforma bağlı olarak 40Kb ve 80Kb arasında. Bu da KVM nin 128Kb toplam belleğe sahip cihazlar üzerinde bile çalışabilmesini sağlar.

KVM, C tabanlı geliştirilmiştir ve %30 dan %80 arasındaki bir hızda çalışan JIT (just-in-time-compiler) olmayan standart Java Sanal Makinesi kadar tam ve hızlı dizayn edilmiştir.

KVM'nin hedef cihazlardaki rolü çok önemlidir. Bazı gerçekleştirimlerde, KVM cihaza dinamik, interaktif, güvenli Java içeriklerini yükleyebilmek ve çalıştırabilmek için mevcut yazılım yığının üstünde kullanılır. Bazı uygulamalarda ise, KVM Java programlama dili içindeki, cihazın düşük seviyeli sistem yazılımlarını ve uygulamalarını da gerçekleştirebilmek için düşük seviyede kullanılabilir [5].

2.4.3. Sınıf Dosyalarının Doğrulanması

Standart Java sanal makinesi sınıf dosyalarının gerçeklenmesi (class file verification) çalışma zamanında gerçekleştirilen bir süreçtir.

Bu işlem, sınıfın geçerli bir Java sınıf dosyası olduğundan ve "iyi-davranışlı" yani tanımlanandan farklı bir yere ulaşmaya veya java.* paketini javax.* paketiyle değiştirmeye çalışmadığından emin olmak için yapılır. Sınıf dosyalarının gerçeklenmesi Java güvenlik modelinde önemli bir rol oynar.

CLDC cihazları için sınıf dosyalarının doğrulanması yoğun bir kaynak operasyonları ve önemli miktarda güç, bellek ve binary kod uzayı kullanır. Sonuç olarak KVM sınıf dosyalarını doğrulanmasını standart Java Sanal Makinesinden farklı tanımlar.

KVM'yi küçültmek için, çoğu sınıf dosyalarının gerçekleştirimi işlemi KVM'nin hatta cihazın dışında yapılır. Bir sınıf bir cihazın içine yerleştirilmeden önce sınıf ,"preverify" tarafından değerlendirilir. "preverify" sınıf dosyasını javac derleyicisi tarafından oluşturur, sınıfın geçerli bir sınıf olduğunu gösteren bytecode'ları ekler. Çalışma zamanında KVM bu sahaları kontrol eder. Eğer bu sahalar bulunmuyorsa veya doğru bilgi içermiyorlarsa, sınıfın yüklenmesi durdurulur ve aykırı durum fırlatılır.

2.4.4. CDC Konfigürasyonu

CDC (Connected Device Configuration) daha az kısıtlı kaynaklara sahip cihazlara yönelik bir J2ME konfigürasyonudur. Bu profilin çekirdeğinde de CVM (Compact Virtual Machine) sanal makinesi bulunmaktadır. Genellikle CLDC den daha fazla hafıza ve işlemci gücüne sahip cihazlarda kullanılır. Ayrıca bu cihazlar sürekli ağ bağlantısına sahiptirler [6].

Bu konfigürasyona ait Java paketleri:

- java.io
- java.lang
- java.lang.ref
- java.lang.reflect
- java.math, java.net
- java.security
- java.security.cert
- java.text, java.util
- java.util.jar, java.util.zip
- javax.microedition.io

2.4.5. C- Sanal Makinesi (CVM)

CVM, Java Sanal Makinesi Spesifikasyonlarına tamamen uysa da, cihazlar ve ağ uygulamaları için iyileştirildiğinden gerçekleştirimi J2SE sanal makinesinden tamamen farklıdır. Garbage collection algoritmaları CVM'nin içine değişik garbage collection algoritmaları yerleştirilebilsin diye sanal makineden tamamen ayılmıştır. Referans gerçekleştirimi, sanal makinenin uzun garbage collection periyotlarına oranla daha kısa garbage collection periyotları kullanır. Garbage collection daha sık ve kısa zaman aralıklarında çalışır. Garbage collector daha dikkatlidir, garbage collection zamanında tüm pointerları bilir böylece CPU devirlerinden fazla harcamamış olur. Platformlar arası taşınabilirliği arttırmak için referans gerçekleştirimi multithreading'i sanal makinenin içinde tanımlar. Sanal makinenin içinde gerçekleşen threadlere "green threads" denir. "green

thread"lerin kullanımı, multithreading için işletim sistemi bağımlılığı olmadığından beri sanal makinenin taşınabilir olmasını sağlamıştır. Sınıf dosyası gerçeklenmesi cihazda gerçekleşir. CDC kullanırken preverification adımı yoktur.

2.5. Profiller

2.5.1. MIDP Profili

MIDP (Mobile Information Device Profile) mevcut profiller arasında ilk ve en yaygın olanıdır ve uygulamanın yaşam döngüsü, kullanıcı grafik arabirimleri, iletişim ağı ve kalıcı depolama ile ilgili kütüphanelerini içerir. MIDP CLDC konfigürasyonunun üzerine oturur ve günümüzde Motorola, Nokia, Ericsson ve RIM (Blackburry) gibi sektör devleri tarafından desteklenmektedir.

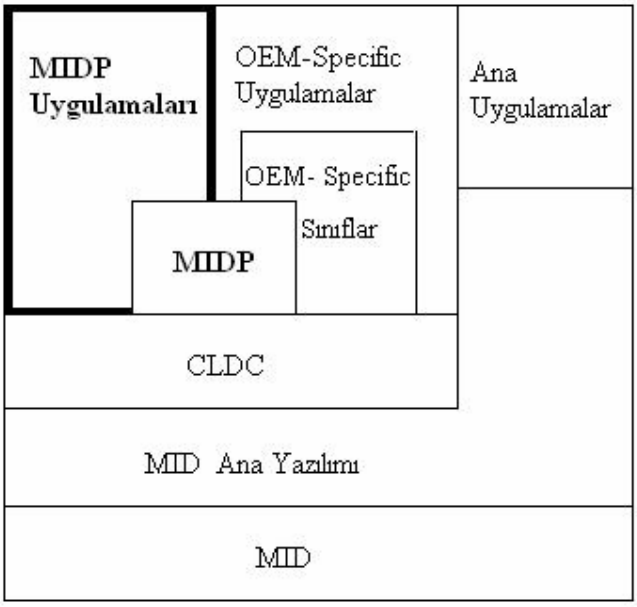
MIDP 1.0 ın desteklediği paketler:

- java.io
- java.lang,java.util
- javax.microedition.io
- javax.microedition.lcdui
- javax.microedition.midlet
- javax.microedition.rms

MIDP 2.0 ın desteklediği paketler:

- java.io
- java.lang
- java.util
- javax.microedition.io
- javax.microedition.lcdui
- javax.microedition.lcdui.game
- javax.microedition.media
- javax.microedition.media.control
- javax.microedition.midlet, javax.microedition.pki
- javax.microedition.rms

Günümüzde J2ME özelliğine sahip mobil telefon uygulamalarının büyük bir çoğunluğu Şekil 2.2'deki mimariyi kullanır:



Şekil 2.2: Mobil telefon uygulamalarında kullanılan mimari

Bu profil Palm işletim sistemi (Palm OS) üzerinde de çalışabilir. Bu profili kullanan cihazlar çok fazla kişiselleşmiş cihazlardır. Çoğu zaman bu cihazların kullanıcısı sadece bir kullanıcıdır. Bu cihazların kullanıcı arayüzü için küçük ekranları, veri girişi için küçük klavyeleri ve limitli veri depolama yetenekleri gibi kısıtlı kaynakları vardır [7].

2.5.2. PDA Profili (PDAP)

Bu profil PDA-stilinde grafiksel kullanıcı arayüzüne ve dokunmatik ekrana sahip kullanıcı arayüzü ve veri depolama yeteneklerine sahip kişisel dijital ajandaların gerekliliklerini karşılamak içindir.

2.5.3. Kuruluş Profili (Foundation Profile)

Kuruluş profili garfiksel kullanıcı arayüzü, veri depolama, dağıtık Java ağ bağlantısı gibi özellikler sağlayan CDC konfigürasyonuna bir taban gibi hizmet

eder. Taban profili olarak işlevlerinin yanında yüksek bant genişiliği ve fazla bağlantı kuran cihazlar için zengin bir ağ desteği sağlar. Bu profil, kişisel bilgisayarlardan daha küçük cihazlar için diğer profillerle kullanılarak daha zengin bir uygulama ortamı sağlar.

2.5.4. Kişisel Profil (Personal Profile)

Kişisel profil pek çok kişisel Java API'leri için yeni bir yurttur. Cep bilgisayarlarını hedef alan kişisel-Java API'leri J2ME mimarisine uyacak şekilde yeniden oluşturuldu. Kişisel java CDC'nin içinde Kuruluş Profili ve Kişisel Profile ayrıldı. Kişisel profile ek olarak JavaPhone ve JavaTV API'leri eklendi.

2.5.5. RMI Profili

RMI profili CDC uzayındaki uygulamalar için dağıtık bir destek sağlar. Bu profil uzak metod çağırımlarının parametrelerine ve dönüş değerlerine bir altyapı sağlar. Kablo protokolü olan JPMP (Java Remote Method Protocol) desteklenmelidir. Aşağıdaki paketler RMI profilinin içinde yer almaz:

- java.rmi.server.disableHttp
- java.rmi.activation.port
- java.rmi.loader.packagePrefix
- java.rmi.registry.packagePrefix
- java.rmi.server.packagePrefix

2.5.6. Kişisel Temel Profil (Personal Basis Profile)

Bu profil CDC konfigürasyonunda ve kuruluş profilinde çalışan cihazların grafiksel yetenekleri için temel bir seviye sağlar. Ayrıca kişisel profiling grafiksel yetenekleri için temel oluşturur.

2.5.7. Multimedya Profili (Multimedia Profile)

Bu profil ses ve diğer medya için CLDC ve CDC konfigürasyonları için temel multimedia desteği sağlar. Java Media Framework'ün fikirleri benimsenmiştir fakat bu profil JMF ile uyumlu değildir. Multimedya profili diğer J2ME profilleriyle kullanılmak üzere seçimlik geliştirilmiştir.

2.5.8. Oyun Profili (Gaming Profile)

Bu profil J2ME cihazları için oyun desteği sağlar. CDC konfigürasyonu bu profilin hedef ortamıdır. Diğer J2ME profilleriyle kullanılabilecek seçimlik bir profildir [8].

3. MIDP PROGRAMLAMA

Mobil telefonlar, kişisel dijital ajandalar gibi küçük kapasiteli araçlarda uygulamalar "MIDP" profili kullanılarak, "javax.microedition.midlet.MIDLET" sınıfı genişletilerek oluşturulur. Bu sınıf, araçta bulunan Uygulama Yönetim Yazılımı ile MIDP uygulamaları arasında bir arayüz gibi davranır.

MIDP uygulamaları oluşturulurken, başlangıç noktası bir MIDlet'tir.

3.1. MIDP Uygulaması Geliştirme

3.1.1. MIDlet nedir?

MIDlet, MIDP uygulamalarının temel noktasını oluşturmak için yaratılan soyut bir sınıftır. MIDlet sınıfı, "javax.microedition.MIDlet" paketi içinde yer alır. Bu nedenle gerçekleştirilmek istenen bir MIDlet'te aşağıda verilen kod parçaları mutlaka yer almalıdır.

```
import javax.microedition.midlet.MIDlet;

public class MerhabaDunya extends MIDlet {
}
```

Eğer bir metin kutusu görüntülenmek istenirse sınıf içinde bir "constructor" tanımlanmalı ve metin kutusunun tanımlanma kodu da "constructor"ın içinde yer almalıdır.

```
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.*;

public class MerhabaDunya extends MIDlet {
        private TextBox textbox;

        public MerhabaDunya() {
                textbox = new TextBox("", "Merhaba Dünya!", 20, 0);
```

```
        }
    }
```

MIDlet'ler "startApp()", "pauseApp()" ve "destroyApp(boolean b)" olmak üzere üç tane önemli metod içerirler. MIDlet başlatıldığı zaman, cihazdaki uygulama yönetim servisi ilk olarak "startApp()" metodunu çağırır.

startApp() : MIDlet'in yaşam döngüsü boyunca defalarca kez çağırılabilir. İlkleme işlemlerinin gerçekleştirildiği bir metod değildir. Uygulamanın çalıştırılacağı cihazda, bir MIDlet'in başlatılacağı mesajı alınınca Uygulama Yönetim Servisi bu metodu çağırır. Metod içinde metin kutusunun aktif duruma getirildiği bir örnek kod parçası aşağıda verilmiştir:

```
public void startApp() {
        Display.getDisplay(this).setCurrent(textbox);
}
```

pauseApp() : Kullanıcı veya cihaz, çalışan uygulamayı kesip başka bir görev gerçekleştirmek istediğinde cihaz tarafından çağırılan bir metoddur. Bu metod çağırıldığında MIDlet "duraklatılmış" duruma geçer.

Eğer ekranda sadece bir metin kutusu gösterilecekse "pauseApp()" metodu boş olarak gerçekleştirilebilir.

```
public void pauseApp() {
}
```

destroyApp (boolean b) : Kullanıcı uygulamayı kapatmak istediğinde veya sistem (bir nedenden dolayı) uygulamanın kapatılmasını isterse bu metod çağırılır. Bu metod, çalışmakta olan uygulamaya kullandığı kaynak varsa, bunları düzenlemesi için olanak tanır. Parametre olarak "TRUE" değerini alırsa, uygulamanın kaynakları temizlemekten başka çaresi yoktur. Parametre olarak "FALSE" değerini alırsa, uygulama çalışmaya devam etmek için "MIDletStateChangeException" fırlatabilir. Eğer metoda parametre geçirilmezse uygulamanın temizlemesi gereken kaynağı yok demektir.

Ekranda sadece bir metin kutusu gösterileceğinden ve herhangi bir kaynak temizlemesine gerek duyulmayacağından "destroyApp(boolean b)" metodu da boş olarak gerçekleştirilebilir.

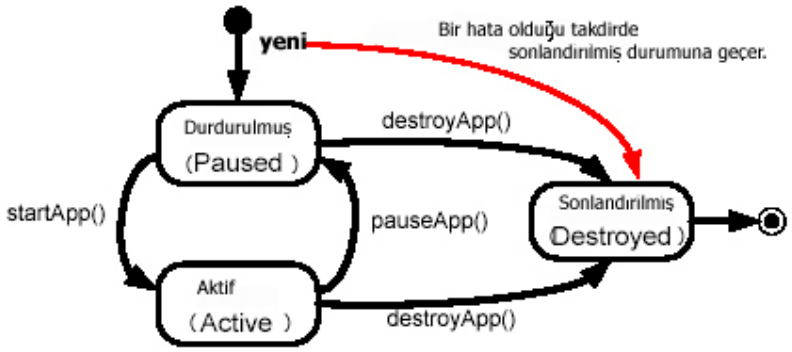
```
public void destroyApp(boolean unconditional) {    }
```

Basit bir MIDlet için temel olarak yukarıda bahsedilen üç metod yeterlidir. Bu üç metodun kullanıldığı "MerhabaDunya" MIDlet'inin kodu aşağıda verilmiştir:

```
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.*;

public class MerhabaDunya extends MIDlet {
        private TextBox textbox;
        public MerhabaDunya() {
                textbox = new TextBox("", "Hi Small World!", 20, 0);
        }
        public void startApp() {
                Display.getDisplay(this).setCurrent(textbox);
        }
        public void pauseApp() {
        }
        public void destroyApp(boolean unconditional) {
        }
}
```

Bir MIDlet yaratıldığı veya başlatıldığı zaman "Duraklatılmış" durumdadır. Eğer herhangi bir aykırı durum oluşursa MIDlet "Sonlandırılmış" duruma geçer. MIDletler "Aktif" durumdan "Duraklatılmış" duruma, "pauseApp()" metodunun işletilmesi tamamlandıktan sonra geçerler. "Sonlandırılmış" duruma ise "destroyApp()" metodu tamamlanınca geçilir [9]. (Şekil 3.1: Bir MIDlet'in Yaşam Döngüsündeki Basamaklar)



Şekil 3.1: Bir MIDlet'in yaşam döngüsündeki basamaklar

3.1.2. Uygulamanın derlenmesi

Derleme işleminin gerçekleştirilebilmesi için "MIDP Geliştirme Ortamı"na ihityaç vardır. Sun'ın MIDP ürünü "http://java.sun.com" internet adresinden indirilebilir. MIDP, "midp-fcs" dizinine kaydedilir. MIDP kaydedildikten sonra bazı ortam değişkenlerine değer verilmesi gerekmektedir:

MIDP=\midp-fcs

MIDPClasses=\midp-fcs\classes

MIDPTools=\midp-fcs\bin

Ortam değişkenlerine değerleri verildikten sonra derleme işlemine geçilebilir.

Standart "javac" derleme komutu kullanılarak, derleme işlemi yapılmaktadır. "javac", J2SE uygulamalarını derlemek için kullanılmaktadır; ancak, J2ME uygulamalarının da "javac" komutu ile derlenmesi için "-bootclasspath" opsiyonu ile kullanılmalıdır. "-bootclasspath" opsiyonu kullanılarak, derleyicinin J2ME kütüphanesini kullanması sağlanır.

J2ME uygulamalarının derlenmesi için;

>javac -g:none -bootclasspath %MIDPClasses% HiSmallWorld.java

komut satırı kullanılmalıdır.

"-g:none" opsiyonu kullanılarak, "*.class" dosyalarının hata ayıklama bilgilerini içermesi önlenir. Bu opsiyon kullanılarak, "*.class" dosyalarının boyutları küçük tutulmaktadır. "%MIDPClasses%" değişkeni, bir ortam değişkenidir. Bu değişeken J2ME sınıflarınının bulunduğu dizini göstermektedir.

3.1.3. "Preverifying" işleminin gerçekleştirilmesi

Güvenlik nedenleri ile standart "Java RuntimeEnvironment", "class" dosyalarını belleğe yüklemeden önce doğrulama işleminden geçirir. Bu doğrulama işlemi, "class" dosyasının geçerli olup olmadığının anlaşılması için gerçekleştirilir.

J2ME cihazları, masaüstü bilgisayarlarına göre daha kısıtlı özelliklere sahip olduklarından bazı J2ME sanal makineleri; "class" dosyalarının doğrulanma işlemini, standart Java sanal makineden farklı şekilde yapmaktadır. J2ME cihazlarında, doğrulama işleminin tamamı cihaz üzerinde yapılmaz. Her "class" dosyası, J2ME geliştirme ortamının sahip olduğu "preverify" özelliği ile önceden doğrulanmalıdır. "Preverify" işlemi, tüm sınıfları doğrulama işleminden geçirir ve sonrasında dosyalara doğrulandıklarını belirten özel bir değer ekler. Çalışma zamanında ise J2ME sanal makinesi bu özel değerleri kontrol eder. Eğer bu özel değer geçerli ise, sanal makine sınıfı çalıştırabilir. Ancak; "class" dosyası özel değere sahip değilse sanal makine aykırı bir durum olduğunu tespit eder ve sınıfın yükleme sürecini durdurur.

Önceden doğrulama işlemi "preverify.exe" dosyası çalıştırılarak yapılmaktadır. Bir uygulamanın önceden doğrulanması için ;

>%MIDPTools%\preverify -classpath %MIDPClasses%;. HiSmallWorld

komut satırı kullanılmaktadır. Önceden doğrulama işlemi sonucunda, "class" dosyaları oluşturulur.

3.1.4. Uygulamanın Çalıştırılması

Sınıfları derlenen ve doğrulama işlemi gerçekleştirilen uygulama çalıştırılmaya hazırdır. Uygulamanın çalıştırılabilmesi için bir emülatöre ihtiyaç vardır. "MIDP Referans Gerçekleştirimi" de bir emülatördür. Emülatörün çalıştırılabilir ismi, "midp" dirve "midp-fcs\bin" dizini altında yer almaktadır. Uygulamanın çalıştırılabilmesi için aşağıdaki komut satırı kullanılmaktadır:

>%MIDPTools%\midp -classpath %MIDPClasses%;.\output MerhabaDunya

Bu komut "-classpath" parametresi ile uygulamada bulunan sınıf dosyalarını çalıştırılan "midp" ye geçirir. ".\output" parametresi ise, önceden doğrulama işlemi sonucunda oluşmuş olan "class" dosyalarının nerede kayıtlı olduklarını göstermek için kullanılır.

Eğer "MerhabaDunya.java" uygulaması sorunsuz bir şekilde çalışırsa ekran görüntüsü Şekil 3.2'deki gibi olacaktır.

Şekil 3.2: MerhabaDunya.java programının ekran görüntüsü

Emülatör kapatıldıktan sonra komut ekranına aşağıdaki gibi çıktılar yazılmaktadır:

D:\java\MerhabaDunya>\midp-fcs\bin\midp -classpath \midp-fcs\classes;.\output

MerhabaDunya

Execution completed successfully

8205 bytecodes executed

7 thread switches

204 classes loaded (149 bytes)

220 objects allocated (9572 bytes)

0 garbage collections

0 bytes collected

0 objects deferred in GC

0 (maximum) objects deferred at any one time

0 rescans of heap because of deferral overflow

0 pointer validations requiring heap scans

Current memory usage 9572 bytes

Heap size 300000 bytes

3.1.5. Uygulamaların JAR dosyası haline getirilmesi

Birçok durumda, MIDP uygulamaları "JAR" dosyaları haline getirilir. Kullanılan ağ protokolü ve onun içerdiği istemci-sunucu yazılımına bağlı olarak, belirli bir protokol üzerinden birçok uygulama yüklemesi yapılırken JAR dosyaları daha verimlidir. Örneğin; HTTP protokolü ile uygulama yüklenirken her sınıf dosyası için bir bağlantıya ihtiyaç duyulurken, JAR dosyası için tek bağlantı yeterli olmaktadır.

Mevcut olan "class" dosyalarını kullanarak jar dosyası oluşturulurken;

>jar cf merhaba.jar -C .\output MerhabaDunya.class

komut satırı işletilir.

"cf" parametresi, "jar" çalışabilir dosyasına "merhaba.jar" isimli bir yeni bir JAR dosyası oluşturması gerektiğini belirtir. "-C" opsiyonu ise, ".\output" dizini altındaki "MerhabaDunya.class" dosyasına erişebilmeyi sağlar.

Uygulamayı, oluşturulan "merhaba.jar" dosyasından çalıştırabilmek için JAR dosyasına yol tanımlamak gerekmektedir.

```
>%MIDPTools%\midp -classpath %MIDPClasses%;.\merhaba.jar
MerhabaDunya
```

Komut satırı işletilerek uygulamanın ilgili jar dosyasından çalışması sağlanmaktadır.

3.1.6. MIDlet takımı geliştirilmesi

Birçok MIDlet, bir MIDlet takımı kullanılarak gruplanabilir. Bir MIDlet takımı; tüm MIDlet'leri içeren bir JAR dosyası, desteleyici sınıflar ve Uygulama Tanımlayıcı Dosyası'ndan oluşur. Uygulama Tanımlayıcı Dosyası, MIDlet takımı hakkındaki bilgileri içeren metinsel bir dosyadır. Bu dosyanın uzantısı ".jad" dır.

MIDlet'lerin takımların bir parçası olarak kullanılması, bazı yönlerden avantaj sağlar. Örneğin, bir takımın içindeki MIDlet'ler cihazdaki kaynakları paylaşabilir.

Bir MIDlet takımı oluşturmak için birden fazla sayıda MIDlet sınıfına gerek vardır. (Örneklerde basit olması açısından "MerhabaDunya1" ve "MerhabaDunya2" MIDlet'leri kullanılmaktadır.) Öncelikle, MIDlet'ler ayrı ayrı derlenmeli ve doğrulama işleminden geçirilmelidir.

```
>javac -g:none -bootclasspath %MIDPClasses% MerhabaDunya1.java
>%MIDPTools%\preverify-classpath%MIDPClasses%;. MerhabaDunya1
>javac -g:none -bootclasspath %MIDPClasses% MerhabaDunya2.java
>%MIDPTools%\preverify-classpath%MIDPClasses%;. MerhabaDunya2
```

Komut satırları işletildikten sonra MIDlet takımı oluşturmak için hazır duruma gelinmektedir.

3.1.7. MIDlet takımı tanımlayıcı dosyası

İlk adım, MIDlet takımı için tanımlayıcı bir dosya oluşturmaktır. Cihazdaki Java Uygulama Yöneticisi, uygulamanın yaşam döngüsünü yönetirken bu tanımlayıcı dosyayı kullanır. Java Uygulama Yöneticisi, uygulamaların indirilmesi, yüklenmesi, çalıştırılması ve silinmesinden sorumludur veya bu işlemlerin yapılmasına katkıda bulunur.

Tanımlayıcı dosya da Java kaynak dosyaları ile aynı dizine kaydedilmelidir. Aşağıda örnek bir tanımlayıcı dosyanın içeriği verilmiştir:

MIDlet-Name: MerhabaDunyaTakimi
MIDlet-Version: 1.0.0
MIDlet-Vendor: Ege Universitesi, Bilgisayar Muhendisligi
MIDlet-Description: Ornek bir MIDlet takimi
MIDlet-Info-URL: http://www.ege.edu.tr/
MIDlet-Jar-URL: http://localhost/merhaba.jar
MIDlet-Jar-Size: 3000
MicroEdition-Profile: MIDP-1.0
MicroEdition-Configuration: CLDC-1.0
MIDlet-1: Merhaba1, , MerhabaDunya1
MIDlet-2: Merhaba 2, , MerhabaDunya2

3.1.8. MIDlet takımlarının JAR dosyası haline getirilmesi

Bir MIDlet takımı için oluşturulan JAR dosyası bir "manifest" içermelidir. "Manifest", JAR dosyasının nasıl konfigüre edildiği, güvenlik bilgileri ve MIDlet'in ne içerdiği bilgilerini çalışma zamanı ortamına sağlar. J2ME çalışma zamanı ortamı, MIDlet takımını yüklemeden önce önlem olarak "manifest" ile tanımlayıcı dosyayı karşılaştırır [10].

"MIDlet-Name", "MIDlet-Version" ve "MIDlet-Vendor" değişkenleri, hem "manifest"te hem de tanımlayıcı dosyada aynı değerlere sahip olmalıdır.

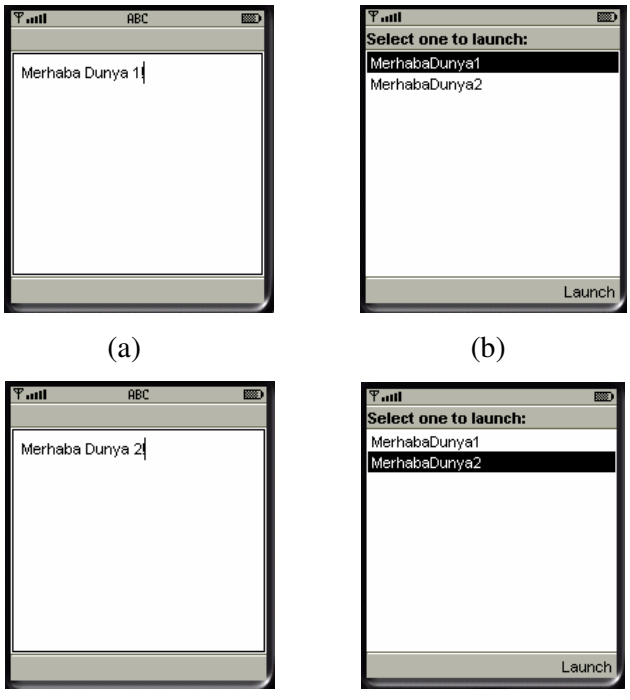
"Manifest" oluşturmak için "jad" uzantılı tanımlayıcı dosya "jar" komutuna parametre olarak verilir.

>jar -cfm merhaba.jar MerhabaDunyaTakimi.jad -C ./output
MerhabaDunya1.class –C ./output MerhabaDunya2.class

JAR dosyası oluşturulduktan sonra MIDlet takımının çalıştırılması için aşağıdaki komut satırı işletilmelidir.

>%MIDP%\bin\midp -classpath %MIDPClasses%; .\merhaba.jar -descriptor
MerhabaDunyaTakimi.jad

MIDlet takımı çalıştırıldığında oluşan ekran görüntüleri aşağıdaki Şekil 3.3'te verilmiştir.

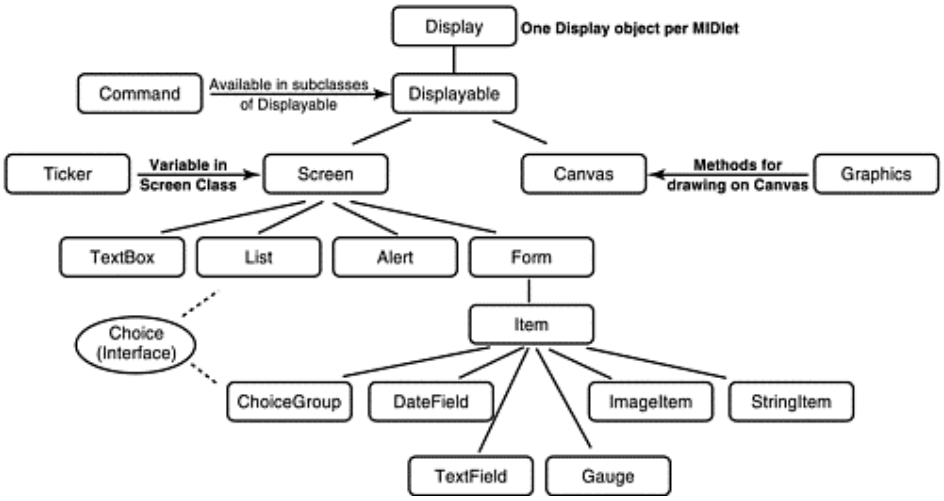


Şekil 3.3: MIDlet takımı çalıştırıldığında oluşan ekran görüntüleri

3.2. MIDP Kullanıcı Arayüz APIleri

Kaynakları kısıtlı olan cihazların, bellekleri ve ekran boyutları kısıtlı olduğundan görüntüleme kapasiteleri de kısıtlıdır. Bu nedenden, MIDP uygulamalarında kullanıcı arayüzü oluşturulurken J2SE'deki AWT sınıfı kullanılamaz. MIDP kendi kullanıcı arayüzü API'sine sahiptir.

MIDP düşük seviyeli ve yüksek seviyeli olmak üzere iki tane kullanıcı arayüz API'si içerir. Düşük seviyeli API "Canvas" soyut sınıfına dayanırken; yüksek seviyeli API'nin kullandığı "Alert, Form, List, TextBox" sınıfları "Screen" soyut sınıfına dayanır. Şekil 3.4 de MIDP kullanıcı arayüz tiplerinin sınıf diyagramı görülmektedir.



Şekil 3.4: MIDP kullanıcı arayüz tiplerinin sınıf diyagramı

3.2.1. MIDP ekran kontrolü

MIDP'de cihazın ekran ve görüntüsünü yöneten "Display" nesnesi vardır. "Display" nesnesi, düşük veya yüksek seviyeli API'ler için kullanıcı arayüzü elemanlarını ekrana çizmek ve bunları ekranda görüntülemek için gereken metodları içerir

Cihazın ekranında kullanıcı arayüz nesnelerinin gösterilebilmesi için, bir "Displayable" nesnesi diğer gösterilmek istenen kullanıcı arayüz nesnelerini içermelidir. Belirli bir zamanda, sadece bir tane "Displayable" nesnesi gösterilebilir. Belirli bir zamanda gösterilen "Displayable" nesnesinin hangisi olduğu "getCurrent()" metodu ile öğrenilebilir. Ekranda gösterilen nesne ise "setCurrent(Displayable birSonrakiEkran)" metodu kullanılarak değiştirilebilir.

3.2.2. Yüksek seviyeli kullanıcı arayüz APIsi

"Screen" sınıfı, tüm yüksek seviyeli "Displayable" nesnelerinin süper sınıfıdır. Alt sınıfları iki farklı türde olabilir. Birinci türde, kullanıcı arayüz elemanı önceden tanımlanmış bir tiptedir ve kullanıcıya bilgi görüntülemek için kullanılır. İkinci türde ise, uygulamalar ekrandan grafiksel eleman eklenmesine veya çıkarılmasına izin verirler (bir "Form" nesnesindeki metin kutusuna ad_soyad girilmesi gibi) [11].

3.2.2.1.Form

Bir "Form", "items" adı verilen onay kutusu, seçenek kutusu, yazı alanı gibi grafiksel elemanları içerebilir. Uygulamalar, "Displayable" nesnesinin içeriğini ancak görünür olmadıkları zaman değiştirebilir.

3.2.2.2.Alert

"Alert", bir mesaj çeşididir. Kullanıcıya bilgi vermek için kullanılır. Cihaz tarafından belirlenmiş süre kadar bu mesaj ekranda görüntülenir. Cihaz tarafından belirlenen süreyi kullanmak yerine, uygulama içerisinde mesajın görüntüleneceği süre milisaniye cinsinden belirtilebilir.

3.2.2.3.List

"List" kullanıcıya birden çok seçenek listeler ve kullanıcının bu seçeneklerden birini seçmesine izin verir. Kullanıcı seçim yapmadan önce, elemanlar arasında geçiş yapmasına da izin verilir.

3.2.2.4.Choice Interface

Çoktan seçmeli alanlardaki kısıtları belirler. Üç türde seçim yapılmasını sağlar. "EXCLUSIVE" seçimde sadece bir tane eleman seçilebilir. "MULTIPLE" seçim, birden fazla eleman seçileceğinde kullanılır. "IMPLICIT" seçim ise, bir "Command" nesnesi başlatıldığında ona odaklanan seçimdir.

3.2.2.5.TextBox

Kullanıcının yazı girmesi, için kullanılan bir metin alanıdır. En fazla girilebilecek karakter sayısı veya yapılabilecek olan giriş türü uygulama tarafından kısıtlanabilir. Uygulama içinde metin kutusu aşağıdaki gibi tanımlanabilir:

metinKutusu = new TextBox("", "Merhaba Dünya!", 20, 0);

İlk parametre metin kutusunun etiketini oluşturur. İkinci parametre metin kutusu ekranda görüntülendiğinde, ekrana yazılacak olan yazıdır. Üçüncü parametre metin kutusuna girilebilecek olan en fazla sayıdaki karakteri belirler. Son parametre ise, girdi kısıtlarını belirler. Girdi kısıtı, girilen karakterlerin "*" ile maskelenmesi şeklinde olabilir.

3.2.2.6.Ticker

"Ticker", ekranda bir yazının sürekli olarak kaymasını sağlar. "Screen" sınıfının bir metodu olan "setTicker(Ticker ticker)" metodu ile "Ticker" örneğinin ilgili "Screen" örneği ile ilişkilendirilmesi işlemini gerçekleştirir.

3.2.2.7.Items

Bir "Form" nesnesine eklenebilecek interaktif grafiksel elemanlar için kullanılan bir süper sınıftır.

"ChoiceGroup" nesnesi, bir grup seçenek veya elemandan oluşur. Tek elemanlık seçim yapılabileceği gibi birden çok elemanın seçildiği tipte bir seçime de izin verilir. Radyo düğmeleri tekli seçimi desteklerken, kontrol kutucukları da çoklu seçimi destekler.

"DateField" nesnesi, "Form" nesnesinde tarih ve saat bilgilerinin görüntülenmesi için kullanır.

"Gauge" nesnesi, ekranda belirli değerlere göre çubuk grafiği çizilmesini gerçekleştirir.

"StringItem" nesnesi, kullanıcıya metinsel bilgileri görüntülemede kullanılır.

"TextField" nesnesi, "Form" nesnesi içindeki metin editörüdür.

3.2.3. Düşük seviyeli kullanıcı arayüz APIsi

Düşük seviyeli kullanıcı arayüz APIsi kullanılarak arayüz oluştururken, çizim yapmak ve ekranı yenilemek için piksel koordinatları, yazı boyutu ve geometrik şekiller gibi birçok düşük seviyeli detayla ilgilenilmelidir.

3.2.3.1.Canvas

"Canvas" sınıfı "Displayable" sınıfının, düşük seviyeli kullanıcı arayüzü oluşturmak için bulunan tek alt sınıfıdır.

“Canvas” tüm ekranı çizmeyi sağlayan bir bileşendir, “Canvas” ile tüm ekran kontrol altına alınabilir. Bu nedenle, “Canvas” alt sınıfı genellikle oyun uygulamaları yazılırken kullanılır.

Yüksek seviyeli API’lerde ekranın yenilenme işlemi tüm “Screen” nesneleri için otomatik olarak yapılır; ancak, “Canvas” sınıfında ekran yenilenmesi “paint(Graphics g)” metodu kullanılarak yapılır. Bu metod çağrıldığında bir “Graphics” nesnesi oluşturulur.

3.2.3.2.Graphics

“Graphics” nesnesinden oluşturulan bir örnek ile ekrandaki tüm çizimler gerçekleştirilir. Çizim işlemleri, ekranın piksel piksel boyanması ile gerçekleştirilir.

3.2.3.3.Images

Bir “Image” nesnesi, grafiksel resim verisini tutar. Ancak, uygulama tarafından çağrıldığında çizim yapabilir. “Graphics” nesnesinde bulunan, “drawImage(Image img, int x, int y, int anchor)” metodu kullanılarak “Canvas” sınıfına gösterilir.

“Sabit” ve “Değişebilir” olmak üzere iki tip “Image” vardır. Sabit “Image”lar, bir kaynaktan gelen resim verisinin yüklenmesi ile oluşturulur. Sabit “Image”lar, sadece yüksek seviyeli kullanıcı arayüzü bileşenleri ile kullanılabilir. Değişebilir “Image”lar ise genelde düşük seviyeli bileşenlerle kullanılır.

3.2.3.4.Fonts

“Fonts” bileşeni; düşük seviyeli kullanıcı arayüzünde, ekrana çizilmiş herhangi bir yazının boyutunu belirler.

3.2.4. Düşük Seviyeli Kullanıcı Arayüzü Örneği

```
import javax.microedition.lcdui.*;
class CanvasDemo extends Canvas {
   protected void paint(Graphics g){
        /***Ekranda (1,1) koordinatlarından başlayarak "CanvasDemo" yazısı
        yazdırılmaktadır.***/
        g.drawString("Canvas Demo",1,1,Graphics.TOP|Graphics.LEFT);
        /***Ekrana verilen parametrelere bağlı oalrak içi dolu bir
        dikdörtgen çizilmektedir.***/
        g.fillRect (20,30,30,20);
        /***Aşağıdaki dört satırda, verilen koordinatlara göre dikdörtgen
        oluşturmak amacıyla dört çizgi ekrana çizilmektedir.***/
        g.drawLine(50,50,75,50);
        g.drawLine(75,50,75,75);
        g.drawLine(75,75,50,75);
        g.drawLine(50,75,50,50);
   }
}
```

3.3. MIDP'de Kullanıcı Etkileşimlerinin Yönetilmesi

Bir uygulama çalışırken, kullanıcının bu sırada cihazla olan etkileşimi olay olarak adlandırılmaktadır. Bir düğmeye basılması, seçenekler arasından bir tanesinin seçilmesi veya metin alanına bilgi girilmesi bir olayı temsil eder. Uygulamalar, olayları beklemek ve gerçekleşen olaylarda ilgili işlemi yapmak için oluşturulur.

Uygulamalar, olabilecek olan olayları "listener interface" gerçekleştirerek dinlerler ve bu olaylara bağlı olarak ilgili işlemleri yapacak olan "callback" metodlarını gerçekleştirirler. "Callback" metodları; genelde uygulama tarafından çağrılmayan, sistem tarafından özel bir olay için uyandırılan metodlardır.

Kullanıcı, uygulamanın dinlemekte olduğu belirli bir olayı tetiklediğinde cihaz otomatik olarak olayla ilişkilendirilmiş olan "callback" metodunu çağırır.

MIDP profili, yüksek ve düşük seviyeli olarak iki türde olay yönetimine sahiptir. Yüksek seviyeli olay ve olay yönetim mekanizmaları daha soyuttur. Geleneksel iş uygulamalarının ihtiyaçlarını karşılarlar. Yüksek seviyeli olaylar, MIDP profilin çalıştığı farklı cihazlar için taşınabilirdir. Düşük seviyeli olaylar ve olay yönetim mekanizmaları ise, basit olayları yakalar ve yönetir. Basit olaylar; basılan belirli bir tuşun yakalanması veya imlecin sürüklenmesi gibi olaylar olabilir. Düşük seviyeli olaylar, cihazlara özgüdür ve yüksek seviyeli olaylara göre daha az taşınabilirdir.

Olayların yönetimi aynı anda, paralel olarak yapılamaz. Ancak, MIDP'deki "Timer" nesnesi, işlerin belirli bir sıra dahilinde, arka planda çalışan bir iş parçacığı tarafından yapılmasına olanak tanır. "Timer callback" metodları, "callback" metodları ile aynı zamanda çalışabilir [12].

3.3.1. Yüksek seviyeli kullanıcı etkileşimlerinin yönetilmesi

Yüksek seviyeli API'de, olaylar iki tip kullanıcı etkileşimi sırasında gerçekleşir. Olaylar, kullanıcı bir formdaki bir değeri değiştirdiğinde veya bir komutu tetiklediğinde oluşabilir. Yüksek seviyeli olaylar için iki tip "listener" bulunmaktadır.

3.3.1.1.ItemStateListener

Bir "item"ın değeri değiştiğinde "itemStateChanged(Item i)" metodu, "callback" metodu olarak çağırılır. "ItemStateListener", bir formdan gelen olaylar için kullanılır.

3.3.1.2.Komutlar (Commands)

MIDP'de kullanıcıların, bir düğmeye basmaları veya menüden bir seçeneği seçmeleri "Command" nesnesini oluşturur.

"Command", üç bölümden oluşur. Bu bölümler; etiket, tip ve önceliktir. Etiket, cihaz tarafından, komutu ekranda görüntülerken kullanılır. Bir komutun tipi, onun anlamını belirler. Komutun tipi; "GERİ", "İPTAL", "YARDIM", "TAMAM", "EKRAN" veya "DUR" olabilir. Komutlar, önem sırasına göre değerlendirilirler. Cihaz komutları değerlendirirken önem sıralarını, öncelik değerlerine bakarak belirler.

3.3.1.3.CommandListener

Komut olaylarını yönetmek için kullanılır. Öncelikle "CommandListener", komut nesnesini bulunduran "Displayable" nesnesi ile ilişkilendirilmelidir. İlişkilendirme işlemi, "setCommandListener(CommandListener cmdListener)" metodu ile yapılır.

3.3.2. Düşük seviyeli kullanıcı etkileşimlerinin yönetilmesi

Düşük seviyeli API, kullanıcıya daha fazla kontrol verilmesi gereken oyun uygulamalarında kullanılır.

3.3.2.1.Tuş kodları ve düşük seviyeli API olayları (key codes and low-level API events)

Düşük seviyeli API'de bir tuşa basılması, uygulama tarafından bir tuş kodu ile bildirilir.

3.3.2.2.Olay dağıtım metodları (event delivery methods)

Yüksek seviyeli API'de olay yönetimi "listener" nesneleri aracığıyla gerçekleştirilir. Ancak; düşük seviyeli API'de "listener" nesnesi yoktur.

"Canvas" sınıfı, olay dağıtım metodlarına sahiptir. Bu metodlar; "showNotify()","hideNotify()" ve "paint(Graphics g)" metodlarıdır.

"showNotify()" metodu, "Canvas" görüntülenmeden önce çağrılır. "hideNotfy()" metodu, "Canvas" görüntüden kaldırıldığında çağrılır.

Olay dağıtım metodları seri olarak çağrılabilir. Aynı anda birden fazla metod çağrılamaz.

3.3.3. Etkileşimlerin yönetildiği yüksek seviyeli kullanıcı arayüzü örneği

AnketMIDlet sınıfında, EntryForm tipinde bir form oluşturulur. Bu form "displayMngr.setCurrent(entryForm)" komut satırı kullanılarak mobil telefonun ekranında görünür hale getirilir(Şekil 3.5 (a)). Kullanıcı Şekil 3.5 (b)'deki gibi ilgili alanlara adını, yaş grubunu ve hobilerini girer. Daha sonra ekranın sağ tuşunun yönettiği "Get" komutuna basıyor. Bu tuşa basınca kullanıcının seçtiği bilgilerin görüntülendiği bir "Alert" oluşturulur ve Şekil 3.5 (c)'deki gibi ekranda görüntülenir. Kullanıcı bu mesajı ekranda gördükten sonra ekranın sağındaki "Done" tuşuna basınca, Şekil 3.5 (a)'daki gibi boş bir form ekranda tekrar görüntülenir. Kullanıcı bu ekranın solunda "Exit" komutunu yöneten tuşa bastığında uygulamayı sonlandırır [13].

AnketMIDlet.java

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class AnketMIDlet extends MIDlet {
String sym = new String();     // textfielda girilen ismi tutan değişken
String soru2 = new String(); // seçilen yaş grubunun tutulduğu değişken
String soru3;                  // seçilen hobileri tutan değişken
//uygulamanın hayat devresini tutması için bir lokal değişken tanımlanır.
private Display displayMngr = null;
//EntryForm sinifindan bir degisken tanimlaniyor.
private EntryForm entryForm = null;
//Alert nesnesine bir referans degiskeni tanimlaniyor.
private Alert resultsAlert = null;
public AnketMIDlet () { }
```

```
private void initListener () {
CommandListener commandListener = new CommandListener() {
        public void commandAction(Command c, Displayable d) {
                soru3 = new String();
                if (c == entryForm.getExitCommand()) {
                        destroyApp(true);
                } else if (c == entryForm.getGetCommand()) {
                  if (entryForm.getcevap1Field().getString().length() > 0) {
                         sym = entryForm.getcevap1Field().getString();
                  }
                  if (entryForm.getsoru2Choice().getSelectedIndex() == 0){
                        soru2 = "Genc";
                  }
                  else {
                        soru2 = "Orta Yasli";
                      }
                  if (entryForm.getsoru3Choice().isSelected(0)){
                        soru3 = "\n *muzik";
                  }
                  if (entryForm.getsoru3Choice().isSelected(1)){
                        soru3 = soru3+"\n *resim";
                  }
                  if (entryForm.getsoru3Choice().isSelected(2)){
                        soru3 = soru3+"\n *sinema";
                  }
                  if (entryForm.getsoru3Choice().isSelected(3)){
                        soru3 = soru3+"\n *tenis";
                  }
                  if (entryForm.getsoru3Choice().isSelected(4)){
                        soru3 = soru3+"\n *yuzme";
                  }
                 displayInformation("Hosgeldin " + sym + " !\n"+ soru2 +"
```

```
                grubundasin. \n" + "Hobilerin: "+ soru3);
                } //END OF else if (c == entryForm.getGetCommand())
        } //END OF public void commandAction
        };
        entryForm.setCommandListener(commandListener);
        }
        //tanimlanan EntryFormun ekranda gosterilmesi icin bir metod
        private void displayEntryForm () {
        if (entryForm == null) {
                entryForm = new EntryForm("Anket");
        }
        initListener(); //o an display managerin EntryFormu gostermesi
        //icin setCurrent kullaniliyor
        displayMngr.setCurrent(entryForm);
}
        //sonsuza kadar ekranda kalabilecek olan alert icin
        //bir metod olusturuluyor
 private void displayInformation(String quoteString) {
        if (resultsAlert == null) {
                resultsAlert = new Alert("Anket Sonuclari", null, null,
        AlertType.CONFIRMATION);
        resultsAlert.setTimeout(Alert.FOREVER);
        }
        resultsAlert.setString(quoteString);
         //o an Alert in gosterilmesini, ancak bir sonraki ekran olarak
         //EntryForm'a donulmesi tanimlaniyor
         displayMngr.setCurrent(resultsAlert, entryForm);
         entryForm.setcevap1Field();
         entryForm.setsoru2Choice();
        entryForm.setsoru3Choice();
}
```

```
    //Display instance alinir ve referansi displayMngr degiskeni içine
koyulur
    //entryForm sinifi olusturuldugunda displayEntryForm metodu cagriliyor
    protected void startApp() {
        displayMngr = Display.getDisplay(this);
        displayEntryForm();
    }
    protected void pauseApp() {  }
    protected void destroyApp(boolean unconditional) {
        notifyDestroyed();
    }
    public void commandAction(Command c, Displayable s) {  }
} //END OF AnketMIDlet.java
```

EntryForm.java

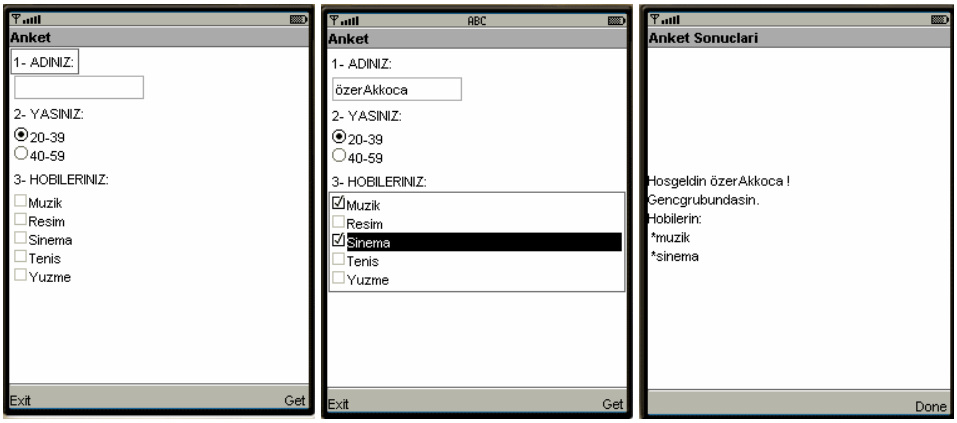
```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
//Form sinifi iki tane consructor a sahiptir.
//bunlarin en azindan birinin yeni sinifta overridden edilmesi gerekir
public class EntryForm extends Form {
// exit ve get komutlari icin referans degiskenleri tanimlaniyor
private Command exitCommand = null;
private Command getCommand = null;
// kullanıcı isminin tutulduğu bileşenler
private StringItem soru1Field = null;
private TextField cevap1Field = null;
// yas grubunu belirleyen bileşenler
private StringItem soru2Field = null;
private ChoiceGroup soru2Choice = null;
// hobilerin tutulduğu bileşenler
private StringItem soru3Field = null;
```

```
private ChoiceGroup soru3Choice = null;
public EntryForm(String title) {
super(title);
soru1Field = new StringItem(null, "1- ADINIZ:");
//textField tanimlaniyor.
//1. parametre -> textfield'in etiketini olusturuyor.
//2. -> "" oldugundan baslangic degeri verilmiyor.
//3. -> kullanici en fazla 10 karakter girebilir.
//4. -> bu alana giris yaparken herhangi bir kisit olmadigi belirtiliyor.
cevap1Field = new TextField("", "", 10, TextField.ANY);
soru2Field = new StringItem(null, "2- YASINIZ:");
String choices_yas[] = {"20-39", "40-59"};
soru2Choice = new ChoiceGroup("", Choice.EXCLUSIVE,
choices_yas, null);
soru3Field = new StringItem(null, "3- HOBILERINIZ:");
String choices_hobi[] = {"Muzik", "Resim", "Sinema",
"Tenis", "Yuzme"};
soru3Choice = new ChoiceGroup("", Choice.MULTIPLE,
choices_hobi, null);
// exit ve get komutlari olusturuluyor,
// exit komutuna en yuksek oncelik veriliyor
exitCommand = new Command("Exit", Command.EXIT, 1);
getCommand = new Command("Get", Command.SCREEN, 2);
// tanımlanan ekran bileşenleri forma ekleniyor
 append(soru1Field);
 append(cevap1Field);
 append(soru2Field);
 append(soru2Choice);
 append(soru3Field);
 append(soru3Choice);
//commandler forma ekleniyor
 addCommand(exitCommand);
```

```
 addCommand(getCommand);
 }
// anket sorularının cevaplarının alınabilmesi ve gerektiğinde ilgili
// alanların boşaltılması için getter ve setter metodlar tanımlanıyor
 public StringItem getsoru1Field() {
        return soru1Field;
 }
 public StringItem getsoru2Field() {
         return soru2Field;
 }
 public StringItem getsoru3Field() {
        return soru3Field;
 }
 public TextField getcevap1Field() {
        return cevap1Field;
 }
 public void setcevap1Field() {
        cevap1Field.setString("");
 }
 public ChoiceGroup getsoru2Choice() {
         return soru2Choice;
 }
 public void setsoru2Choice() {
         soru2Choice.setSelectedIndex(0,true);
 }
 public ChoiceGroup getsoru3Choice() {
         return soru3Choice;
 }
 public void setsoru3Choice() {
         soru3Choice.setSelectedIndex(0,false);
         soru3Choice.setSelectedIndex(1,false);
         soru3Choice.setSelectedIndex(2,false);
```

```
            soru3Choice.setSelectedIndex(3,false);
            soru3Choice.setSelectedIndex(4,false);
        }
       //commandlere erisebilmek icin getter metodlar tanimlaniyor
        public Command getExitCommand() {
            return exitCommand;
        }
        public Command getGetCommand() {
            return getCommand;
        }
       }
```

Uygulama çalıştırıldığında Şekil 3.5' dekine benzer bir görüntü meydana gelir.



(a) (b) (c)

Şekil 3.5: AnketMIDlet uygulaması çalıştırıldığında oluşan ekran görüntüleri

4. MIDP VERİ DEPOLAMA

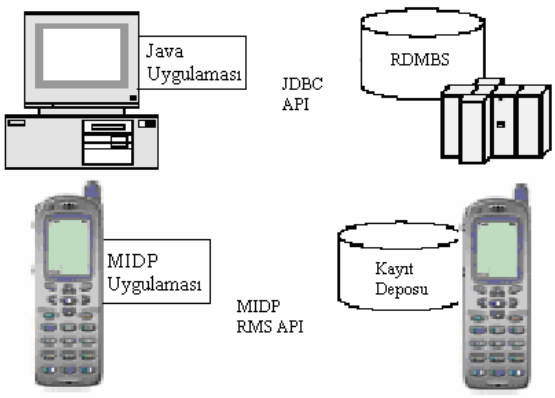
Büyük bir mainframe platformunun bir veritabanını yerleştirmek için gerekliliği bir süre öncedir. Ancak, bir mainframe üzerine sıkıştırılmış tüm dünyadaki veriler işini uzaktan yapan bir mobil ve/veya bağlantısız kullanıcıya yardımcı olamamaktadır. Uzak yerleşimlerdeki taşınabilir verinin gerekliliği daha küçük ve mobil veritabanı sistemlerinin geliştirilmesini teşvik etmiştir. Bugün, nispeten geniş veritabanları bir dizüstü bilgisayar üzerinde bulunabilmektedir. Eğer farklı sebepler yoksa, bu veritabanları daha büyük veritabanlarının bir alt kümesini taşırlar.

J2ME platformları ve özellikle MID profili, daha küçük ve mobil cihazlar için doğal ilerleyiş içindeki bir sonraki adımlardır. Bu nedenle bu cihazlar bazı kalıcı veri depoları ile donatılmıştır. Bir dizüstü bilgisayar üzerine sığdırılan veritabanları günümüzde daha gelişmiş olabilmektedir. Bu veritabanları otuz yıl öncesinin ilişkisel veritabanı yönetim sistemleri (RDBMS) ile rekabet etmektedirler. Tam bir RDBMS'in şu an MIDP platformunun fiziksel limitleri yüzünden çalıştırılmasının zor olurken, MIDP API küçük de olsa mobil cihazlar için kısıtlı veri deposu sağlamıştır. Şimdi, kullanıcılarının ceplerine veritabanlarının küçük alt kümeleri yerleştirilebilmektedir. MIDP'deki kalıcı veri depolama mekanizmasının ismi Kayıt Yönetim Sistemi (Record Management System - RMS)'dir [14].

4.1. JDBC İle İlişkisi

Diğer Java sürümleri ile yazılmış uygulamalarda, J2SE ve J2EE gibi, "third party" üreticiler genellikle veri depolama mekanizmalarını kullanıcılara sağlarlar. Bu sürümlerde, çeşitli üreticiler tarafından sağlanmış veritabanlarına ortak bir erişim ve veri işleme için bir Java API'sinin sağlanması gerekmektedir. Tipik olarak API bir RDBMS'e erişimi sağlar. JDBC'nin diğer Java sürümü uygulamalarında yaptığı gibi, MIDP RMS API de MIDP cihazlar üzerindeki veriye erişim ve onu işleme için bir standart sağlamaktadır. (Şekil 4.1)

JDBC API çeşitli üreticilerin veritabanlarına ortak bir ulaşım sağlarken, MIDP RMS API platform gerçekleştiricilerinin basit byte dizisi depo mekanizmalarına ortak bir erişim protokolü sağlamaktadır.



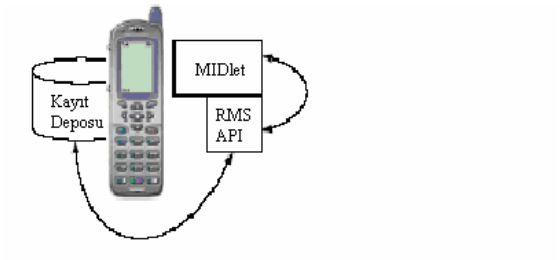
Şekil 4.1: MIDP uygulamalarının kayıt depolarına ulaşması

4.2. Depolama Yapısı

API, uygulamanın bölümlerinin nasıl gerçekleştirildiğinin belirtilmesine gerek kalmadan kullanıcı ve sağlayıcı taraflara bir anlaşma sunmaktadır. JDBC gibi, MIDP RMS API bir komut seti sağlamaktadır, bu durumda mobil telefon ve çağrı cihazı uygulamaları her ne kadar küçük miktarda da olsa veri depolayabilmektedirler. Aynı zamanda, RMS mobil telefon ve çağrı cihazı üreticilerine bu veriyi nerede ve nasıl saklayacakları konusunda özgürlük sağlamaktadır. Çünkü platform geliştiriciler kalıcı veri mekanizmasını sağlamada görev alırlar, uygulamaların verilere erişimini sağlayacak veritabanı sürücüsü veya diğer "third party" yazılımlara gerek duyulmaz. Dahası, "higher-end veritabanı sistemlerinde bulunan "heavy-duty" eş zamanlılık sağlamaları, sonuç seti ele alma, veritabanlarının basitliği ve sorgu dillerinin azlığı ile, RMS, MIDP platformu üzerinden veri depolama ve geri alma için sadece bir tane ana sınıfa ihtiyaç duymaktadır, bu sınıfın ismi RecordStore'dur.

4.2.1. Kayıt deposu

Kayıt Yönetim Sistemi'nin kalbinde kayıt deposu vardır. Bir kayıt deposu kayıtlar topluluğudur ve bir kayıt bir byte dizisidir. Platform geliştirici ve aynı zamanda RMS gerçekleştirici, cihaz üzerinde bir kayıt deposunun yerini belirler. Ancak cihaz üzerinde fiziksel olarak bu kayıt deposunun nerede bulunduğunun bir önemi yoktur, çünkü bir MIDlet tarafından doğrudan erişilemez. MIDP uygulamaları sadece sağlanan bir API üzerinden RMS'e erişmektedir. (Şekil 4.2)



Şekil 4.2: MIDletin kayıt deposuna ulaşmak için RMS API kullanımı

Cihazın "normal" işlemlerini kayıt deposunun yürütebilmesi işi de platform geliştiricinin görevidir. Bu operasyonlar kapanmaları, tekrar başlamaları ve batarya değişimleri içermektedir.

Kayıt depoları bir MIDlet takım ile ilişkilidir. Öyle ki, kayıt deposundaki veriler bir takım içindeki herhangi sayıdaki MIDlet tarafından paylaşılıyor ve kullanılıyor olabilmektedir. Gerçekten, aynı kayıt deposuna erişim için 2 ayrı MIDlet istekte bulunduğunda, her birine sistemdeki aynı kayıt deposuna bir nesne referansı verilmektedir. MIDlet takımı cihazdan kaldırıldığında, takım ile ilişkilendirilmiş tüm kayıt depoları da cihazdan kaldırılmalıdır. Her kayıt deposu, MIDlet takımı için de geçerli olan, "unique" bir isme sahip olmalıdır. Kayıt deposu isimleri 32 karakter (Unicode karakter olarak) uzunluğunda ve "case-sensitive" olabilmektedir. Eğer bir MIDlet bir takımın bir parçası değilse, bu tek MIDlet'i içeren bir sanal takımdır. Bu durumda, MIDlet kayıt deposunun sahibi olmaktadır. MIDlet platformdan kaldırıldığında, kayıt deposu da kaldırılır. Kayıt deposu her değiştirilişinde, platform veritabanını "long integer" formatında bir tarih ve zaman damgasıyla kaydetmektedir. System.currentTime-Millis()

çağrısından dönen “long integer” ile bir kayıt deposu kaydedilmektedir. Bu metod çağırımı 1 Ocak 1970 12:00 a.m. tarihinden itibaren geçen milisaniyelerin sayısını döndürmektedir. Platform, veritabanı değiştikçe yeni bir versiyon numarası vererek kayıt deposunu kaydetmesi gerekmektedir. Bu versiyon numarası her değişiklik ile birlikte artmaktadır. RMS gerçekleştirici ilk versiyon numarasına karar vermektedir, ancak bu değer 0’dan büyük olmalıdır. Tarih/zaman ve versiyon damgası platforma veritabanının senkronizasyonunda yardımcı olmaktadır, fakat bunlara uygulama tarafından erişilebilmekte ve kullanılabilmektedir.

RMS’in birçok gerçekleştirim yönüyle birlikte, platform gerçekleştirici atomik, senkronize ve serileştirilmiş bir şekilde kayıt deposuna erişimi sağlamalıdır, çoklu erişimlerde bile veritabanının bozulmamasını garantilemelidir. MIDP uygulamaları ayrı kayıtları veya tüm kayıt deposunu RMS API üzerinden kilitleyemez. Platform gerçekleştirici verilerin entegrasyonunu garantileyebilmelidir, yani birçok sayıda “thread” kullanan bir MIDP uygulamasına, kayıt deposunu güncellerken bir önceki “thread” tarafından sağlanan verinin üzerine yazılmasından kaçınılması için özel ilgi gösterilmelidir. Mesela, iki “thread”, A ve B, veri işlemekte ve ikisi de X kaydını güncellemeye çalışsın, kayıt deposu gerçekleştirimi bir sistem hatası veya veri bozulması olmadan X’i her iki thread’in de güvenli bir şekilde güncelleme yapmasını garantilemektedir. Gerçekten, kayıt deposuna yapılan çağrımlar eşzamanlı erişimlerden sakınmak için serileştirilmiştir. Bu örnekte, A’nın X kaydını güncellemesine izin verilmekte ve sonra B’ye X kaydını güncellemesine izin verilmektedir [15].

4.2.2. Kayıt deposunda bulunan kayıtlar

Bir kayıt basit bir byte dizisidir. Kayıt deposundaki her kaydın bir “unique” tamsayı tanımlayıcısı bulunmaktadır, recordId. Kayıt deposu içine yaratılan ilk kaydın recordId değeri 1’dir. Kayıt deposuna eklenen her kayıt için recordId artırılmaktadır. Bir kayıt deposunu tabloya benzettiğimizde, bir byte bir

hücre ile temsil edilmektedir ve her byte dizisi kayıt deposu içinde recordId tarafından tanımlanmış hücre serileri olarak düşünülebilmektedir. (Çizelge 4.1)

Çizelge 4.1: Kayıt deposunda bayt dizilerinin temsili temsil edilmesi

Kayıt ID					
1	**bayt 0**	**bayt 1**	**bayt 2**	**.**	**bayt n**
2	**bayt 0**	**bayt 1**	**bayt 2**	**.**	**bayt n**
3	**bayt 0**	**bayt 1**	**bayt 2**	**.**	**bayt n**
.	**.**	**.**	**.**	**.**	**.**
.	**.**	**.**	**.**	**.**	**.**
.	**.**	**.**	**.**	**.**	**.**
n	**bayt 0**	**bayt 1**	**bayt 2**	**.**	**bayt n**

recordId, kayıt deposundaki çeşitli byte dizilerine bir çeşit indeks gibi düşünülebilmektedir. Ancak, recordId'ler bir kayıt, kayıt deposundan silindikten sonra tekrar kullanılamamaktadır. Bu nedenle recordId'leri gerçek bir indeks gibi görmek ne güvenli ne de uygundur.

recordId kayıt deposu içindeki belli bir kayda erişmek veya kontrol etmek için kullanılmaktadır. Ancak kayıtlara erişmek ve kayıtları kontrol etmek için sadece recordId kullanılmayabilir. Onun yerine bir enumerator kullanılabilir.

4.3. RMS APIsi

API, cihazlar üzerinde MIDlet uygulamalarının veri depolama ve geri alma işlemleri gerçekleştirmesine izin verir. javax.microedition.rms paketi MIDP Kayıt Yönetim Sistemi için gerekli tüm API'yi içermektedir. Bir kayıt sadece bir byte dizisi olduğundan, tüm paket içerisinde sadece bir tane somut sınıf bulunmaktadır, RecordStore. Bu sınıf RMS kayıt deposunun gerçekleştirimin yapan sınıftır.

4.3.1. Kayıt deposu yaratılması ve erişimi

RecordStore için olan API doğrudandır. Bir kayıt deposu yaratmak için basit bir statik metodu vardır ve depodaki ekleme, çıkarma ve güncelleme işlemleri için örnek metotlar bulunmaktadır. Kayıt deposunu yok etmek ve kayıt deposu dışından yardımcı bilgileri elde etmek için metotlar dışında, kayıt deposu içindeki kayıt sayısı gibi, API için fazla bir şey kalmamaktadır.

4.3.2. Kayıt deposu yaşam döngüsü

Bir kayıt deposu aynı metot ile yaratılmakta ve açılmaktadır, openRecordStore(String recordStoreName, boolean createIfNecessary), MIDlet takımındaki varolan bir kayıt deposunu, çalışan ilgili MIDlet'in açma girişiminde bulunduğunda kullanılan statik bir RecordStore metodudur. Sistem eğer aynı isimde bir kayıt deposu yoksa ve "createIfNecessary" mantıksal değeri "true" olarak set edildiyse, yeni bir tane kayıt deposu yaratmaktadır. Böylece, mesela "Customers" isminde daha önceden varolmayan yeni bir veritabanı oluşturmak için, aşağıdaki kod çalıştırılmaktadır.

```
try {
        RecordStore anRMS = RecordStore.openRecordStore("Customers"
        , true);
}
 catch (RecordStoreFullException fullStore) {
        //handle a full record store problem
}
 catch (RecordStoreNotFoundException notFoundException) {
//handle store not found which should not happen with the
//createIfNecessary tag set to true
}
 catch (RecordStoreException recordStoreException) {
//handling record store problems
}
```

closeRecordStore() metodu bir kayıt deposu örneğini kapatmak için kullanılmaktadır. Kayıt deposu, “open” metodunun çağrılma sayısı kadar “close” metodu çağrılmazsa gerçekte kapanmaz. Kayıt deposunu açmak ve kapatmak için yapılan MIDlet çağrımları sayısı tüm takım üzerinde izlenmektedir ve kayıt deposunun gerçekten kapanması için kapatılma sayısı ile açılma sayısı eşit olmalıdır. Bunun nedeni bu kayıt deposunun takım içinden birden fazla MIDlet tarafından paylaşılıyor olabilmesidir. Kayıt deposunun kapatılmış kabul edilebilmesi için, buna erişimi olan tüm MIDlet uygulamalarının kapatılmış olması gerekmektedir. Son olarak kapatıldığında, kayıt deposuna olan tüm dinleyiciler kaldırılmaktadır.

Bir MIDlet takımı için uygun olan kayıt depolarının bir listesi elde edilebilmektedir. Bir MIDlet uygulaması içinden, listRecordStores()statik metot çağırımı bir String dizisi geri döndürmektedir. Bu dizi, takım için hangi MIDlet ile ilişkiliyse ona ait kayıt deposu isimlerini içermektedir. Eğer MIDlet takımının hiç kayıt deposu yok ise, bu metot “null” döndürmektedir. RecordStore sınıfındaki bir statik metot kayıt deposunun yok edilmemesini sağlamaktadır. deleteRecordStore(String recordStoreName) metodu, takım için verilen isimdeki veri deposunu silmektedir, ancak bu kayıt deposu o takım içinde var olmalı ve takım içinden herhangi bir MIDlet tarafından o an için açılmış olmamalıdır. Bu son iki durum eğer “false” ise silme metodu tarafından RecordStoreException fırlatılmasına neden olmaktadır.

4.3.3. Kayıt erişimi

Kayıt operasyonlarının bir kayıt deposu üzerinde gerçekleştirilebilmesi için bu kayıt deposu örneğinin açılmış olması gerekmektedir. Aksi takdirde, kapalı bir kayıt deposuna ulaşmaya çalışırken bir RecordStoreNotOpen aykırı durumu fırlatılmaktadır. Var olan bir kaydı ya da byte dizisini veritabanından çekmek için bir recordId’ye gerek duyulmaktadır. Bir recordId ve getRecord(int recordId) metodu ile gönderilen bu id altında saklanan byte dizisi döndürülmektedir. Mesela, recordId’si 2 olan kayda ulaşmak için *anRMS*

tarafından referans edilmiş açılmış kayıt deposu üzerinden aşağıdaki kod çağrılmaktadır.

byte[] b = anRMS.getRecord(2);

Gönderilen recordId'ye ait bir kayıt olmaması durumunda ise, bir InvalidRecordIDException fırlatılmaktadır. getRecord() metoduna alternatif olarak bir offset belirterek bir kaydı doğrudan okumamızı sağlayan bir metot bulunmaktadır. getRecord(int recordId, byte[] buffer, int offset)metodu belirtilen recordId'deki kaydın içeriğini offset parametresindeki belirtilen yerdeki tamponun içine, yani byte dizisine geçirir. Ayrıca, bu metot birden fazla tipte aykırı durum fırlatabilmektedir, ArrayIndexOutOfBoundsException eğer kayıttan çekilen byte dizisi tamponun kabul edebileceğinden daha büyük ise fırlatılmaktadır.

Depo içindeki kayıtlar üzerinde ekleme, silme ve güncelleme işlemleri için bir kayıt deposunun birçok metodu bulunmaktadır. Bir kayıt deposu örneğine bir kayıt eklemek için addRecord(byte [] data, int offset, int numBytes) metodu kullanılmaktadır. İlk parametre olarak gönderilen byte dizisi kayıt deposu içine eklenecek veriyi göstermektedir. Bir ofset indeksi ve byte sayısı ise eklenecek byte dizisinin bir parçası olarak kullanılabilmektedir. Mesela, aşağıdaki kod parçasında, referans edilmiş kayıt deposuna yeni bir kayıt olarak "test" stringinin saklanmasını sağlamaktadır.

```
String test = "This is a test";
byte[] b = test.getBytes();
anRMS.addRecord(b, 8, 6);
```

Başarılı bir şekilde bu işlem tamamlandığında bu addRecord(byte [] data, int offset, int numBytes) metodu yeni eklenen kaydın recordId'sini döndürmektedir. Kayıt deposundan bir kaydın silinmesi işleminde ise deleteRecord(int id) metodu kullanılmaktadır, parametre olarak kaldırılacak kaydın recordId'si gönderilmektedir. Kaldırılmış kaydın tanımlayıcısı kayıt deposuna eklenecek sonraki ekleme işlemlerinde kullanılmamaktadır. Güncelleme

işlemi ise setRecord(int id, byte [] data,int offset, int numBytes) metodu ile gerçekleştirilmektedir. Kayıt deposundaki bir kaydı güncelleme belirli bir recordId'de saklanan tüm byte dizisinin değiştirilmesi şeklinde olmaktadır. Bu yüzden ekleme işleminde kullanılan argümanların aynısı kullanılmaktadır. Ofset indeksi ve byte sayısı yeni byte dizisinin tümü veya bir parçası ile varolan kaydın değiştirilmesi için kullanılmaktadır [16].

Ayrıca kayıt deposundan ek bilgiler çekmek için birçok uygun metot daha vardır ve bunlar Çizelge 4.2'de listelenmiştir.

Çizelge 4.2: Kayıt deposu metodları

Metod	Amaç
getLastModified()	Kayıt deposunda son değişiklik yapıldığı zamanı dönürür.
getName()	Kayıt deposu örneğinin ismini döndürür.
getNextRecordID()	Bir sonraki addRecord operasyonu için bir sonraki kayıt tanımlayıcısının tamsayı değerini döndürür.
getNumRecord()	Kayıt deposu örneğindeki kayıt sayısını döndürür.
getRecordSize(int recordID)	recordID ile tanımlanmış kayıtların bayt cinsinden büyüklüğünü döndürür.
getSize()	Kayıt deposunun byte cinsinden büyüklüğünü döndürür.
getSizeAvailable()	Kayıt deposunun büyüyebileceği maksimum büyüklüğü dönürür.
getVersion()	Kayıt deposu için en son versiyon numarasını döndürür.

Kayıt depolarını yaratma ve kaldırma veya bir kayıt deposu içindeki veriyi beceriyle kullanmak için kullanılan API doğrudandır. Ayrıca, çok basit veritabanlarında verileri yerleştirmek ve karşılaştırmak için yardımcı sınıflar da vardır.

4.3.4. Kayıt deposu aykırı durumları

Bir kayıt deposundaki verilerin başarıyla yönetilmesi ve RecordStore sınıfının kullanılması aykırı durumlara neden olabilmektedir. RecordStore sınıfı veya örneği ile uğraşırken oluşan herhangi bir bilinmeyen problem oluştuğunda fırlatılan genel bir aykırı durum vardır, RecordStoreException.

RecordStoreException'ın altında yer alan daha özel alt sınıflar ve fırlatılma olaylarının tanımları Çizelge 4.3'de listelenmiştir.

Tüm bu aykırı durumlar geliştiricinin yakalayıp ele alması gereken kontrol edilmiş aykırı durumlardır.

Çizelge 4.3: Kayıt deposu aykırı durumları

Aykırı Durum	Tanımı
InvalidRecordIDException	Referans edilen recordID bulunmuyorsa fırlatılır.
RecordStoreFullException	Kayıt deposu doluyken kayıt eklenmey veya güncellenmeye çalışşılırsa oluşur.
RecordStoreNotFoundException	Bulunmayan bir kayıt deposunu açmaya veya silmeye çalışıldığında oluşur
RecordStoreNotOpenException	Açık olamayan bir kayıt deposuna ulaşılmak istendiğinde fırlatılır.

4.3.5. Kayıt deposu dinleyicisi

Bir kayıt deposuna aynı takım içinden birçok MIDlet tarafından erişilebilmektedir. Uygulamaların kayıt deposu verilerindeki değişikliklere reaksiyon göstermesi ve koordinasyonunu sağlaması için, bir kayıt deposu olayları ele alma arayüzü kümesi sağlanmıştır. Bir MIDlet takımı içindeki herhangi bir kayıt deposuna yapılan değişiklikler için herhangi bir nesne bir dinleyici olarak ayarlanabilmektedir. Bu nesne sadece RecordListener arayüzünü gerçekleştirmelidir ve değiştirme işlemleri için bir dinleyici olarak bir kayıt deposu örneği ile birlikte kayıt edilmelidir.

Nesneyi, bir kayıt deposu üzerinde gerçekleşen değiştirme işlemleri için geçerli bir dinleyici olarak tayin etmek için önce kayıt deposu ile birlikte kayıt edilmiş olmalıdır. Bir listner'ın kayıt işlemleri için 2 metot kumlanılmaktadır.

1. addRecordListener(RecordListener listener)

2. removeRecordListener(RecordListener listener)

Bir kayıt deposu kapatıldığında, tüm dinleyiciler kaldırıldığında, bir dinleyicinin bir kayıt deposuyla kayıt edilmesinin bir etkisi yoktur.

RecordListener arayüzü üç metodu olan olay ele alma nesnesine gerek duymaktadır. Bunlar:

1. recordAdded (RecordStore recordStore, int recordId)

2. recordChanged (RecordStore recordStore, int recordId)

3. recordDeleted (RecordStore recordStore, int recordId)

Her "callback metodu" değiştirilmiş belirli kayıt deposu ve değişen, eklenen ve silinen kaydın recordId'si ile birlikte çağırılmaktadır. Bu metotlar; yeni bir kayıt, kayıt deposun eklendiğinde (recordAdd), varolan bir kayıt üzerinde değişiklik yapıldığında (recordChanged), veya depodan bir kayıt silindiğinde (recordDeleted) çağırılmaktadırlar.

Kayıt deposundaki kayıtlar eklendiğinde, değiştirildiğinde veya silindiğinde sadece raporlayan basit bir örnek dinleyici aşağıdaki kodda listelenmiştir.

```
import javax.microedition.rms.*;
public class TestListener implements RecordListener {
  public void recordAdded(RecordStore rs, int id) {
    try {
      System.out.println(rs.getName() + " added record " + id);
    } catch (RecordStoreNotOpenException e) {
//exception handling procedures
    }
  }
  public void recordChanged(RecordStore rs, int id) {
```

```
    try {
      System.out.println(rs.getName() + " changed record " + id);
    } catch (RecordStoreNotOpenException e) {
//exception handling procedures
    }
  }
  public void recordDeleted(RecordStore rs, int id) {
    try {
      System.out.println(rs.getName() + " removed record " + id);
    } catch (RecordStoreNotOpenException e) {
//exception handling procedures
    }
  }
}
```

Bu dinleyiciyi bir kayıt deposu örneği ile birlikte kayıt etmek için, aşağıdaki program kodu gerekmektedir.

```
RecordStore anRMS = RecordStore.openRecordStore("TestRMS" , true);
anRMS.addRecordListener(new TestListener());
```

İlk satır TestRMS isimli kayıt deposunu açmak için kullanılmaktadır. Metoda geçirilen "createIfNecessary" parametresi "true" mantıksal değeri olarak verildiğinde, kayıt deposu yaratılmakta ve sonra eğer daha önceden yoksa açılmaktadır. İkinci satır TestListener 'ın örneğini MIDlet takımı içinden herhangi bir MIDlet tarafından gerçekleştirilen kayıt değişikliklerine tepki vermesi için kayıt etmektedir.

RMS kayıt depolarının basit doğalarından dolayı, kayıt dinleyiciler MIDlet'lerin veri doğrulama mekanizmalarını daha kolay gerçekleştirmesine izin vermektedirler, "out of space" mesajları ve daha sofistike veritabanları tarafından otomatik olarak yakalanan diğer veri ile ilgili aktiviteler gibi uyarıları tetiklemektedir.

4.3.6. Kayıtları karşılaştırma

J2SE ortamında iki benzer nesneyi değerlendirmek veya karşılaştırmak için Comparable arayüzü bir karşılaştırma operatörü tanımlanmasını sağlamaktadır. Bu karşılaştırma işleminin sonucunda bir nesnenin diğer nesneye eşit olduğu, ondan daha büyük veya daha küçük olduğunun bulunması şeklinde sonuçlanmaktadır. Böylece mesela, sipariş veren müşteri numaraları soyadına veya sosyal güvenlik numarasına göre olabilmektedir.

"Comparable" arayüzü J2ME'de yok iken, Kayıt Yönetim Sistemi içinde bir kayıt deposunun kayıtlarının karşılaştırılmasına izin verecek eşdeğer bir arayüz sağlanmıştır. RecordComparator arayüzü herhangi bir nesneye verilen herhangi iki RMS kaydı için bir karşılaştırma kolaylığı kurulmasını sağlamaktadır. Gerçekten, bir RecordComparator gerçekleştiricisi iki byte dizisini karşılaştırmaktadır. Çünkü genel olarak iki byte dizisini kabul etmek ve karşılaştırmak için oluşturulmuştur, ve kayıt deposu kayıtlarının karşılaştırılması dışında kullanılmak için de ayarlanabilmektedir. Genelde, bir kayıt karşılaştırıcısı sayım amaçları için kayıt deposu kayıtlarının sıralama ve dizme işlemleri için uygundur.

RecordComparator arayüzü girilen iki kaydı incelemek ve 0,1 veya –1 olarak değerlendirmek için basit bir compare(byte[] rec1, byte[] rec2) metoduna ihtiyaç duymaktadır. 0 değeri arama veya sıralama istemine göre kayıtların eşdeğer veya eşit olduğunu belirtmektedir. 1 değeri arama veya sıralama istemine göre birinci kaydın ikinci kaydı izlediğini belirtmektedir. Son olarak, -1 dönüş değeri birinci kaydın ikinci kayıttan daha önce geldiğini belirtmektedir (Çizelge 4.4).

Çizelge 4.4: Çeşitli RecordComparator sahalarına atanmış değerler

Statik Saha	Atanan Değer
RecordComparator.EQUIVALENT	0
RecordComparator.FOLLOWS	1
RecordComparator.PRECEDES	-1

Örneğin, bir kayıt deposuna basit stringlerin eklenmiş olduğunda karşılaştırma gerçekleştirimi aşağıdaki kod parçasındaki gibi olacaktır.

```
import javax.microedition.rms.*;
public class TestComparator implements RecordComparator {
  public int compare(byte[] rec1, byte[] rec2) {
    String r1 = new String(rec1);
    String r2 = new String(rec2);
    if (r1.compareTo(r2) > 0)
      return (RecordComparator.FOLLOWS);
    else if (r1.compareTo(r2) < 0)
      return (RecordComparator.PRECEDES);
    else return (RecordComparator.EQUIVALENT);
  }
}
```

Bu durumda, aşağıdaki satırların bir MIDP uygulamasında çalıştırılması sonucunda “a test” ile “is” String’i karşılaştırıldığında şöyle bir sonuç okunmaktadır : “Comparator found —> -1”. Gerçekten “a test” string’i “is” stirng’inden önce gelmektedir.

```
anRMS = RecordStore.openRecordStore("TestRMS" , true);
  String test = "This is a test";
  byte[] b = test.getBytes();
  anRMS.addRecord(b, 8, 6);
```

```
  anRMS.addRecord(b, 5, 2);
RecordComparator rc = new TestComparator();
byte[] r1 = anRMS.getRecord(1);
byte[] r2 = anRMS.getRecord(2);
System.out.println("Comparator found --> " + rc.compare(r1,r2));
```

1 – Kayıt deposunu aç/yarat ve string kayıtları depola

2 – Kayıt karşılaştırıcısının bir örneğini yarat

3 – Kayıt deposundan kayıtları getir.

4 – Getirilen iki kaydı karşılaştırmak için karşılaştırıcıyı kullan

Karşılaştırıcı kayıt deposundaki kayıtları karşılaştırırken tek başına kullanışlı olurken, bir kayıt filtreleyici ve kayıt sıralayıcı ile birlikte kullanıldığında daha güçlü bir araç olabilmektedir.

4.3.7. Kayıtları filtreleme

RecordComparator'a benzer bir şekilde RMS API bir kayıt filtreleme arayüzü sunmaktadır, herhangi bir nesnenin kayıt deposundaki kayıtlar için bir süzgeç gibi hizmet etmesine izin vermektedir. RecordFilter arayüzü gerçekleştiren nesnesinin basit bir metodu gerçekleştirmesine ihtiyaç duymaktadır. Bu metot, matches(byte[] candidate), gönderilen byte dizisini içindeki verinin filtreleme kriterine uyup uymadığını belirlemek için kontrol etmektedir. Bu metot gönderilen kaydın filtreleme kriterine uyup uymadığını gösteren mantıksal bir değer döndürür. Bir kayıt filtresi, mesela, kayıt deposunda "A" (küçük veya büyük) harfi ile başlayan bir kayıt olup olmadığını belirleyecek şekilde ayarlanabilmektedir. Bir filtre için yazılacak kod aşağıdakine benzeyecektir:

```
import javax.microedition.rms.*;
public class TestFilter implements RecordFilter {
  public boolean matches(byte[] rec) {
    String r = new String(rec);
    return ((r.charAt(0) == 'a') || (r.charAt(0) == 'A'));
  }
}
```

Artık filtre bir kaydın verilen kritere uyup uymadığının belirlenmesi için aşağıdaki gibi kullanılabilir. Bu filtre kullanılarak sistem çıktısı “The first record starts with ‘A’.” olarak üretilir.

```
anRMS = RecordStore.openRecordStore("TestRMS" , true);
String test = "A test";
byte[] b = test.getBytes();
anRMS.addRecord(b, 0, b.length);
RecordFilter rf = new TestFilter();
if (rf.matches(anRMS.getRecord(1)))
  System.out.println("The first record starts with 'A'");
  else
  System.out.println("The first record does not start with 'A'");
```

1 – Kayıt deposunu aç/yarat

2 – Bir string içeren kayıt sakla/depola

3 – Yukarıdaki kayıt filtresinin bir örneğini yarat

4 – Eşleşen kayıtların hepsini almak için bu filtreyi kullan

4.3.8. Kayıtlar üzerinde "Enumeration" işlemi

Kayıt filtreleme ve karşılaştırma işlemleri kayıt deposundaki kayıtlar üzerinde sıralama yapılırken bize yardımcı olan RMS'in iki özelliğidir. Enumeration'lar vektörler gibi veri yapısı nesnelerinin her elemanı üzerinde bir dizi işlem yapabilmemizi sağlamaktadır. Bir "enumeration" veri elemanı dizilerini hazırlar ve bize bu diziden elemanları çekecek olan erişim metodlarını sağlar. Aynı şekilde, MIDP'nin RMS'i de kayıt deposundaki kayıtları "enumerate" etmek için bize RecordEnumeration arayüzünü sağlamıştır. Dahası, bir kayıt karşılaştırıcısı ve/veya kayıt filtreleyicisi kayıt enumerator'ı ile birlikte kullanılabilmektedir. Kayıt "enumeration" işlemi herhangi bir kayıt deposu örneği üzerinde enumerateRecords(RecordFilter filter, RecordComparator comparator, boolean keepUpdated) metodu çağırılarak gerçekleştirilebilir. Filtreleme işlemi eğer parametre null değil ise gerçekleştirilir. Filtre veya karşılaştırıcı "null" değerler alabilir.

1 Eğer kayıt filtreleyicisi null ise, enumeration işleminde tüm kayıtlar döndürülür.

2 Eğer kayıt karşılaştırıcısı null değil ise, enumeration'daki kayıtlar karşılaştırıcıya göre sıralanır.

3 Eğer karşılaştırıcı null ise, enumeration'daki kayıtlar sıralanmaz ve belirli olmayan bir şekilde dolaşılır.

Bir kayıt enumeration'ı kayıtları almak için bir dizi erişim metodu ile birlikte gelir. nextRecord() metodu çağırıldığında dizideki bir sonraki eleman veya kayıt döner. Aynı şekilde, previousRecord() metodu bize dizideki bir önceki byte dizisini geri döndürür. MIDlet'ler gerçek kayıt veya byte dizileriyle ilgilenmekten çok recordId'ler ile ilgilendiklerinden nextRecordId() ve previousRecordId() metotları da sağlanmıştır. hasNextElement() ve hasPreviousElement() metotları ise dizinin ileri veya geri yönlerinde bir sonraki elemanın olup olmadığını belirten bir mantıksal değer döndürür. numRecords() metodu ise dizideki toplam kayıt sayısını geri döndürür [17].

Yukarıda anlatılanları örnekleyen kod aşağıda gösterilmiştir. Örnek bir kayıt deposunda 'A' harfi ile başlayan isimleri alfabetik sırada listelemektedir.

```
anRMS = RecordStore.openRecordStore("TestRMS", true);
byte[] george = "George".getBytes();
byte[] bob = "Bob".getBytes();
byte[] andy = "Andy".getBytes();
byte[] harry = "Harry".getBytes();
byte[] adam = "Adam".getBytes();
byte[] amos = "Amos".getBytes();
byte[] fred = "Fred".getBytes();
anRMS.addRecord(george, 0, george.length);
anRMS.addRecord(bob, 0, bob.length);
anRMS.addRecord(andy, 0, andy.length);
anRMS.addRecord(harry, 0, harry.length);
anRMS.addRecord(adam, 0, adam.length);
anRMS.addRecord(amos, 0, amos.length);
anRMS.addRecord(fred, 0, fred.length);
RecordComparator rc = new TestComparator();
RecordFilter rf = new TestFilter();
RecordEnumaration rEnum = anRMS.enumarateRecords(rf, rc, false);
while (rEnum.hasNextElement()) {
        byte[] nextRec = rEnum.nextRecord();
        String nextName = new String(nextRec);
        System.out.println(nextName);
}
rEnum.destroy();
```

1 – Kayıt deposunu aç/yarat

2 – İsimler içeren birçok kayıt yarat

3 – Yukarıdan karşılaştırcıyı kullan

4 – Yukarıdan filtreyi kullan

5 – Seçilen kayıtları bulmak için karşılaştırıcı ve filtreyi kullanarak bir sıralama oluştur

6 – Kaynakları serbest bırakmak için sıralamayı kapat

Bu kodu MIDlet uygulamasının bir parçası olarak çalıştırdıktan sonra, sistem çıktısı olarak aşağıdaki sonuçlar üretilir:

Adam

Amos

Andy

Bu son örnekle, RecordComparator, RecordFilter ve RecordEnumeration RMS arayüzleri bir arada kullanılarak bir MIDP uygulamasında verileri veri deposunda saklamak, filtrelemek, sıralamak ve ayıklamak için kullanılırken güçlü bir mekanizma sağlarlar.

5. AĞ BAĞLANTISI

J2ME'nin en kritik özelliği ağ bağlantısı yapabilmesidir. J2ME cihazları ağa bağlı değilken de kullanışlı olmalarına rağmen bu cihazlar ağ bağlantısı yaptığında ağın güçlü kaynaklarından yararlanır.

Kullanıcı ağa bağlandığında yazıcı, faks makinesi, elektronik posta, ağ dosya sistemi, veritabanları gibi ağ servislerine ulaşabilir.

5.1. "Generic Connection Framework" Nedir?

"Generic Connection Framework" J2ME mimarisi ile tüm ağ bağlantısı iletişimi için kuruluş sağlar. Konfigürasyon katmanı ile "Generic Connection Framework" arayüzü temel arayüzler ile tanımlanmıştır. "Generic Connection Framework" protokol gerçekleştirimi sağlamaz.

Satıcılar, gerekli "Generic Connection Framework" arayüzlerini gerçekleştiren profili sağlamalıdırlar.

"Generic Connection Framework" javax.microedition.io paketinin içinde bulunur ve

1 Bir sınıf (Connector)

2 Bir exception (ConnectionNotFoundException)

3 Sekiz arayüz (Çizelge 5.1)

içerir.

Çizelge 5.1: Arayüz tipleri ve amaçları

GCF	Arayüzün Amacı
Connection	GCF'nin temel bağlantı tipidir. Diğer tüm bağlantı tipleri Connection'ı genişletir
ContentConnection	Bağlantının içeriği için bağlantıyı yönetir. İçeriğin uzunluğunu, tipini inceleyecek temel metodları sağlar.
Datagram	Datagram bağlantısında geçen veri için bir konteyner sağlar.
DatagramConnection	Datagram bağlantısını yönetir.
InputConnection	Input-stream tabanlı bağlantıları yönetir.
OutputConnection	Output-stream tabanlı bağlantıları yönetir.
StreamConnection	Stream yeteneklerini yönetir. InputConnection ve OutputConnection metodlarını birleştirir.
StreamConnectionNotifier	Belirli bir portu dinler ve portta bir aktivite yakaladığında StreamConnection oluşturur.

"Generic Connection Framework" J2ME mimarisinin konfigürasyon katmanında yer alır. Framework bu katmanda gerçekleştirilerek, aynı "Connector" ve arayüzler tüm profillerde kullanılabilir.

"Generic Connection Framework" CDC ve CLDC tarafından desteklenir. Bağlantı arayüzleri, konfigürasyonların içiçe ilişkisi sayesinde J2ME mimarisinde hep bulunur. Bu konfigürasyonlar arasında uyumluluğu arttırır [18].

5.1.1. Connector Sınıfı

Connector sınıfı, connector'ın statik metodlarından biri kullanılarak bağlantı protokolünün örneğini yaratmak için kullanılır. Amacı bağlantı

protokolünün örneğini yaratmaktır. Connector sınıfının tüm metodları statiktir ve bu amaç için kullanılır.

Connector sınıfı, connection örneği döndüren open() metodunun üç varyasyonunu tanımlar. Connector input ve output streamleri döndüren metodları içerir.

open() metodu connection tipinin örneğini döndürür. Dönen örnek Connection'ın alt sınıfı olsa da daha karmaşıktır. "open" metodu aşağıdaki imzalara sahiptir.

```
1  open(String name)
2  open(String name, int mode)
3  open(String name, int mode, boolean timeouts)
```

İsim URI (Uniform Resource Name/Tekdüze Kaynak Adı) olmalı ve plan, adres ve parametre listesinden oluşmalıdır. İsim parametresinin genel yapısı aşağıdaki gibidir.

<scheme>:<address>;<parameters>

"scheme" bağlantının nasıl yapıdığını(soket, http, dosya, datagram) belirtir. "address" bağlantının ne için yapıldığını, "parameter" ise bağlantı hızı gibi protokol tarafından gereken diğer bilgileri tanımlar. Bazı URI örnekleri aşaığıda gösterilmiştir. Dikkat edilirse "parameter" her durumda gerekli değildir.

```
1  http://www.ctimn.com:8080
2  socket://localhost:8080
3  file:c:/myfile.txt (Sadece Windows)
4  file:/myfile.txt (Unix)
5  datagram://127.0.0.1:8099
6  comm:0;baudrate=9600
```

"mode" parametresi erişim tiplerini (sadece-yazma, sadece-okuma, sadece-yazma) belirtir. Bu tipler Connector sabitlerinde READ, READ_WRITE ve WRITE olarak tanımlanmıştır.

“timeouts” parametresi zaman aşımı olduğunda bağlantının InterruptedIOException fırlatıp fırlatmayacağını belirleyen parametredir. Eğer bu parametre işaretliyse uygulama zaman aşımı olduğunda aykırı durumu gerçekleştirmek zorundadır.

Connector sınıfı “Generic Connection Framework” kullanarak değişik tipte bağlantılar yaratmayı sağlayan sınıftır. Özel protokol gerçekleştirimleri doğrudan doğruya yaratılmak için dizayn edilmiştir.

Connector arayüzünce tanımlanan diğer metodlar:

1 openInputStream()

2 openOutputStream()

3 openDataInputStream()

4 openDataOutputStream()

Bu metodlar Connection yaratıldığı anda değişik tipte girdi ve çıktı streamleri yaratmak için kullanılırlar. Çoğu zaman uygulamalar kendisine yazılan veya kendisinden okunan stream yerine Connection örneğiyle ilgilidirler. Bu dört metottan birini kullanarak, uygulama bağlantı örneğiye ilgilenmeye gerek duymadan doğrudan stream elde edebilirler. Aşağıdaki örnek stream elde etmede kullanılabilecek iki yol arasındaki farkı göstermektedir.

```
try {
OutputConnection connection =
(OutputConnection)Connector.open("socket://127.0.0.1:8888");
OutputStream os = connection.openOutputStream();
os.close();
connection.close();
} catch (IOException x) {
//Exception gerçekleşir
}
```

İlk yol OutputStream yaratmamızı sağlar ve openOutputStream() metodunu çağırmamız için Connection ile haberleşmemizi gerektirir. Dönüş değerini OutputConnection tipine dönüştürmemiz için zorlar.

```
try {
OutputStream os =
Connector.openOutputStream("socket://127.0.0.1:8888");
os.close();
} catch (IOException x) {
//Exception gerçekleşir
}
```

İkinci yol; OutputStream sağlayarak Connection örneğiyle etkileşim kuran kod satırları çıkarılır. Dönüş değeri için bir zorlama yoktur. openOutputStream() metodu kullanılırken bir sorunla karşılaşılır. Bağlantıyı kim kapatacak? Bu durumda stream döndüğünde openOutputStream() metoduyla bağlantı kapatılır. Bağlantı output stream sağlanacak kadar uzun kalır. Stream sağlandığında bağlantı kapatılır. Bu sadece stream tabanlı bağlantılarda çalışır.

5.2. HTTP- Tabanlı Bağlantı

HTTP başlık bilgisi gibi HTTP'ye özgü bilgiyle ilgilenmezken HTTP Get bağlantısını kurmak için GCF ContentConnection arayüzü kullanılır.

5.2.1. Bağlantı oluşturma

ContentConnection arayüzü zengin veri değişimleri sağlar. Bu tip bağlantılar StreamConnection arayüzünü genişletir ve bağlantının tipinin, içeriğin uzunluğunun ve kullanılan şifreleme yönteminin saptanması için uygulamaya metod tanımlar.

1 String getEncoding()

2 long getLength()

3 String getType()

Çoğu durumda, ContentConnection doğrudan kullanılmaz; HttpConnection arayüzü gibi protokole özgü bağlantı tipleri için temel arayüz olarak hizmet verir. Aşağıdaki örnek ContentConnection'ın HTTP ile nasıl kullanıdığını gösterir.

```
ContentConnection connection = (ContentConnection) Connector.open(
"http://www.catapult-technologies.com/ctimain.htm", Connector.READ);
```

Bu örnekte belirtilen URL'deki (www.catapultechnologies.com) HTML sayfasını okuyoruz. Bağlantı önceden belirlenmiş olan HTTP portundan(port 80) oluşturulur. Bu örnekte sayfayı okumak için açılan bağlantı sadece-okuma tipindedir.

Bağlantı bir kere sağlandığında, input stream'i oluşturuyoruz:

```
DataInputStream is = connection.openDataInputStream();
```

Bu belirli örnekte Connector'ın openDataInputStream metodunu kullanarak input stream'i kolaylıkla oluşturduk. Daha önceden açıklanan içeriğin uzunluğunu, içeriğin tipini ve şifreleme yöntemini sağlayan metodlar ContentConnection örneğinin içinde bulunur [19].

ContentConnection metodları bağlantı hakkında bilgi edinmek için aşağıdaki gibi kullanılabilir:

```
System.out.println("encoding: "+ connection.getEncoding());
System.out.println("length: "+ connection.getLength());
System.out.println("type: "+ connection.getType());
```

5.2.2. Bağlantıyı kullanma

Bağlantı bir kere kurulduğunda verinin alınması için kullanılabilir. Aşağıdaki örnek ağ bağlantısından content connection kullanılarak HTML sayfası okumayı örnekliyor. Bu örnekte alınan verinin büyüklüğü önemli olduğundan mobil telefon emülatörünün ekranında sadece verinin gerekli kısma yer alacak şekilde veri ayrıştırılır. Alınan verinin tüm içeriği consolda gösterilecektir böylece ilerleyişin nasıl olduğu görülebilecektir. Yaratılacak sınıfın adı "MsgClient" tır.

```
package com.ctimn;
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class MsgClient extends MIDlet implements CommandListener {
  private Form outputForm;
  private Display display;
  private List menu;
  private Command okCmd = new Command("OK", Command.OK, 1);
  private Command exitCmd = new Command("Exit", Command.EXIT, 1);
  private static final String[] choices = {
     "1 HTTP Example"
  };
  protected void startApp() throws MIDletStateChangeException {
    display = Display.getDisplay(this);
    outputForm = new Form("Server Messages");
    menu = new List("Select:", List.IMPLICIT, choices, null);
    menu.addCommand(okCmd);
    outputForm.addCommand(okCmd);
    outputForm.addCommand(exitCmd);
    menu.addCommand(exitCmd);
    outputForm.setCommandListener(this);
    menu.setCommandListener(this);
```

```
  display.setCurrent(menu);
}
protected void pauseApp() {
}
protected void destroyApp(boolean unconditional)
    throws MIDletStateChangeException {
}
public void commandAction(Command cmd, Displayable displayable) {
  if (cmd == exitCmd){
    handleExit();
  } else if ((displayable == menu) && (cmd == okCmd)) {
    handleOK(((List)displayable).getSelectedIndex());
  } else {
    display.setCurrent(menu);
  }
}
private void handleExit(){
  try {
    notifyDestroyed();
    destroyApp(true);
  } catch (MIDletStateChangeException x) {
    x.printStackTrace();
  }
}
private void handleOK(int idx){
  display.setCurrent(outputForm);
  getHttpMessage();
}
private void getHttpMessage(){
  int c = 0;
  String dataIn = null;
  StringItem item = new StringItem("Reading from URL", "");
  outputForm.append(item);
  try {
```

```
    ContentConnection connection = (ContentConnection)
       Connector.open(
       "http://www.catapult-technologies.com/ctimain.htm",
       Connector.READ);
    DataInputStream is = connection.openDataInputStream();
    try {
     System.out.println("encoding: "+ connection.getEncoding());
     System.out.println("length: "+ connection.getLength());
     System.out.println("type: "+ connection.getType());
     StringBuffer sb = new StringBuffer("");
     for (int ccnt=0; ccnt < connection.getLength(); ccnt++){
       c = is.read();
       sb.append((char)c);
      }
      dataIn = sb.toString();
      item = new StringItem("Title: ", getTitle(dataIn));
      outputForm.append(item);
    } finally {
      is.close();
    }
  } catch (IOException x) {
    System.out.println("Problems sending or receiving data.");
    x.printStackTrace();
  }
 }
 private String getTitle(String data){
  String titleTag = "<TITLE>";
  int idx1 = data.indexOf(titleTag);
  int idx2 = data.indexOf("</TITLE>");
  return data.substring(idx1 + titleTag.length(), idx2);
 }
}
```

1. Kullanıcı arayüzü kurulur.
2. Bağlantı açılır.
3. Input stream açılır.
4. Bağlantı bilgileri görüntülenir.
5. Girdi okunur.
6. Baytlar karakter çevrilir.
7. Başlık genişletilir.

"try" kaynak açıldıktan sonra yer almalıdır. Açma komutunun try...finally bloğunda yer alması açma işlemi sırasında problem oluşursa aykırı durum fırlatılması içindir. Bağlantı tanımsız bir konumdayken akış kontrolu close() komutu ile sağlanır.

5.3. Soket Tabanlı Bağlantılar

Soketler iki sistem arasındaki iletişimi gerçekleştirirler ve ağ bağlantısının akışını sağlarlar. Soket gelen veriyi okumak için InputStream veya veri yazmak için OutputStream kullanır. "Generic Connection Framework" soketlerle çalışamak üzere iki arayüz tanımlamıştır; StreamConnectionNotifier ve StreamConnection.

StreamConnectionNotifier bağlantı kurmak için bekleyen kullanıcılara port sağlamak için sunucu tarafından kullanılır. StreamConnection soket bağlantısı kurmak için kullanılır.

StreamConnection arayüzü InputConnection ve OutputConnection tek bir bağlantıda okuma ve yazma yeteneklerini kullanabilmek için genişletir. Genelde InputConnection, DataInputConnection, OutputConnection ve DataOutputConnection arayüzleri "Generic Connection Framework" tarafından daha karmaşık bağlantı tipleri oluşturmak için kombinasyonlar şeklinde kullanılır.

Çoğu bağlantı protokolü okuma ve yazma yeteneklerini desteklediğinden beri bunlar ayrı ayrı pek yararlı değildirler.

İstemci, dinleyiciden bir soket bağlantısı isteğinde bulunmadan önce, dinleyici hedef portu dinliyor olmalıdır. StreamConnectionNotifier kullanarak soket dinleyici uygulamaya bağlamak için "open" komutu kullanılır:

StreamConnectionNotifier connection = (StreamConnectionNotifier)

Connector.open("serversocket://:4444", Connector.READ_WRITE);

Connector, "serversocket" planını aramak için StreamConnectionNotifier açacağını bilir. Port numarası ":" işaretinden ismin adres kısmında ":" işaretinden sonra belirtilir. Port seçimi rastgeledir böylece istemci ve soket dinleyici aynı port numarasını kullanabilir. Eğer port numarası uygun değilse veya başka bir servis porta bağlandıysa bağlantı reddedilir. Bir kere StreamConnectionNotifier elde edildikten sonra soket dinleyici istemcinin acceptAndOpen() metodunu kullanarak denemede bulunması için bekler [20].

StreamConnection sc = connection.acceptAndOpen();

Client soket dinleyiciye bağlanmak için girişimde bulunduğunda, acceptAndOpen() metodu bağlantını kurulduğunu doğrular ve StreamConnection soket açar. Bir kere StreamConnection kurulunca soket dinleyici veri okumaya ve veri yazmaya hazırdır. Bağlantının dinleyici tarafından kabul edilmesi için istemcinin denediği bağlantı soket bağlantısı kurmaya çalışıyor olmalıdır. Dinleyicinin gerçekleştiremeyeceği veya anlamayacağı bağlantı tipleri reddedilir.

İstemciler soket dinleyicisine doğrudan StreamConnection açarak bağlanırlar.

StreamConnection connection = (StreamConnection)

Connector.open("socket://127.0.0.1:4444", Connector.READ_WRITE);

İstemci tarafında soket açmak için, "soket" planı uygulanır ve ismin adres kısmında sunucu ve port numarası belirtilmelidir. İstemci ve soket dinleyici için açılan bağlantılarda port numarasının aynı olmasına özellikle dikkat edilmelidir. Eğer port numaraları farklıysa bağlantı kurulmayacaktır. Üstelik eğer sunucu adresteki soket dinleyici aynı portu dinlemiyor veya bu sokete bağlanamayacak başka tip bir servis bu porta bağlandıysa bir aykırı durumla bağlantı reddedilecektir. Eğer bağlantı başarıyla kurulduysa soket soketin girdi ve çıktı streamlerinden veri okumaya ve yazmaya hazırdır.

Genellikle streame ilk veri yazan istemci olur ama bu soketlerin bir gerekliliği değildir. Bir kere bağlantı başarıyla kurulduktan sonra isteyen taraf dialoğa başlayabilir.

Soketler sistemler arasında yararlı iletişim sağlarlar; bununla birlikte soketler bağlantıyı ve TCP/IP gibi düşük seviyeli veri taşıma mekanizmalarını sağlarlar.

5.3.1. Sokete yazma

Başarılı bir bağlantı kurulduktan sonra openOutputStream() veya openDataOutputStream() metodları kullanılarak SteramConnection'dan output stream yaratılır.

```
OutputStream os = connection.openOutputStream();
DataOutputStream os = connection.openDataOutputStream();
```

Output Steram oluşturulduktan sonra uygulama OutputStream'in değişik metodlarını kullanarak veri yazmaya başlayabilir. OutputStream, stream'e yazarken daha zengin veri desteği sağlamak için başka stream sınıflarının içine yerleştirilebilir. Örneğin, OutputStream karakter tabanlı içerik yerine bayt tabanlı içerik için OutputStreamWriter'a geçebilir. OutputStreamWriter filtre görevi

yapar. Veriyi karakter gösterimden bayt gösterime OutputStreamWriter metodlarıyla çevirir ve bunu uygun OutputStream metoduna geçirir.

5.3.1.1.OutputStream

OutputSteram stream'e bayt veriyi yazmak, stream'i boşaltmak ve kapatmak için gerekli temel metodları sağlar. Diğer tüm output stream sınıfları OutputSteram'i genişletirler.

5.3.1.2.DataOutput

DataOutput, temel tipleri output stream'e yazılabilecek bir bayt dizisine çevirmek için metodları tanımlayan bir arayüzdür. Ayrıca bu arayüz Java String'lerini UTF-8 formatında bayt dizisi gibi output stream'e yazacak özelliğe sahiptir.

5.3.1.3.DataOutputStream

DataoutputStream, OutputStream'i genişletir. Makineden bağımsız davranıştaki bayt, karakter ve UTF şeklinde şifrelenmiş verinin iletişimini sağlamak için DataOutput arayüzünü gerçekleştirir.

5.3.1.4.ByteArrayOutputStream

ByteArrayOutputStream OutputStream'i genişletir ve bayt stream'lerin yazılabilmesi için dinamik tampon özellikleri sağlar. Veri ByteArrayOutputStream'e yazıldığında tampon otomatik olarak büyür. Bu sınıf verinin bayt dizisi şeklinde alınabilmesi için toByteArray() ve string olarak alınabilmesi için toString() metodlarına sahiptir.

5.3.1.5.Writer

Writer karakter dizilerinin bayt şeklinde yazılabilmesini sağlayan soyut bir sınıftır. Java bayt stream'leri ve karakter tabanlı stream'ler arasınada köprü vazifesi gören sınıfları belirtmek için java.io paketinin içinde "Writer" ismini kullanır.

Writer kullanmanın temel yararı verinin bayt gösterimi ve karakter gösterimi arasında çevirimin otomatik olarak yapılmasıdır. Tüm writer sınıfları Writer'ı genişletir.

Bu sınıf soyut olduğundan uygulamalarda doğrudan kullanılamaz. Writer özelliğine gerek duyan uygulamalar OutputStreamWriter kullanmalıdırlar.

5.3.1.6.OutputStreamWriter

OutputStreamWriter Write'ı genişletir ve uygulamalar içi karakterleri output stream'e yazmayı sağlar.

5.3.1.7.PrintStream

PrintStream, OutputStream'i genişletmeye uygun bir sınıftır ve stream verisinin yazdırılmasını sağlar. PrintStream değişik veri tipleri için println() metoduna sahiptir. println() yazdığı verinin sonuna "\n" karakterini ekler.

5.3.2. Soketten okuma

Soketten okuma API'leri çıktı API'lerine benzer fakat okuma işlemi yaparlar. Input stream bir kere oluşturulduğunda, stream'den veri alabilecek pekçok sınıf vardır.

5.3.2.1.InputStream

InputStream stream'den bayt veriyi okumak ve stream'i kapatmak için temel metodları içerir. Diğer tüm input stream sınıfları InputStream'i genişletirler.

5.3.2.2.DataInput

DataInput bayt stream'ini okuyan ve bunu Java'nın temel veri tiplerine dönüştüren metodlar içeren arayüzdür.

5.3.2.3.DataInputStream

DataInputStream, InputStream'i genişletir ve okumak ve input stream'deki bayt serilerini Java'nın temel veri tiplerine dönüştürmek için DataInput arayüzünü gerçekleştirir.

5.3.2.4.ByteArrayInputStream

ByteArrayInputStream InputStream'i genişletir ve bayt input stream'den okurken tamponlama özelliklerini sağlar.

5.3.2.5.Reader

Reader API içindeki diğer reader'lar için soyut temel bir sınıftır. Java bayt stream'leri ve karakter tabanlı stream'ler arasınada köprü vazifesi gören sınıfları belirtmek için java.io paketinin içinde "Reader" ismini kullanır.

Writer kullanmanın temel yararı okuma işlemi sırasında verinin bayt gösterimi ve karakter gösterimi arasında çevirimin otomatik olarak yapılmasıdır. Tüm reader sınıfları Reader'ı genişletir.

5.3.2.6.InputStreamReader

InputStreamReader, Reader'ı genişletir ve bayt stream'den karakter verisinin okunmasını sağlar.

5.3.3. Soketler ne zaman kullanılmalı?

Soketler iki sistem arasında iletişim kurmanın ve veri değişmenin ilkel fakat basit bir metodudur. Soketlerin düşük yüküne bağlı olarak veri değişmenin en hızlı yollarından biridir. Soketler bağlantı sağlarlar, ama değişimde bulunulacak verinin formatı uygulamayı gerçekleştiren kişiye kalmıştır. Soketlerle yapacaklarımızın kıstları çok azdır.

Soketler hızın önemli olduğu uygulamalarda oldukça yararlıdır. Soket kullanarak tescilli bir veri taşıma mekanizmasını gerçekleştirmiş oluyoruz. Eğer sistem açık iletişim standartlarının kabul ediyorsa ve hem istecinin hem de sunucunun kontrolü altında değilse, soket uygulamada iletişim yeteneklerini gerçekleştirmek için iyi bir yol değildir. Bu gibi durumlarda HTTP daha uygun olur.

Soketler, tamamen taşıma mekanizması sağladığından beri, diğer veri format ve protokolleri soketlerle birleştirilerek kullanılır. Örneğin soketler XML şema ile ile kullanılabilir. Bu, tescilli olmayan veya genel tanımlı XML şema kullanılması soketlerin avantajına olur.

5.4. Datagram Tabanlı Bağlantılar

Datagramlar ağ üzerinden veri paketlerinin gönderilmesi için dizayn edilmişlerdir. Datagramlar soketlerden farklı davranırlar; iki sistem arasında kuvvetli bir bağlantı yoktur. Soketlerde eğer istemci soketi desteklemeyen bir sisteme bağlanmaya çalışıyorsa veya soket bağlantıları için hazır değilse, aykırı durum fırlatılır.

Datagramda diğer uçtaki dinleyici datagram almaya uygun olmasa bile datagram gönderilir. Datagram kullanırken bağlantının başarılı olduğu varsayılır. Üstelik soketlerde farklı olarak, veri gönderimi için datagram kullanmak güvenilir değildir. Paketin kaybolduğu protokol tarafından anlaşılmaz ve birden fazla paket gönderildiğinde paketlerin gönderildiği sırada ulaşacağının garantisi yoktur. Datagramlar, gönderilen ayrı ayrı gönderilen paketlerin birleştirileceğine dair destek yoktur. Bu nedenlerden datagram güvenilir olmayan veri taşıma mekanizması olarak adlandırırlır. Güvenilir olmayan, verinin gönderildiği sırada ulaşacağını veya verinin ulaşacağını garanti etmeyen anlamndadır.

Peki neden datagramları kullanıyoruz? Asıl neden hızdır. Ses paketlerinde eksik veri farkedilmeyebilir. Hız, veri bütünlüğünden daha önemlidir.

Mevcut pekçok datagram protokolleri vardır. En sık kullanılan User Datagram Protocol (UDP)'dür. "Generic Connection Framework" referans gerçekleştirimi tarafından bu protokolün gerçekleştirimi sağlanır. Datagram ve DatagramConnection arayüzleri sayesinde "Generic Connection Framework" değişik datagram protokollerini destekler [21].

5.4.1. Datagramlar ne zaman kullanılmalı?

İlk bakışta datagramların kullanılamamaları için pekçok neden varmış gibi gözükür. Veri iletimimin, akış kontrolünün ve hata yakalamanın güvensizliği gibi. Çoğu uygulama için datagramların yüksek hızını veri bütünlüğüne tercih eder. Gerçek zamanlı ses ve video uygulamaları verinin her baytının ulaşıp ulaşmadığından çok ulaşma hızıyla ilgilidirler. Eğer eksik bir veri olursa seste veya görüntüde bir duraklamaya neden olur. Bu duraksama uygulamalarda istenen bir özellik olmasa da vernin ulaşmasının ve sıraya sokulmasının beklenmesine tercih edilir. Ses ve video uygulamalarında hızın azalması kabul edilemez.

UDP paketlerin ulaşacağını veya ulaşma sırasını garanti etmediğinden, başlık veya metadata TCP gibi güvenli protokollere göre daha basittir.

Datagramlar ulaşma hızı çok önemli olduğunda kullanışlıdırlar. J2ME ortamında, kolaylıklarına bağlı olarak TCP'ye alternatif olabilirler. Örneğin; iki cihaz arasında Ir portundan veri gönderiminde kullanılabilir.

Datagramların diğer bir özelliği programcının paket büyüklüğünü kontrol edebilmesidir. İstenilen büyüklükte paket gönderilebilir.

5.4.2. Datagramlar J2ME'de nasıl çalışır?

Değişik datagram tiplerinin kullanılabilmesi için "Generic Connection Framework" içinde datagramlar genelleştirilmiştir. Sonuç olarak, J2ME'deki datagram API'leri J2SE'dekinden farklıdır.

"Generic Connection Framework" içinde datagramları içeren iki sınıf vardır: DatagramConnection ve Datagram. DatagramConnection sınıfı uygulamaları porta bağlamak ve Datagram sınıfı da port bağlantısı üzerinden veri transfer etmek için kullanılır. Datagramların stream'ler gibi davranmadığına dikkat etmek gerekir. Stream'le farkı, stream'e yazılan her bayt hemen stream'in bir parçası haline gelirler ve tamponlama işlemi olmadığı varsayılarak her türlü stream gönderilir. Datagramlarda ise datagram DatagramConnection da yer alana kadar tüm veri datagram tamponda tutulur. Datagram bağlantıda bir kere yer aldıktan sonra, bağlantı veriyi belirtilen hedefe göndermeye başlar.

Datagramları J2ME API kullanarak göndermek için, uygulama üç şeyi sağlamalıdır: datagramın gönderileceği adres, alıcı sistemin dinlediği port ve veri. İstemcinin veri göndermek için uygulama tarafında kullanacağı port dinamik olarak tahsis edilmiştir.

DatagramConnection örneği datagram istemci veya sunucu olmaya bağlı olarak değişik isim parametresi alır. İstemci olarak bağlantı açmak hedef sunucu belirtilmelidir. Aşağıdaki örnek istemci tipinde DataConnection açar [22].

datagram://127.0.0.1:5555

Alıcı tarafta bağlantı açmak için sadece port belirtilir.

datagram://:5555

İstemci bağlantısı oluşturulurken, uygulama sunucunun dinlediği port belirtilmelidir. Sistemde istemcinin kullanacağı port geliştiriciden saklanır ve dinamik olarak atanır.

Alıcı tarafta bağlantı açılırken, gelen uygulama kendini porta bağlar. Eğer bu portlar tanımlı değilse, istemci tarafından gönderilen veri kaybolacaktır. Datagramlar paketin ulaşacağını garanti etmediğinden, istemci veriyi bir kere yollar ve ters giden birşey olduğunda bilgilendirilmez

Bir kere DatagramConnection kurulunca, tek bir bağlantı üzerinden birden fazla datagram gönderilebilir. Datagram cevabı aynı datagram kullanılarak gönderilir.

6. MOBİL TELEFON İLE SİPARİŞ ALMA UYGULAMASI

Tez çalışmasının bu bölümünde, günümüzde birçok firmada sipariş alma işleminde kullanılan el terminali cihazları yerine mobil telefonların kullanılmasını sağlayacak bir programın gerçekleştiriminden, özelliklerinden avantajlarından ve dezavantajlarından bahsedilecektir.

Sipariş alma işlemi, firmaların bayii (müşteri) ihtiyaçlarını önceden alıp gereken ürünleri zamanında bayiye eksiksiz olarak teslim etmesi için gereken bir işlemdir. Birçok firma sipariş alma işlemini yaparken bazı firmalar ise bu işlemi gereksiz bulup yapmamaktadır. Sipariş alma yerine ürünleri nakliye araçlarına yükleyerek bayilerine götürmekte ve istenen ürünleri vermektedirler. Bu durumun bazı sakıncaları vardır. Öncelikle müşteri ihtiyaçları önceden bilinmediği için araçlara hangi üründen ne kadar yükleneceği kestirilemez. Bu nedenle ya araçlara gereğinden fazla ürün yüklenir ve gereksiz yere enerji harcanır ya da araçtaki ürünler tüm müşterilere yetmeyebilir ve stokta ürün bulunmasına rağmen müşterilere ürün satılamaz. Bu durum müşteri memnuniyetsizliğine ve müşterilerin tedarikçi firmalarını değiştirmelerine neden olabilir.

Sipariş alma işlemi iki farklı şekilde yapılmaktadır. İlk yöntem müşteri isteklerini bir kâğıda yazarak sipariş almak, diğeri ise “El Terminali” denilen ve bu iş için üretilmiş cihazlarla siparişi almak. Siparişi kağıda yazarak almak donanım ihtiyacı yönünden düşünüldüğünde en uygunu gibi görünmesine rağmen pek tercih edilmez. Çünkü plasiyerlerin yazım hatalarından kaynaklanabilecek hatalara (hesaplama ve ürün isimlendirmesi gibi) açıktır. Günümüzde sipariş alma işlemi daha çok el terminalleri ile yapılmaktadır. Bu tez çalışmasının amacı el terminalinin yaptığı işlemleri mobil telefonlarla daha ucuza ve daha gelişmiş bir sistemde yapabilmektir. Ayrıca müşterilerin de kendi mobil telefonlarından sipariş vermelerini sağlamak ve böylece müşterinin plasiyerden bağımsız sipariş vermesini sağlamaktır.

6.1. Mobil Telefonla ile El Terminalinin Karşılaştırılması

Mobil telefonla sipariş alma uygulamasının el terminaline göre bazı avantajları vardır. Bunlar;

- Mobil telefonlar içindeki elektronik devreler yönünden el terminallerine çok benzemesine rağmen fiyat olarak el terminallerine göre çok ucuzdur. Bunun nedeni, telefonların çok geniş bir kitleye hitap etmesi, birçok firma tarafından üretilmesi ve rekabet ortamında gün geçtikçe fiyatlarının düşmesidir.
- Mobil telefonların hemen hemen herkes tarafından kullanılıyor olması ve uygulamayı kullanacak olan firmaya ek bir maliyet getimemesi.
- J2ME platformunda yazıldığı için mobil telefonların büyük bir kısmı tarafından desteklenmesi ve değişen isteklere göre programın kolaylıkla geliştirilebilmesi.
- Mobil telefonla sipariş alma uygulamasında stoktaki ürün bilgileri internet üzerinden anlık olarak güncellendiğinden sipariş alma işleminin stok durumuna göre yapılabilmesi.

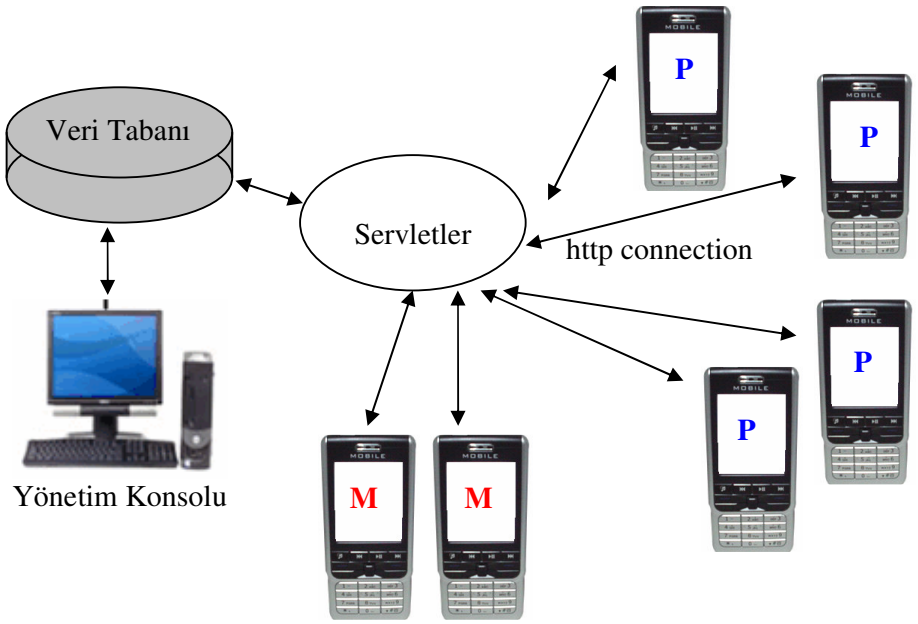
Mobil telefonla sipariş alma uygulamasının el terminaline göre bazı olumsuz yönleri de bulunmaktadır. Bunlar;

- Mobil telefonların tuş takımının kısıtlı olması.
- Ekran boyutlarının küçük olması.
- Toz ve darbeye karşı daha hassas olması.

6.2. Mobil Telefonla Sipariş Alma Uygulamasının Yapısı

Mobil telefonla sipariş alma uygulamasında kullanılacak donanımlar; plasiyer sayısı kadar CLDC 1.0 ve MIDP 1.0 destekli internet bağlantısı olan mobil telefondan, yönetim konsolu ve stok veri tabanının bulunacağı internet bağlantısı olan bir bilgisayardan oluşmaktadır. Bunun yanı sıra özellikle çok sık sipariş veren müşterilerin kendi mobil telefonlarında kullanacakları sipariş verme

programı da bu uygulamada bulunacaktır. Şekil 6.1'de sistemin yapısı basit olarak görülmektedir.



Şekil 6.1: Mobil telefonla sipariş alma sisteminin yapısı

Yönetim konsolundaki program standart stok kontrolü işini yapan programlara benzemektedir. Buradan stoktaki ürünler takip edilebilmekte, yeni ürünler ve müşteriler eklenebilmekte, plasiyerlerin satış performansları takip edilebilmektedir. Ayrıca plasiyerlerden veya müşterilerden herhangi biri sipariş verdiğinde işlem Yönetim konsolunda anında görüntülenerek kullanıcıyı bilgilendirmektedir.

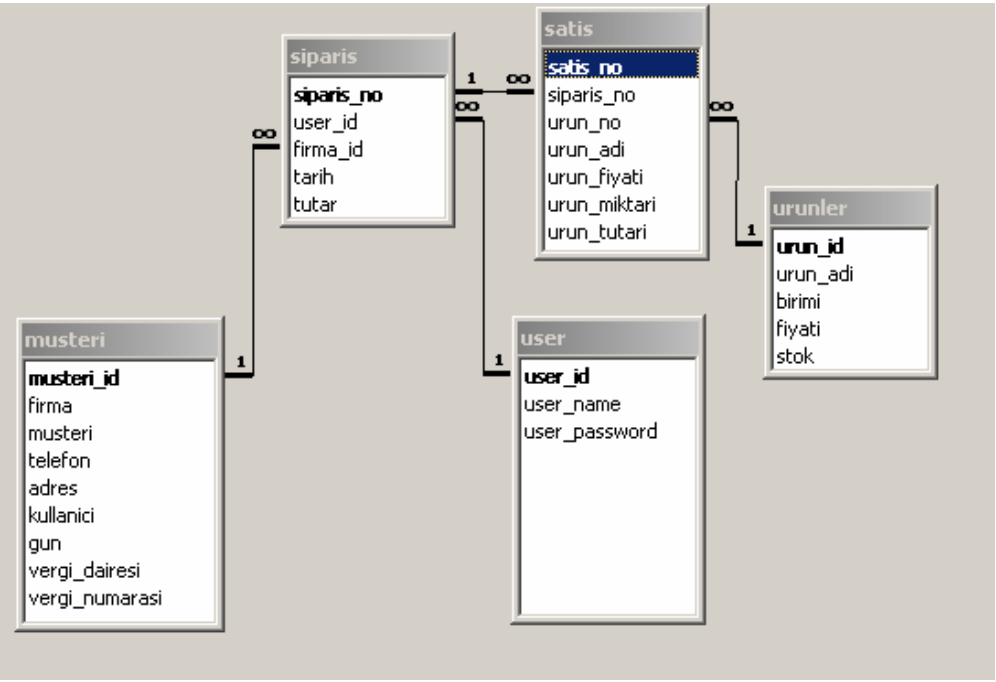
Mobil telefonlardaki uygulama yazılımı ise servletlere internet üzerinden veri göndererek servletleri çalıştıran ve servletlerden gönderilen sonuçları ekrana yazdıran program kodlarından oluşmaktadır. Bu noktada asıl işi yapan kısım servletlerdir.

Servletler veri tabanına bağlanarak kendisinden istenen verileri listeleyerek mobil telefona göndermektedir.

Sistemde mobil telefonlar için iki farklı yazılım mevcuttur. Yazılımlardan birincisi plasiyerler için, ikincisi ise müşteriler içindir. Müşteri yazılımı plasiyer yazılımının kısıtlanmış bir versiyonudur.

6.3.Uygulamada Kullanılan Veri Tabanı

Mobil telefonla sipariş alma uygulamasında basit, anlaşılır ve kurulumu kolay olması nedeniyle Access veri tabanı kullanılmıştır. Ancak bunun dışında MySQL veya MsSQL veri tabanı da kullanılanilir. Servletlerin veritabanı ile iletişime geçebilmesi için önemli olan sadece veri kaynakları (ODBC) ayarları kısmında "veri" isimli bir DSN'nin oluşturulması ve veri kaynağı olarakta uygulamanın veri tabanının seçilmesidir. Şekil 6.2'de veri tabanında kullanılan tablolar, bu tablo içindeki alanlar ve tablolar arasındaki ilişkiler görülmektedir.



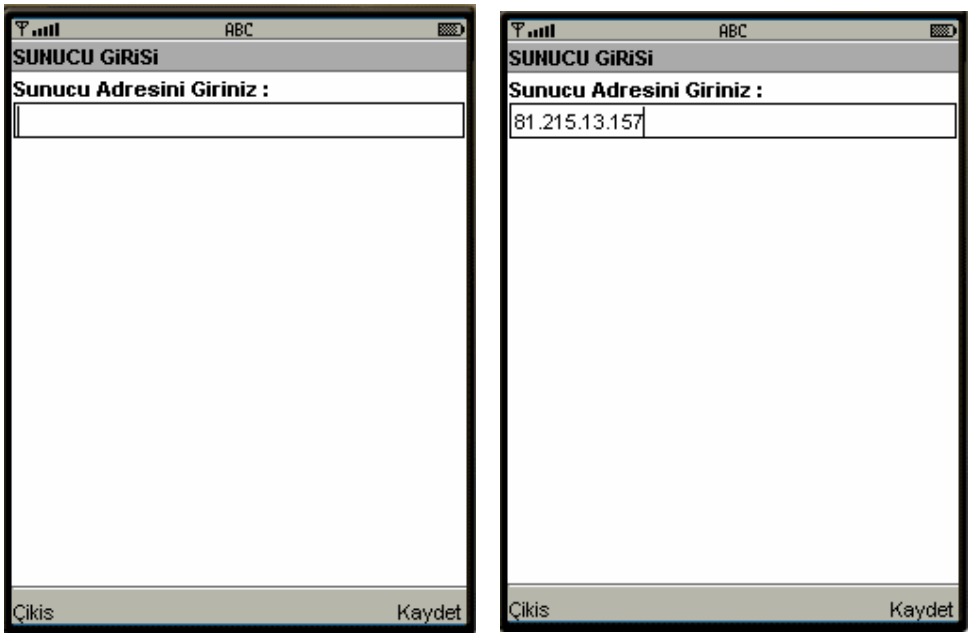
Şekil 6.2: Uygulamada kullanılan veri tabanının yapısı

6.4. Plasiyer Sipariş Alma Programı

Uygulamanın plasiyer sipariş alma kısmında plasiyer tarafından mobil telefon ile ürünlerin stok durumu takip edilebilmekte, yeni müşteri eklenebilmekte, müşterilerin eski siparişleri incelenebilmekte ve yeni siparişleri alınabilmektedir.

6.4.1. Sunucu IP adresinin belirlenmesi

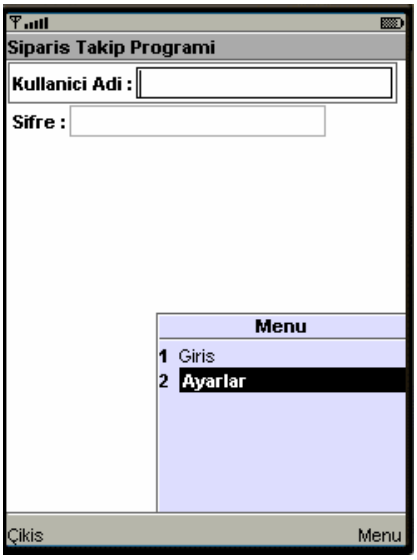
Program mobil cihaza kurulduğunda, ilk çalışmada sunucu bilgisayarın (servletlerin bulunduğu bilgisayarın) IP numarasını isteyecektir (Şekil 6.3a). Bu numara mobil telefonun dosya sistemine kaydedilecek ve sonraki çalışmalarda bu numara kullanılacaktır. Sunucu adresi IP numarası şeklinde veya domain adresi şeklinde girilebilir. Adres girildikten sonra **Kaydet** tuşuna basılarak bir sonraki ekrana geçilir. **Çıkış** tuşuna basıldığında ise programdan çıkılır (Şekil 6.3b).



(a) **(b)**

Şekil 6.3 Sunucu adresini girme ekranı (a) ve sunucu adresinin girilmesi (b)

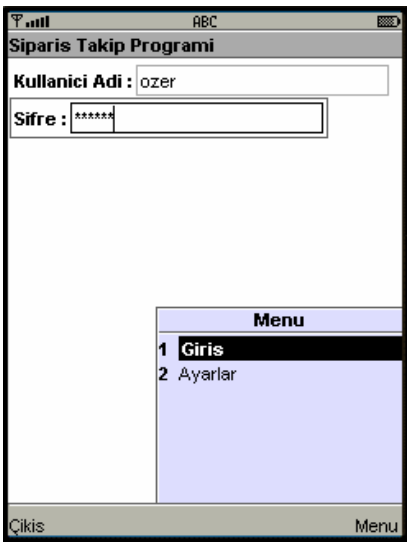
Eğer IP numarası yanlış girilirse veya sunucu bilgisayarın IP numarası değiştiğinde bu numara programın ilk açıldığı ekrandaki ayarlar kısmından değiştirilebilmektedir (Şekil 6.4).



Şekil 6.4: Sunucu adresinin değiştirilmesi

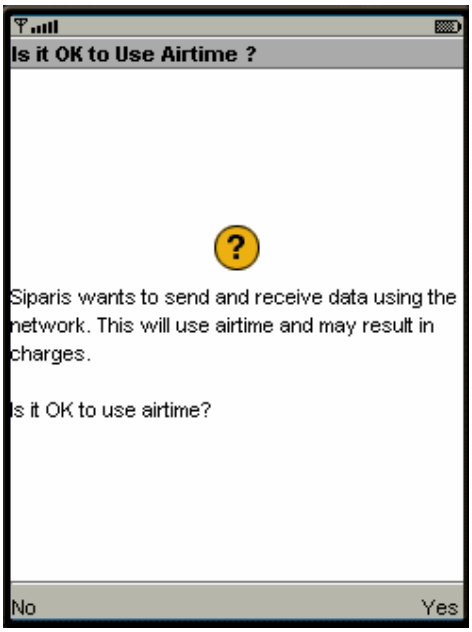
6.4.2. Kullanıcı giriş ekranı

Programın bu kısmında plasiyer kendi kullanıcı adını ve şifresini girer. Kullanıcı adı harf ve/veya rakamlardan oluşmaktadır. Şifre ise sadece rakamlardan oluşmaktadır ve * sembolleri ile gizlenmektedir (Şekil 6.5).



Şekil 6.5: Kullanıcı giriş ekranı

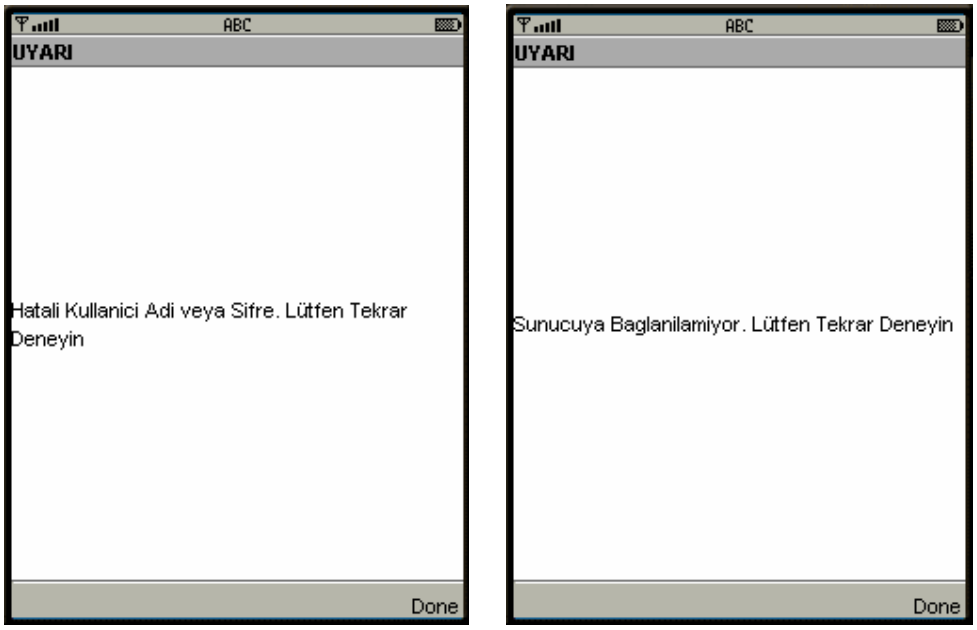
Kullanıcı adı ve şifre girildikten sonra Giriş düğmesine basılır. Bu esnada telefonun işletim sistemi, programın internet üzerinden bir veri göndereceğini, buna izin verip vermediğinizi sorar. Bu sadece bir kez olur. Programın diğer kısımlarında bu soru sorulmaz (Şekil 6.6).



Şekil 6.6: İnternet üzerinden veri gönderilmesi için onay ekranı

Bu soruya "İzin ver" şeklinde cevap verildiğinde mobil telefon Sunucudaki **Login** servletine kullanıcı adı ve şifre bilgilerini gönderir. Servlet ise veri tabanına bağlanarak **user** tablosunda sorgulama yaparak bu kullanıcı adı ve bu şifreye sahip bir kaydın olup olmadığına bakar. Böyle bir kayıt varsa bu kaydın Id numarasını mobil telefona gönderir. Mobil telefon servletten dönen numarayı **Kullanıcı Id** olarak alır ve programın ana menüsünü görünür hale getirir. Eğer servletten herhangibir numara geri gelmezse, bu kullanıcı adı ve/veya şifrenin yanlış olduğunu gösterir. Uygulama gerekli uyarı mesajını göstererek tekrar kullanıcı giriş ekranını gösterir (Şekil 6.7a).

Eğer Login servletine bağlanırken bir hata meydana gelirse (sunucu kapalıysa, sunucu adresi yanlışsa veya internete bağlanmada bir sorun varsa) uygulama ilgili mesajı göstererek tekrar kullanıcı giriş ekranını gösterir (Şekil 6.7b).



(a) **(b)**

Şekil 6.7: Hatalı kullanıcı adı veya şifre uyarısı (a) ve sunucuya bağlanamama uyarısı (b)

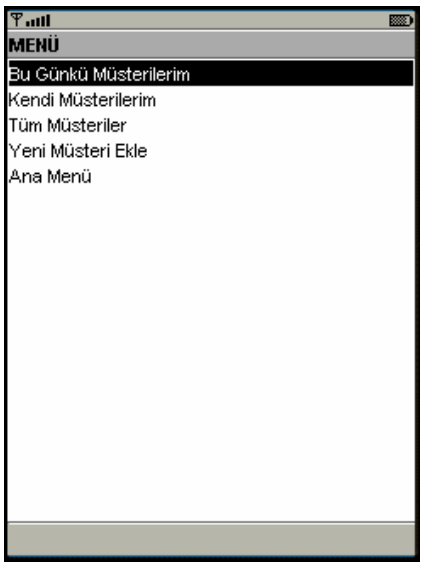
6.4.3. Ana menü

Programın bu kısmında üç seçenek bulunmaktadır. İlk seçenek MÜŞTERİLER, ikinci seçenek ÜRÜNLER ve üçüncü seçenek ise ÇIKIŞ'tır (Şekil 6.8).

Şekil 6.8: Ana menü

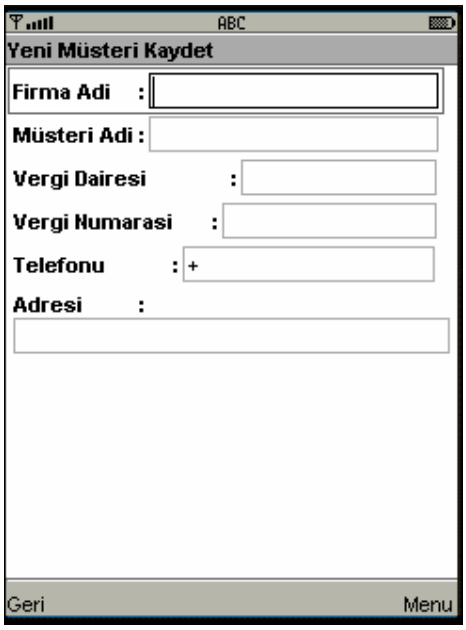
6.4.4. Müşteriler menüsü

Bu menüde müşteriler ile ilgili işlemler yapılır. Plasiyer müşteri ekleme, tüm müşterileri listeleme, sadece kendi müşterilerini listleme ve sadece o günkü müşterilerini listeleme işlemlerini yapabilir (Şekil 6.9).



Şekil 6.9: Müşteriler menüsü

Yeni müşteri ekleme seçeneği seçildiğinde müşteri ekleme ekranı açılır (Şekil 6.10). Plasiyer bu ekranı kullanarak yeni bir müşteri ekleyebilir. Müşteriden sipariş alma günü olarak müşteri ekleme günü belirlenir. Örneğin müşteri pazartesi günü eklenmişse plasiyerin pazartesi günü müşterileri arasına eklenir.



Şekil 6.10: Yeni müşteri ekleme ekranı

Yeni müşteri ekleme ekranında tüm bilgilerin girilmesi zorunludur. Aksi taktirde ekleme işlemine izin verilmemektedir. Müşteri ekleme işlemi **MusteriEkle** servleti ile yapılır. Uygulama servlete kullanıcı Id ile birlikte haftanın gününü ve müşteri bilgilerini gönderir. Eğer müşteri veri tabanına eklenirse uygulama kullanıcıya bunu bir mesaj ile bildirir. Eğer bir hata oluşmuşsa ve müşteri veri tabanına eklenememişse yine uygulama bunu kullanıcıya bildirir.

Tüm müşterileri listeleme, kendi müşterilerini listeleme ve o günkü müşterileri listeleme seçenekleri **MusteriListele** servletini çalıştırır. Parametre olarak gerekirse kullanıcı Id sini ve haftanın gününü gönderir. **MusteriListele** servleti ise istenen kayıtları **musteri** tablosundan çekerek uygulamaya gönderir. Uygulama bu listeyi bir menü haline getirerek ekrana yansıtır (Şekil 6.11).

Şekil 6.11: Müşteriler menüsü

Bu menüden işlem yapılmak istenen müsteri seçilip mobil telefonun **select/seç** tuşuna basılarak seçilen müşteri ile ilgili işlemlerin menüsü gelir (Şekil 6.12).

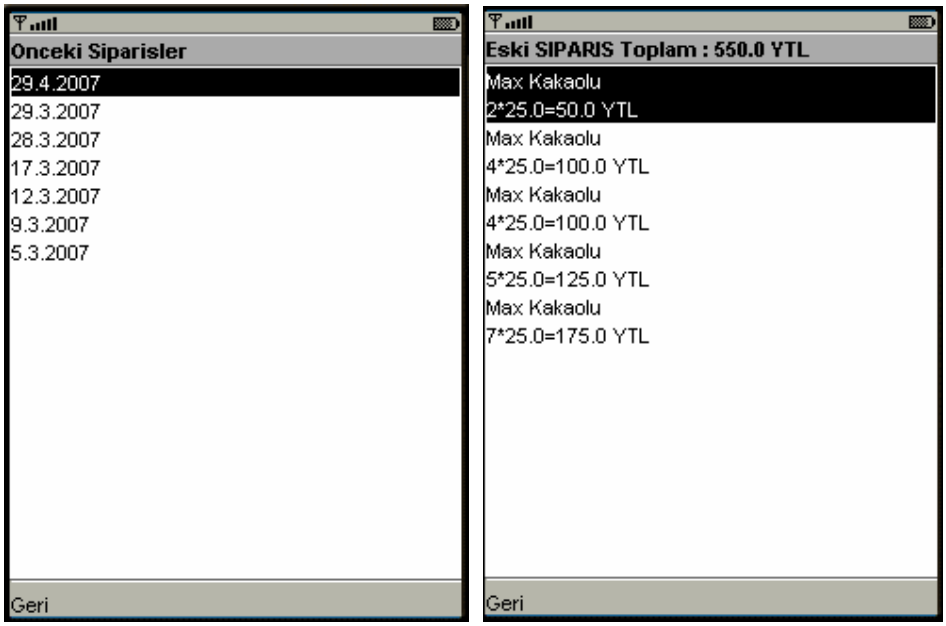
Şekil 6.12: Müşteri işlemleri menüsü

Müşteri işlemleri menüsünde müşteri bilgileri incelenebilir, müşterinin daha önceki haftalarda vermiş olduğu siparişler incelenebilir ve müşterinin o günkü siparişleri ile ilgili işlemler yapılabilir.

Firma bilgileri seçildiğinde firmanın tüm bilgileri ayrıntılı bir şekilde gösterilir (Şekil 6.13). Bu bilgileri **MusteriBilgileri** servleti veri tabanından çekerek uygulamaya gönderir.

Şekil 6.13: Müşteri bilgileri ekranı

Eski siparişler seçeneği seçildiğinde ise firmanın en son yaptığı 10 siparişin tarihi listelenir (Şekil 6.14a). Bu işlemi **EskiSiparisListele** servleti gerçekleştirir. Uygulama servlete Firma No değişkenini gönderir. Listelenen tarihlerden birisi seçilerek o tarihteki siparişin içeriği ve toplam tutarı görülebilir (Şekil 6.14b). Sipariş ayrıntılarının veri tabanından çekilmesi işlemini ise **SiparisListele** servleti gerçekleştirir. **SipariListele** servleti hem eski tarihli siparişi hemde yeni siparişin ayrıntılarını listelemek için kullanılır. Uygulama parametre olarak **Siparis No** değişkenini servlete gönderir.

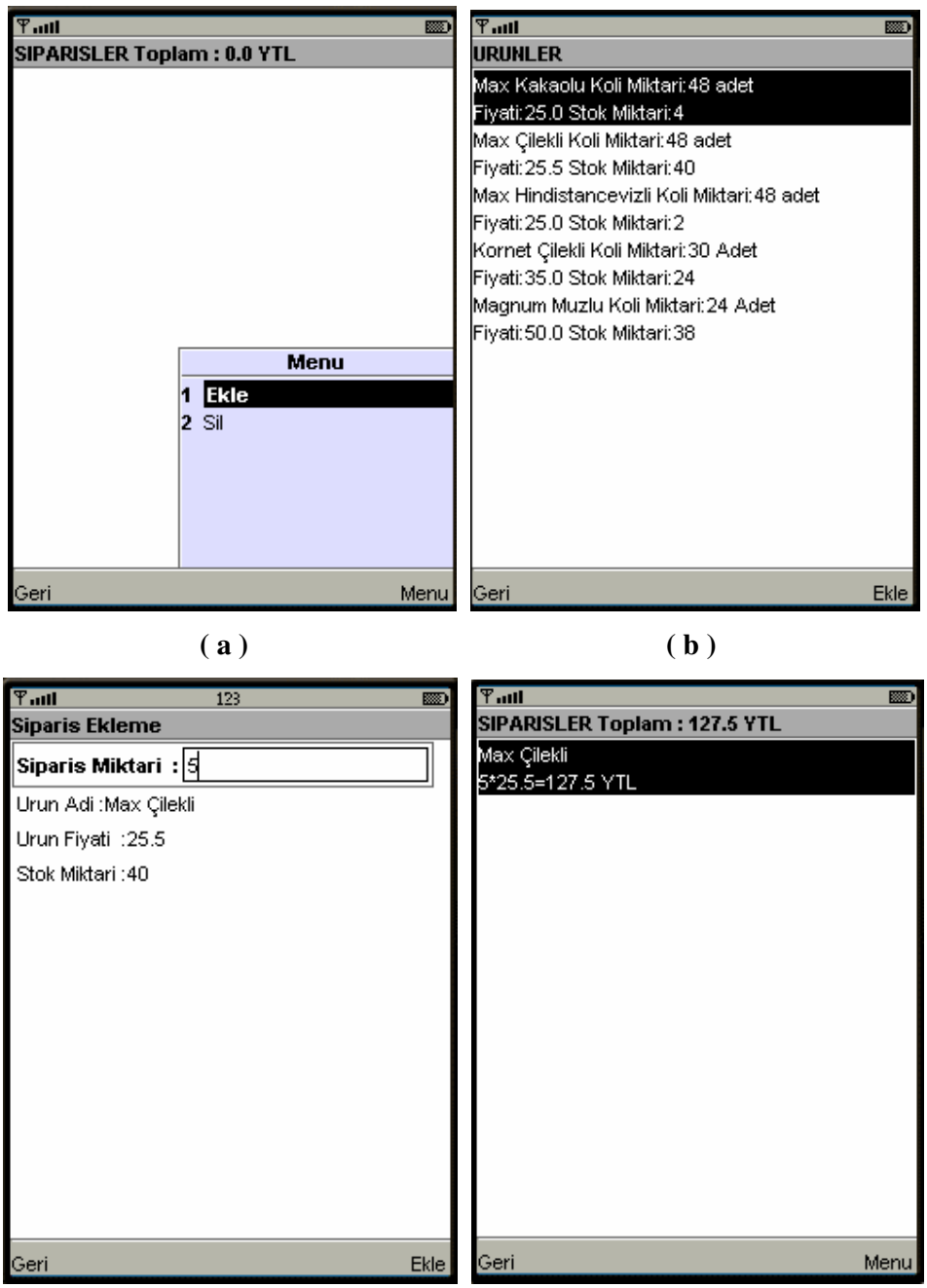


(a) **(b)**

Şekil 6.14: Eski siparişlerin listesi (a) ve eski siparişlerden birisinin ayrıntıları (b)

Bu günkü sipariş seçeneği seçildiğinde firmanın bu günkü sipariş ayrıntıları gelir. Sipariş alma işlemi henüz başlamışsa bu ekrandaki liste boş gelecektir ve toplam tutarda 0 YTL olacaktır (Şekil 6.15a). Bu ekranda menüden ekle seçeneğine tıklandığında stoklardaki ürünlerin bir listesi görüntülenecektir. Bu listede her ürün için ürünün ismi, kolideki miktarı, fiyatı ve stokta bulunan miktar bilgileri görüntülenmektedir (6.15b). Plasiyer bu ürünlerden müşterinin istediğini seçecektir. Uygulama seçilen ürünle ilgili bir ekran açacak ve plasiyer bu ekrana müşterinin istediği miktarı girecektir (6.15c). Eğer stokta o üründen istenen miktarda varsa sipariş ekleme işlemi gerçekleşecek ve müşterinin sipariş listesi görüntülenecektir (6.15d). Eğer üründen stokta yeterli miktarda yoksa kullanıcı bir mesajla bilgilendirilerek tekrar sipariş listesi görüntülenecektir.

Bu işlemler müşterinin siparişleri tamamlanıncaya kadar devam edecektir. Sipariş ekleme işlemi **SiparisEkle** servleti ile gerçekleştirilmektedir. Uygulama servlete sipariş numarasını, ürün numarasını ve miktarını göndermektedir.



Şekil 6.15: Müşteri siparişi listesinin boş hali (a), ürünlerin listesi (b), seçilen ürünün sipariş miktarının girilmesi (c) ve müşteri sipariş listesinin sipariş eklenmiş hali (d)

6.4.5. Ürünler menüsü

Bu seçenek seçildiğinde stoklardaki tüm ürünler listelenir. Bu listede her ürün için ürünün adı, koli miktarı, ürün fiyatı ve stoktaki miktarı gösterilmektedir (Şekil 6.16). Ürünlerin listesini uygulamaya **UrunleriListele** servleti gönderir. Bu servlet veri tabanındaki tüm ürünleri çekip uygulamaya gönderir.

Şekil 6.16: Ürün listesi ekranı

6.5. Müşteri Sipariş Verme Programı

Uygulamanın bu kısmı plasiyer sipariş alma programının daraltılmış bir versiyonudur. Bu program çok sık sipariş veren müşterilerin mobil telefonlarına yüklenmelidir. Müşteri bu program aracılığıyla siparişini kendisi verebilmektedir.

6.6. Uygulamanın Yönetim Konsolu

Uygulamanın son bölümü yönetim konsolu bölümüdür. Bu bölümün kodları Visual Basic ortamında yazılmıştır. Yönetim konsolu ile stok takibi yapılabilmekte, stoğa yeni ürün eklenebilmekte, her plasiyerin performansı takip edilebilmekte, müşterilerin hangi tarihte hangi üründen ne kadar ürün siparişinde bulundukları izlenebilmekte, günlük fatura dökümü yapılabilmektedir. Ayrıca plasiyerlerin ve müşterilerin verdikleri siparişler anlık olarak yönetim konsolunun sağ alt köşesinden takip edilebilmektedir. Şekil 6.17 de yönetim konsolunun bir ekranı görülmektedir.

Şekil 6.17: Yönetim konsolu ekranı

7. UYGULAMANIN YÜKLENMESİ VE ÇALIŞTIRILMASI

7.1. Uygulamanın Mobil Telefona Yüklenmesi

Uygulamanın yükleneceği mobil telefonun Java uyumlu olması ve CLDC 1.0 (veya CLDC 1.1) konfigürasyonunu ve MIDP 1.0 (veya MIDP 2.0) profilini desteklemesi gerekmektedir. Ayrıca mobil telefonun GPRS ile internete bağlanabilmesi gerekmektedir. Uygulama cihaza data kablosu aracılığı ile ya da cihazın desteklemesi halinde Bluetooth ile yüklenebilir. Uygulamanın **Siparis.jar** adlı dosyası plasiyerlerin mobil telefonuna, **Musteri.jar** dosyası ise müşterinin mobil telefonuna yüklenmelidir.

7.2. Servletlerin Sunucu Bilgisayara Yüklenmesi

Servletlerin sunucu bilgisayara yüklenebilmesi ve çalışır hale gelmesi için öncelikle Java 2 SDK'nın yüklü olması ve bir web server programının yüklü olması gerekir. Uygulamanın test aşamasında javanın J2SDK1.4.2_13 sürümü ile web sunucu olarak Apache Tomcat 5.028 sürümü kullanılmıştır. Uygulamanın çalıştırılabilmesi için farklı java sürümleri versiyonları ve farklı web sunucular kullanılabilse de belirtilen programlar birbirleri ile %100 uyumlu ve tamamen ücretsiz yazılımlar olması nedeni ile tercih edilmiştir.

Tomcat web sunucusu kurulduktan sonra bazı ayarların yapılması gerekmektedir. Bunun için Tomcat programının yüklü olduğu klasörün bir alt klasörü olan **conf** klasöründeki **web.xml** dosyası açılmalıdır. Bu dosyada aşağıda gösterilen kodlar "<!—" ve " -->" sembolleri ile pasif edilmiş olarak gelmektedir. Bu semboller kaldırılarak aktif edilmesi gerekmektedir.

```
<!—          // Bu satır kaldırılmalıdır
  <servlet>
    <servlet-name>invoker</servlet-name>
    <servlet-class>
     org.apache.catalina.servlets.InvokerServlet
```

```
            </servlet-class>
            <init-param>
               <param-name>debug</param-name>
               <param-value>0</param-value>
            </init-param>
            <load-on-startup>2</load-on-startup>
         </servlet>
-->  // Bu satır kaldırılmalıdır

<!—          // Bu satır kaldırılmalıdır
         <servlet-mapping>
            <servlet-name>invoker</servlet-name>
            <url-pattern>/servlet/*</url-pattern>
         </servlet-mapping>
-->  // Bu satır kaldırılmalıdır
```

Yukarıda belirtilen satırlar kaldırıldıktan sonra **web.xml** dosyası kaydedilerek kapatılır. Daha sonra uygulamanın **.class** uzantılı servlet dosyaları Tomcat programının kurulu olduğu klasörün içindeki **\webapps\ROOT\WEB-INF\classes** klasörüne kopyalanmalıdır.

7.3. Veri Tabanının Sunucu Bilgisayara Yüklenmesi

Veritabani.mdb isimli dosya istenilen herhangi bir klasöre kopyalanabilir. Ancak denetim masası - yönetimsel araçlar kısmındaki veri kaynakları (ODBC) programı çalıştırılarak yeni **"veri"** adında bir sistem DSN oluşturulmalı ve veritabanı olarak **veritabani.mdb** dosyası belirtilmelidir.

7.4. Yönetim Konsolunun Yüklenmesi

Yönetim konsolu tek bir **.exe** dosyası olduğu için istenilen bir klasöre kopyalanarak çalıştırılabilir.

8. SONUÇ

Bilişim ve iletişim sektörünün hızlı ilerlemesi sonucu mobil cihazlar hayatımızın bir parçası olmuş durumda. Gün geçtikçe mobil cihazların yetenekleri artmakta, fiyatları düşmekte ve hayatımızda önemli bir yer kaplamaktadırlar.

Ülkemizde mobil telefon abonelerinin sayısına baktığımızda neredeyse herkesin en az bir mobil telefonu olduğu görülmektedir. Bu cihazların herkes tarafından kullanılıyor olması sonucunda mobil telefon üreticileri birçok uygulamayı cihazlarına entegre etmişlerdir. Artık bir mobil telefon ile sesli iletişim kurmak dışında görüntü kaydetmek, fotoğraf çekmek, müzik dinlemek, ofis dökümanlarını düzenlemek, internete bağlanmak gibi pek çok işlem gerçekleştirilebilmektedir.

Bu tez çalışmasında mobil telefona, sipariş almak için kullanılan bir el terminali özelliği kazandırılmak istenmiş ve bununla ilgili bir uygulama gerçekleştirilmiştir. Uygulama gerçekleştirilirken el terminali ile sipariş alan anan firmaların da görüşü alınmıştır. Geliştirilen uygulamanın firmaların büyük bir kısmının taleplerini karşıladığı görülmüştür.

KAYNAKLAR

[1] Altıntaş, A.B., "Java'yı Tanıyalım," *Java ve Yazılım Tasarımı,* Papatya, İstanbul, 15-20, 2003.

[2] Pekgöz, N., "Java Program Dili Esasları," *Java,* Pusula, İstanbul, 24-35, 2003.

[3] Muchow, J.W., "Configurations and Profiles," *Core J2ME Technology and MIDP*, Prentice Hall PTR , New Jersey - ABD , 90-102, 2001.

[4] Keogh J.E., "CLDC Configuration," *J2ME: The Complete Reference*, McGraw-Hill/Osborne,Atlanta – ABD, 77-85, 2003.

[5] Muchow, J.W., "What is KVM," *Core J2ME Technology and MIDP*, Prentice Hall PTR , New Jersey - ABD , 120-125, 2001.

[6] White, J., "CDC Configuration," *Java 2 Micro Edition*, Manning Publications Co., Greenwich – İngiltere, 75-87, 2002.

[7] Keogh J.E., "MIDP Profile," *J2ME: The Complete Reference*, McGraw-Hill/Osborne, Atlanta – ABD, 125-126, 2003.

[8] Stephens, M., "J2ME Profiles," *Wireless Java with J2ME in 21Days,* Sams, Indiana - ABD, 144-146, 2001.

[9] Liechty, R., *MIDP 2.0: Tutorial On Signed MIDlets*, 2004. www.forum.nokia.com

[10] Muchow, J.W., "Creating JAR File From MIDlet Group," *Core J2ME Technology and MIDP*, Prentice Hall PTR , New Jersey - ABD , 255-261, 2001.

[11] Gupta, V., "High Level User Interface API," *Wireless Programming with J2ME, Hungry Minds Inc*., New York ABd, 2002.

[12] Keogh J.E., "Managing User Interfaces," *J2ME: The Complete Reference*, McGraw-Hill/Osborne,Atlanta – ABD, 178-194, 2003.

[13] Stephens, M., "User Interface Application," *Wireless Java with J2ME in 21Days,* Sams, Indiana - ABD, 258-262, 2001.

[14] Suomela, H., *FileConnection API Developer's Guide,* 2006. www.forum.nokia.com

[15] Ghosh, S., *J2ME record management store,* 2002. http://www.ibm.com/developerworks/library/wi-rms/

[16] Muchow, J., *Working With The RMS,* 2001. http://www.microjava.com/articles/techtalk/rms

[17] Giguere, E., *Searching a Record Store,* 2004. http://developers.sun.com/mobility/midp/articles/databaserecordstore/index.html

[18] Feng, Y., *Network Programming with J2ME Wireless Devices*, 2003. http://www.wirelessdevnet.com/channels/java/features/j2me_http.phtml

[19] Chichkov, I., *Introduction to Accessing Remote Databases,* 2004. www.forum.nokia.com

[20] Arora, A., Haywood, C., ve Pabla, K.S., "Extending the Reach of Wireless With JXTA Technology," Sun Microsystems Inc., **10**, 21-25, 2002.

[21] Mahmoud, Q.H., "Datagram Connection," *J2ME Low-Level Network Programming with MIDP 2.0,* 2003. http://developers.sun.com/mobility/midp/articles/midp2network/

[22] *Hemphill, D., The Network Connection*, 2002. http://www.microjava.com/articles/techtalk/network

Ek – 1 Servlet Programının Kodları

UrunleriListele.java

```
import java.io.*;
import java.sql.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class UrunleriListele extends HttpServlet {
public void doGet(HttpServletRequest request,HttpServletResponse response) throws
IOException, ServletException{
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        ResultSet veriler=null;
        Connection con=null;
        try{
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                con=DriverManager.getConnection("jdbc:odbc:Veri","","");
                Statement sorgu=con.createStatement();
                veriler=sorgu.executeQuery("SELECT * FROM urunler");
        }catch(Exception e){
                out.println("Baglanti dizisinde Hata!");
        }
        try{
                while(veriler.next()){
                        out.println(veriler.getString("urun_id"));
                        out.println(veriler.getString("urun_adi"));
                out.println(veriler.getString("birimi"));
                        out.println(veriler.getString("fiyati"));
                        out.println(veriler.getString("stok"));
                }       // while sonu.....
        }catch(Exception e){
                out.println("while dongusunde Hata!");
        }
        try{
                veriler.close();
```

```
                con.close();
        }catch(Exception f){
                out.println("Kapatmada hata");
        }
}
}
```

SiparisSil.java

```
import java.io.*;
import java.sql.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SiparisSil extends HttpServlet {
public void doGet(HttpServletRequest request,HttpServletResponse response) throws
IOException, ServletException{
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        Connection con=null;
        ResultSet veriler=null;
        String sil=null;
        int siparis_no=0;
        float tutar=0;
        float siparis_eski_tutar=0;
        float yeni_tutar=0;
        int eski_stok=0;
        int yeni_stok=0;
        int urun_miktari=0;
        int urun_no=0;
        int satis_no=0;
        try{
                sil=request.getParameter("ad");
                String[] gelen=sil.split(":");
                satis_no=Integer.parseInt(gelen[0]);
                urun_no=Integer.parseInt(gelen[1]);
                urun_miktari=Integer.parseInt(gelen[2]);
                tutar=Float.parseFloat(gelen[3]);
```

```
            siparis_no=Integer.parseInt(gelen[4]);
}catch(Exception ee){
            out.println("request ile parametre almada hata");
}
try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con=DriverManager.getConnection("jdbc:odbc:Veri","","");
            String sqlsorgu="Delete From satis where satis_no=?";
            PreparedStatement prp=con.prepareStatement(sqlsorgu);
            prp.setInt(1,satis_no);
prp.executeUpdate();
}catch(Exception e){
            out.println("Baglanti ve Prepare de hata");
}
try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con=DriverManager.getConnection("jdbc:odbc:Veri","","");
            Statement sorgu=con.createStatement();
            veriler=sorgu.executeQuery("SELECT * FROM siparis");
}catch(Exception e){
            out.println("Baglanti dizisinde Hata!");
}
try{
            while(veriler.next())
            if(siparis_no==(veriler.getInt("siparis_no"))){
                        siparis_eski_tutar=veriler.getFloat("tutar");
                        break;
            }
}catch(Exception e){
}
try{
            con.close();
}catch(Exception f){
            out.println("Kapatmada hata");
}
try{
            yeni_tutar=siparis_eski_tutar-tutar;
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con=DriverManager.getConnection("jdbc:odbc:Veri","","");
```

```
            String sqlsorgu="Update siparis set tutar=? where siparis_no=?";
            PreparedStatement prp=con.prepareStatement(sqlsorgu);
            prp.setFloat(1,yeni_tutar);
prp.setInt(2,siparis_no);
prp.executeUpdate();
}
catch(Exception e){
            out.println("Baglanti ve Prepare de hata");
}
try{
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con=DriverManager.getConnection("jdbc:odbc:Veri","","");
            Statement sorgu=con.createStatement();
            veriler=sorgu.executeQuery("SELECT * FROM urunler");
            while(veriler.next())
            if(urun_no==(veriler.getInt("urun_id"))){
                        eski_stok=veriler.getInt("stok");
                        break;
            }
}catch(Exception e){
            out.println("Baglanti dizisinde Hata!");
}
try{
            con.close();
}catch(Exception f){
            out.println("Kapatmada hata");
}
try{
            yeni_stok=eski_stok+urun_miktari;
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            con=DriverManager.getConnection("jdbc:odbc:Veri","","");
            String sqlsorgu="Update urunler set stok=? where urun_id=?";
            PreparedStatement prp=con.prepareStatement(sqlsorgu);
            prp.setInt(1,yeni_stok);
            prp.setInt(2,urun_no);
            prp.executeUpdate();
}catch(Exception e){
            out.println("Baglanti ve Prepare de hata");
}
```

```
} // metod sonu....
}  // uygulama sonu...
```

SiparisListesi.java

```
import java.io.*;
import java.sql.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SiparisListesi extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException, ServletException{
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        ResultSet veriler=null;
        Connection con=null;
        String siparisNo=null;
        String gun=null;
        String user=null;
        try{
                siparisNo=request.getParameter("ad");
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                con=DriverManager.getConnection("jdbc:odbc:Veri","","");
                Statement sorgu=con.createStatement();
                veriler=sorgu.executeQuery("SELECT * FROM satis");
        }catch(Exception e){
                out.println("Baglanti dizisinde Hata!");
        }
        try{
                while(veriler.next()){
                        if(siparisNo.equals(veriler.getString("siparis_no"))){
                                out.println(veriler.getString("satis_no"));
                        out.println(veriler.getString("urun_no"));
                                out.println(veriler.getString("urun_adi"));
                        out.println(veriler.getString("urun_fiyati"));
                        out.println(veriler.getString("urun_miktari"));
                        out.println(veriler.getString("urun_tutari"));
```

```
                    }
                }       // while sonu.....
        }catch(Exception e){
                out.println("while dongusunde Hata!");
        }
        try{
                veriler.close();
                con.close();
        }catch(Exception f){
                out.println("Kapatmada hata");
        }
}
}
```

SiparisKontrol.java

```
import java.io.*;
import java.sql.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SiparisKontrol extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException, ServletException{
        String user=null;
        String firma=null;
        String tarih=null;
        boolean kayitVar=false;
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        ResultSet veriler=null;
        Connection con=null;
        try{
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        con=DriverManager.getConnection("jdbc:odbc:Veri","","");
        Statement sorgu=con.createStatement();
        veriler=sorgu.executeQuery("SELECT * FROM siparis");
       }catch(Exception e){
```

```
 out.println("Baglanti dizisinde Hata!");
}
 try{
 String ad=request.getParameter("ad");
 String[] gelen=ad.split(":");
 user=gelen[0];
 firma=gelen[1];
 tarih=gelen[2];
 while(veriler.next()){
                        if(gelen[0].equals(veriler.getString("user_id")))
                                if(gelen[1].equals(veriler.getString("firma_id")))
                                if(gelen[2].equals(veriler.getString("tarih")))
                                        kayitVar=true;
            }
 }
 }catch(Exception e){
 out.println("while dongusunde Hata!");
 }
 if (kayitVar==false){
            try{
                        String sqlsorgu="INSERT INTO siparis
                        (user_id,firma_id,tarih) VALUES (?,?,?)";
                        PreparedStatement prp=con.prepareStatement(sqlsorgu);
                        prp.setString(1,user);
                        prp.setString(2,firma);
                        prp.setString(3,tarih);
                        prp.executeUpdate();
 }catch(Exception e){
                        out.println("Baglanti dizisinde Hata!");
 }
 }
 try{
            veriler.close();
            con.close();
 }catch(Exception f){
            out.println("Kapatmada hata");
 }
 try{
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

```
            con=DriverManager.getConnection("jdbc:odbc:Veri","","");
            Statement sorgu=con.createStatement();
            veriler=sorgu.executeQuery("SELECT * FROM siparis");
          }catch(Exception e){
            out.println("Baglanti dizisinde Hata!");
          }
            try{
            while(veriler.next()){
                              if(user.equals(veriler.getString("user_id")))
                                        if(firma.equals(veriler.getString("firma_id")))
                                        if(tarih.equals(veriler.getString("tarih"))){
                                                  out.println
                                                            (veriler.getString("siparis_no"));
                                                  out.println
                                                            (veriler.getString("tutar"));
                              }
            }
            }catch(Exception e){}
            try{
                        veriler.close();
                        con.close();
            }catch(Exception f){
                        out.println("Kapatmada hata");
            }
}
}
```

SiparisEkle.java

```
import java.io.*;
import java.sql.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SiparisEkle extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException, ServletException{
            int eski_urun=0;
```

```
 int urun_toplam=0;
 float eski_toplam=0;
 float toplam=0;
 String siparis_no=null;
 String urun_no=null;
 String siparis_miktari=null;
 String urun_adi=null;
 String urun_fiyati=null;
 float urun_fiyati_F=0;
 String urun_miktari=null;
 float urun_tutari=0;
 response.setContentType("text/plain");
 PrintWriter out = response.getWriter();
 ResultSet veriler=null;
 Connection con=null;
 try{
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
 con=DriverManager.getConnection("jdbc:odbc:Veri","","");
 Statement sorgu=con.createStatement();
 veriler=sorgu.executeQuery("SELECT * FROM urunler");
}catch(Exception e){
 out.println("Baglanti dizisinde Hata!");
}
 try{
 String ad=request.getParameter("ad");
 String[] gelen=ad.split(":");
 siparis_no=gelen[0];
 urun_no=gelen[1];
 siparis_miktari=gelen[2];
 while(veriler.next()){
                        if(gelen[1].equals(veriler.getString("urun_id"))){
                                urun_adi=veriler.getString("urun_adi");
                                urun_fiyati=veriler.getString("fiyati");
                                urun_miktari=veriler.getString("stok");
                                break;
                        }
 }
 }catch(Exception e){
 out.println("while dongusunde Hata!");
```

```
}
try{
        veriler.close();
        con.close();
}catch(Exception f){
        out.println("Kapatmada hata");
}
if (Integer.parseInt(siparis_miktari)>Integer.parseInt(urun_miktari)){
        out.println("HATA");
}
else{
        out.println("EKLE");
        urun_tutari=(Integer.parseInt(siparis_miktari)*
        Float.parseFloat(urun_fiyati));
        urun_fiyati_F=Float.parseFloat(urun_fiyati);
        try{
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                con=DriverManager.getConnection("jdbc:odbc:Veri","","");
                String sqlsorgu="INSERT INTO satis
                (siparis_no,urun_no,urun_adi,urun_fiyati,urun_miktari,urun
                _tutari) VALUES (?,?,?,?,?,?)";
                PreparedStatement prp=con.prepareStatement(sqlsorgu);
                prp.setString(1,siparis_no);
                prp.setString(2,urun_no);
                prp.setString(3,urun_adi);
                prp.setFloat(4,urun_fiyati_F);
                prp.setString(5,siparis_miktari);
                prp.setFloat(6,urun_tutari);
                prp.executeUpdate();
        }catch(Exception e){
        out.println("Baglanti ve Prepare de hata");
        }

        try{
                con.close();
        }catch(Exception f){
                out.println("Kapatmada hata");
        }
        try{
```

```
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                con=DriverManager.getConnection("jdbc:odbc:Veri","","");
                Statement sorgu=con.createStatement();
                veriler=sorgu.executeQuery("SELECT * FROM siparis");
                while(veriler.next()){
                        if(siparis_no.equals(veriler.getString ("siparis_no"))){
                                eski_toplam=veriler.getFloat("tutar");
                                break;
                        }
                }
        }catch(Exception e){
        out.println("Baglanti ve Prepare de hata ilk");
        }
        try{
                veriler.close();
                con.close();
        }catch(Exception f){
                out.println("Kapatmada hata");
        }
        try{
                toplam=eski_toplam+urun_tutari;
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                con=DriverManager.getConnection("jdbc:odbc:Veri","","");
                String sqlsorgu="Update siparis set tutar=? where
                siparis_no=?";
                PreparedStatement prp=con.prepareStatement(sqlsorgu);
                prp.setFloat(1,toplam);
                prp.setString(2,siparis_no);
                prp.executeUpdate();
        }catch(Exception e){
        out.println("Baglanti ve Prepare de hata son");
}
        try{
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                con=DriverManager.getConnection("jdbc:odbc:Veri","","");
                Statement sorgu=con.createStatement();
                veriler=sorgu.executeQuery("SELECT * FROM urunler");
                while(veriler.next()){
                        if(urun_no.equals(veriler.getString("urun_id"))){
```

```
                                        eski_urun=veriler.getInt("stok");
                                        break;
                                }
                        }
                }catch(Exception e){
                out.println("Baglanti ve Prepare de hata ilk");
                }
                try{
                        veriler.close();
                        con.close();
                }catch(Exception f){
                        out.println("Kapatmada hata");
                }
                try{
                        urun_toplam=eski_urun-Integer.parseInt(siparis_miktari);
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                        con=DriverManager.getConnection("jdbc:odbc:Veri","","");
                        String sqlsorgu="Update urunler set stok=? where
                        urun_id=?";
                        PreparedStatement prp=con.prepareStatement(sqlsorgu);
                        prp.setInt(1,urun_toplam);
                        prp.setString(2,urun_no);
                        prp.executeUpdate();
                }catch(Exception e){
                out.println("Baglanti ve Prepare de hata son");
                out.print(e);
                }
        }
}
}
```

MusteriListele.java

```
import java.io.*;
import java.sql.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
public class MusteriListele extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException, ServletException{
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        ResultSet veriler=null;
        Connection con=null;
        String musteri=null;
        String durum=null;
        String gun=null;
        String user=null;
        try{
                musteri=request.getParameter("ad");
                String[] gelen=musteri.split(":");
                durum=gelen[0];
                gun=gelen[1];
                user=gelen[2];
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                con=DriverManager.getConnection("jdbc:odbc:Veri","","");
                Statement sorgu=con.createStatement();
                veriler=sorgu.executeQuery("SELECT * FROM musteri order by
                firma");
        }catch(Exception e){
                out.println("Baglanti dizisinde Hata!");
        }
        try{
                if (durum.equals("A"))
                while(veriler.next()){
                                out.println(veriler.getString("musteri_id"));
                                out.println(veriler.getString("firma"));
                }
                if (durum.equals("B"))
                while(veriler.next()){
                                if(user.equals(veriler.getString("kullanici")) ){
                                        out.println(veriler.getString("musteri_id"));
                                        out.println(veriler.getString("firma"));
                                }
                }
                if (durum.equals("C"))
```

```
                while(veriler.next()){
                        if(gun.equals(veriler.getString("gun"))&&
                        user.equals(veriler.getString("kullanici")) ){
                                out.println(veriler.getString("musteri_id"));
                                out.println(veriler.getString("firma"));
                        }
                }
        }catch(Exception e){
                out.println("while dongusunde Hata!");
        }
        try{
                veriler.close();
                con.close();
        }catch(Exception f){
                out.println("Kapatmada hata");
        }
}
}
```

MusteriEkle.java

```
import java.io.*;
import java.sql.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MusteriEkle extends HttpServlet {
public void doGet(HttpServletRequest request,HttpServletResponse response) throws
IOException, ServletException{
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        Connection con=null;
        String firma=null;
        String musteri=null;
        String telefon=null;
        String adres=null;
        String vdairesi=null;
        String vnumarasi=null;
```

```
String kullanici=null;
String gun=null;
try{
        musteri=request.getParameter("ad");
        String[] gelen=musteri.split(":");
        out.println("Firma  :"+gelen[0]);
        out.println("Ad  :"+gelen[1]);
        out.println("Vergi Dairesi  :"+gelen[2]);
        out.println("Vergi Numarasi  :"+gelen[3]);
        out.println("Telefon  :"+gelen[4]);
        out.println("Adres  :"+gelen[5]);
        out.println("kullanici  :"+gelen[6]);
        out.println("Gun  :"+gelen[7]);
        firma=gelen[0].replace('_',' ');
        musteri=gelen[1].replace('_',' ');
        vdairesi=gelen[2].replace('-',' ');
        vnumarasi=gelen[3];
        telefon=gelen[4];
        adres=gelen[5].replace('_',' ');
        kullanici=gelen[6];
        gun=gelen[7];
}catch(Exception ee){
        out.println("request ile parametre almada hata");
}
try{
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        con=DriverManager.getConnection("jdbc:odbc:Veri","","");
        String sqlsorgu="INSERT INTO Musteri
        (firma,musteri,telefon,adres,kullanici,gun,vergi_dairesi,
        vergi_numarasi) VALUES (?,?,?,?,?,?,?,?)";
        PreparedStatement prp=con.prepareStatement(sqlsorgu);
        prp.setString(1,firma);
        prp.setString(2,musteri);
        prp.setString(3,telefon);
        prp.setString(4,adres);
        prp.setString(5,kullanici);
        prp.setString(6,gun);
        prp.setString(7,vdairesi);
        prp.setString(8,vnumarasi);
```

```
                prp.executeUpdate();
        }catch(Exception e){
                out.println("Baglanti ve Prepare de hata");
        }
        try{
                con.close();
        }catch(Exception f){
                out.println("Kapatmada hata");
        }
} // metod sonu....
}  // uygulama sonu...
```

MusteriBilgileri.java

```
import java.io.*;
import java.sql.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MusteriBilgileri extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException, ServletException{
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        ResultSet veriler=null;
        Connection con=null;
        String numara=null;
        String firmaAdi=null;
        String musteriAdi=null;
        String vergiDairesi=null;
        String vergiNumarasi=null;
        String adres=null;
        String telefon=null;
        try{
                numara=request.getParameter("ad");
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                con=DriverManager.getConnection("jdbc:odbc:Veri","","");
                Statement sorgu=con.createStatement();
```

```
                veriler=sorgu.executeQuery("SELECT * FROM musteri where
                musteri_id= "+numara);
        }catch(Exception e){
                out.println("Baglanti dizisinde Hata!");
        }
        try{
                while(veriler.next()){
                        out.println(veriler.getString("firma"));
                        out.println(veriler.getString("musteri"));
                        out.println(veriler.getString("vergi_dairesi"));
                        out.println(veriler.getString("vergi_numarasi"));
                        out.println(veriler.getString("telefon"));
                        out.println(veriler.getString("adres"));}
        }catch(Exception e){
                out.println("while dongusunde Hata!");
        }
        try{
                veriler.close();
                con.close();
        }catch(Exception f){
                out.println("Kapatmada hata");
        }
}
}
```

Login.java

```
import java.io.*;
import java.sql.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Login extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException, ServletException{
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        ResultSet veriler=null;
```

```
        Connection con=null;
        try{
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                con=DriverManager.getConnection("jdbc:odbc:Veri","","");
                Statement sorgu=con.createStatement();
                veriler=sorgu.executeQuery("SELECT * FROM user");

        }catch(Exception e){
                out.println("Baglanti dizisinde Hata!");
        }
        try{
                String ad=request.getParameter("ad");
                String[] gelen=ad.split(":");
                while(veriler.next()){
                        if(gelen[0].equals(veriler.getString("user_name")))
                            if(gelen[1].equals(veriler.getString("user_password"))){
                                out.print(veriler.getString("user_id"));
                            }
                }       // while sonu.....
        }catch(Exception e){
                out.println("while dongusunde Hata!");
        }
        try{
                veriler.close();
                con.close();
        }catch(Exception f){
                out.println("Kapatmada hata");
        }
}
}
```

EskiSiparisListele.java

```
import java.io.*;
import java.sql.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*
```

```
public class EskiSiparisListele extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException, ServletException{
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        ResultSet veriler=null;
        Connection con=null;
        String musteri=null;
        String tarih=null;
        String tarih_=null;
        int sayac;
        try{
                musteri=request.getParameter("ad");
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                con=DriverManager.getConnection("jdbc:odbc:Veri","","");
                Statement sorgu=con.createStatement();
                veriler=sorgu.executeQuery("SELECT * FROM siparis where
                firma_id="+musteri+ " order by siparis_no DESC" );
        }catch(Exception e){
                out.println("Baglanti dizisinde Hata!");
        }
        try{
        java.util.Calendar calendar = Calendar.getInstance();
        calendar.setTime(new java.util.Date());
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH);
        int date = calendar.get(Calendar.DATE);
        tarih=date+"."+month+"."+year;
        sayac=0;
                while(veriler.next()){
                        tarih_=veriler.getString("tarih");
                        if ( tarih.equals(tarih_)==false && sayac<10){
                                out.println(veriler.getString("siparis_no"));
                                out.println(tarih_);
                                out.println(veriler.getString("tutar"));
                                sayac++;
                        }
                }       // while sonu.....
        }catch(Exception e){
```

```
                out.println("while dongusunde Hata!");
        }
        try{
                veriler.close();
                con.close();
        }catch(Exception f){
                out.println("Kapatmada hata");
        }
}
}
```

Ek – 2 Siparis Takip Programının Kodları

SiparisTakip.java (Midlet)

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
import javax.microedition.io.*;
import java.io.*;
import java.util.*;

public class SiparisTakip extends MIDlet implements CommandListener {

//********************GENEL DEĞİSKENLER***********************************//

public Display display=Display.getDisplay(this);
public int siraNo=0;                        //kullan?c?n?n id numaras?
public String kullaniciId=null;                 //kullan?c?n?n id numaras?
public String URL=null;
public String tarih=null;
public String siparisTutari=null;
public int siparisNo=0;

//**********SUNUCU AYARLARI İLE İLGİLİ DEĞİŞKENLER**********************//

String SunucuAdi=null;
TextField TxtSunucu=new TextField("Sunucu Adresini Giriniz :","",40,TextField.ANY);
Command CmdSunucuCikis=new Command("Çikis",Command.STOP,1);
Command CmdSunucuKaydet=new Command("Kaydet",Command.OK,1);
Form FrmSunucu=null;
public RecordStore KayitSunucu = null;
static final String REC_STORE = "kayit";


//*********SİFRE GİRME EKRANI DEĞİŞKENLERİ*******************************//
TextField TxtKullaniciAdi=new TextField("Kullanici Adi :","",15,TextField.ANY);
TextField TxtSifre      =new TextField("Sifre :","",15,TextField.PASSWORD +
TextField.NUMERIC);
Command CmdSifreCikis=new Command("Çikis",Command.STOP,1);
```

```
Command CmdSifreGiris=new Command("Giris",Command.OK,1);
Command CmdSifreAyarlar=new Command("Ayarlar",Command.OK,1);
Form FrmGiris=null;

//*************ANA MENÜ LİSTE DEĞİŞKENLERİ*******************************//
List LstAnaMenu;
String anaMenuSecenekler[] = {"MUSTERILER", "ÜRÜNLER","ÇIKIS" };
int anaMenuSecilen=0;

//*************Müsteriler Menüsü değişkenleri*********************************//
List LstMusteriMenu;
String musteriMenuSecenekler[] = {"Bu Günkü Müsterilerim","Kendi Müsterilerim","Tüm
Müsteriler",
                                  "Yeni Müsteri Ekle","Ana Menü"};
int musteriMenuSecilen=0;

//***********Musteri Ekleme Formu değişkenleri*******************************//
TextField TxtFirma=new TextField("Firma Adi     :","",15,TextField.ANY);
TextField TxtMusteriAdi=new TextField("Müsteri Adi :","",15,TextField.ANY);
TextField TxtVergiDairesi=new TextField("Vergi Dairesi              :","",10,TextField.ANY);
TextField TxtVergiNumarasi=new TextField("Vergi Numarasi      :","",11,TextField.NUMERIC);
TextField TxtAdres=new TextField("Adresi          :","",100,TextField.ANY);
TextField TxtTelefon=new TextField("Telefonu           :","",13,TextField.PHONENUMBER);
Command CmdMusteriKaydet=new Command("Kaydet",Command.OK,1);
Command CmdMusteriKaydetGeri=new Command("Geri",Command.BACK,1);
Command CmdMusteriKaydetTemizle = new Command("Temizle",Command.OK,1);
Form FrmMusteriKaydet=null;
int HataMusteriEkle=0;

//******************Musteri Listeleme değişkenleri**************************//
int HataMusteriListele=0;
public String musteriIdDizi[], musteriAdiDizi[];
public int musteriSayisi=0;
List LstMusteriler;
Command CmdLstMusterilerGeri=new Command("Geri",Command.BACK,1);

//******************Musteri ?slemleri Menüsü değişkenleri*******************//
int secilenMusteriNo=0;
String secilenMusteriAdi=null;
```

```
int secilenMusteriId=0;
String musteriIslemleriMenuList[] = {"Bugünkü Siparis", "Eski Siparisler","Firma Bilgileri"};
List LstMusteriIslemleri;
Command CmdMusteriIslemleriGeri=new Command("Geri",Command.BACK,1);
int musteriIslemleriSecilen=0;

//**********Musteri Bilgileri inceleme Ekranı Değişkenleri*********//
int HataMusteriBilgileri=0;
String inceleFirmaAdi=null;
String inceleMusteriAdi=null;
String inceleVergiDairesi=null;
String inceleVergiNumarasi=null;
String inceleTelefon=null;
String inceleAdres=null;
Form FrmMusteriBilgileri;
Command CmdMusteriBilgileriGeri=new Command("Geri",Command.BACK,1);

//**********Urunler Listesi Değişkenleri**************************//
String UrunIdDizi[],UrunAdiDizi[], UrunBirimiDizi[], UrunFiyatiDizi[],UrunStokMiktariDizi[];
List LstUrunler;
Command CmdUrunlerGeri=new Command("Geri",Command.BACK,1);
int urunSayisi=0;
int HataUrunler=0;


//**********Musteri Siparisleri Değişkenleri**************************//
int HataSiparisKontrol=0;
int HataSiparisListele=0;
int siparisSayisi=0;
String SatisNoDizi[],
SatisUrunIdDizi[],SatisUrunDizi[],SatisUrunFiyatiDizi[],SatisUrunMiktariDizi[],SatisUrunTutari
Dizi[];
List LstMusteriSatisUrunler;   // müsterinin siparis etti?i ürünler
List LstUrunlerSiparis;        // Müsterinin secece?i ürünler
Command CmdSiparisListesiGeri=new Command("Geri",Command.BACK,1);
Command CmdSiparisListesiEkle=new Command("Ekle",Command.OK,1);
Command CmdSiparisListesiSil=new Command("Sil",Command.OK,1);
Command CmdUrunlerSiparisGeri=new Command("Geri",Command.BACK,1);
Command CmdUrunlerSiparisEkle=new Command("Ekle",Command.OK,1);
```

```
//***************Siparis ekleme formu değişkenleri*************************//
Form FrmSiparis;
TextField TxtSiparisMiktari=new TextField("Siparis Miktari  :","",13,TextField.NUMERIC);
Command CmdSiparisEkle = new Command("Ekle",Command.OK,1);
Command CmdSiparisGeri = new Command("Geri",Command.BACK,1);
int secilenUrun=0;
int siparisEklendi=0;
int HataSiparisEkleme=0;

//**************ProMesaj Formu değişkenleri*********************************//
Form FrmMesaj;
Command CmdTamam = new Command("Tamam",Command.OK,1);

//**************Siparis Silme Değişkenleri*********************************//
Form FrmSilOnay;
Command CmdSilEvet = new Command("Evet",Command.OK,1);
Command CmdSilHayir = new Command("Hayir",Command.BACK,1);
int sil=0;
int HataSiparisSil=0;

//***************Eski Siparislerin Listelenmesi Değişkenleri******************//
String eskiSiparisNoDizi[],eskiSiparisTarihDizi[],eskiSiparisTutarDizi[];
int eskiSiparisSayisi=0;
int HataEskiSiparis=0;
List eskiSiparisler;
Command CmdEskiSiparislerGeri = new Command ("Geri",Command.BACK,1);
String EskiSiparisTutar,EskiSiparisNo,EskiSiparisTarih;

public SiparisTakip()
{
//**********SUNUCU FORMUNUN HAZIRLANMASI*****************************//

FrmSunucu=new Form("SUNUCU GiRiSi");
FrmSunucu.append(TxtSunucu);
FrmSunucu.addCommand(CmdSunucuCikis);
FrmSunucu.addCommand(CmdSunucuKaydet);
```

```
//*********SİFRE GİRME EKRANININ HAZIRLANMASI**************************//
FrmGiris=new Form("Siparis Takip Programi");
FrmGiris.append(TxtKullaniciAdi);
FrmGiris.append(TxtSifre);
FrmGiris.addCommand(CmdSifreCikis);
FrmGiris.addCommand(CmdSifreGiris);
FrmGiris.addCommand(CmdSifreAyarlar);

//**********Musteri Kaydetme Ekranının Hazırlanması*************************//
FrmMusteriKaydet =new Form("Yeni Müsteri Kaydet");
FrmMusteriKaydet.append(TxtFirma);
FrmMusteriKaydet.append(TxtMusteriAdi);
FrmMusteriKaydet.append(TxtVergiDairesi);
FrmMusteriKaydet.append(TxtVergiNumarasi);
FrmMusteriKaydet.append(TxtTelefon);
FrmMusteriKaydet.append(TxtAdres);
FrmMusteriKaydet.addCommand(CmdMusteriKaydet);
FrmMusteriKaydet.addCommand(CmdMusteriKaydetGeri);
FrmMusteriKaydet.addCommand(CmdMusteriKaydetTemizle);

//***************Musteri Bilgileri Gösterme Ekranının Hazırlanması***********//
FrmMusteriBilgileri =new Form("Müsteri Bilgileri");
FrmMusteriBilgileri.addCommand(CmdMusteriBilgileriGeri);

//***************Siparis Ekleme formunun Hazırlanması************************//
FrmSiparis = new Form("Siparis Ekleme");
FrmSiparis.append(TxtSiparisMiktari);
FrmSiparis.addCommand(CmdSiparisEkle);
FrmSiparis.addCommand(CmdSiparisGeri);

//**************ProMesaj Formunun Hazırlanması*********************************//
FrmMesaj = new Form("ILETI");
FrmMesaj.addCommand(CmdTamam);

//***************SiparisSilme Ekranının Hazırlanması************************//
FrmSilOnay =new Form("Siparis Silme");
FrmSilOnay.append("Siparisi Gerçekten Silmek istiyor musunuz?");
FrmSilOnay.addCommand(CmdSilEvet);
FrmSilOnay.addCommand(CmdSilHayir);
```

```
}
    public void startApp() {
        ClassSunucu Snc = new ClassSunucu(this);
        Snc.openRecStore();
        Snc.sunucuKontrol();
    }

    public void pauseApp() { }

    public void destroyApp(boolean unconditional) { }

    public void ProFormGoster(Form form){
    form.setCommandListener(this);
    display.setCurrent(form);
    }

    public void ProCik(){
        ClassSunucu Snc = new ClassSunucu(this);
        Snc.closeRecStore();
        destroyApp(false);
        notifyDestroyed();
    }

    public void mesajHata(String yazi){
        Alert alTest;
        alTest = new Alert("UYARI", yazi, null,AlertType.WARNING);
        alTest.setTimeout(Alert.FOREVER);
        display.setCurrent(alTest);
    }

    public void mesaj(String yazi){
        Alert alTest;
        alTest = new Alert("BILGI", yazi, null,AlertType.INFO);
        alTest.setTimeout(Alert.FOREVER);
        display.setCurrent(alTest);
    }

    public void ProLstAnaMenuGoster(){
        LstAnaMenu = new List("MENÜ", List.IMPLICIT, anaMenuSecenekler,null);
```

```
        LstAnaMenu.setCommandListener(this);
        display.setCurrent(LstAnaMenu);
    }
    public void ProLstMusteriMenuGoster(){
        LstMusteriMenu = new List("MENÜ", List.IMPLICIT, musteriMenuSecenekler,null);
        LstMusteriMenu.setCommandListener(this);
        display.setCurrent(LstMusteriMenu);
    }

    public void ProMusteriKaydetEkraniTemizle(){
        TxtFirma.setString("");
        TxtMusteriAdi.setString("");
        TxtVergiDairesi.setString("");
        TxtVergiNumarasi.setString("");
        TxtAdres.setString("");
        TxtTelefon.setString("+");
    }

    public void ProMusteriKaydet(){
    if (TxtFirma.size()==0 || TxtMusteriAdi.size()==0 ||
        TxtVergiDairesi.size()==0 || TxtVergiNumarasi.size()==0 ||
        TxtAdres.size()==0 || TxtTelefon.size()<=1){
        mesaj("Lütfen Tüm Bilgileri Eksiksiz Giriniz");
    }
    else
    {
        int dayOfWeek = Calendar.getInstance().get( Calendar.DAY_OF_WEEK );
        dayOfWeek--;
        String gun=String.valueOf(dayOfWeek);
        String kullanici=String.valueOf(siraNo);
        String TempFirma=TxtFirma.getString().replace(' ','_');
        String TempMusteriAdi=TxtMusteriAdi.getString().replace(' ','_');
        String TempVergiDairesi=TxtVergiDairesi.getString().replace(' ','_');
        String TempAdres=TxtAdres.getString().replace(' ','_');
        TempAdres=TempAdres.replace(':','_');
        HataMusteriEkle=0;

URL="http://"+SunucuAdi+":8080/servlet/MusteriEkle?ad="+TempFirma+":"+TempMusteriAdi+
":"+
```

```
TempVergiDairesi+":"+TxtVergiNumarasi.getString()+":"+TxtTelefon.getString()+":"+TempAdr
es+":"+kullanici+":"+gun;
      MusteriEkle MstrEkle = new MusteriEkle(this);
      MstrEkle.gonder();
      if (HataMusteriEkle==1){
         mesajHata("Müsteri Eklenemedi. Sunucuya Ba?lanilamiyor");
      }
      if (HataMusteriEkle==0){
         mesaj("Yeni Müsteri Eklendi.");
         ProMusteriKaydetEkraniTemizle();
      }
   }
}

   public void ProMusteriListele(int sayi){
      musteriIdDizi = new String[200];
      musteriAdiDizi= new String[200];
      MusteriListele MstrLst = new MusteriListele(this);
      switch (sayi){
         case 1:int dayOfWeek = Calendar.getInstance().get( Calendar.DAY_OF_WEEK );
               dayOfWeek--;

URL="http://"+SunucuAdi+":8080/servlet/MusteriListele?ad=C:"+dayOfWeek+":"+siraNo;break;
         case 2:URL="http://"+SunucuAdi+":8080/servlet/MusteriListele?ad=B:0:"+siraNo;break;
         case 3:URL="http://"+SunucuAdi+":8080/servlet/MusteriListele?ad=A:0:0";break;
      }
      MstrLst.listele();
      if (HataMusteriListele==1){
         mesajHata("Müsteriler Listelenemiyor. Sunucuya Ba?lanilamadi.");
      }
      else{
         ProMusteriListesiGoster();
      }
   }

   public void ProMusteriListesiGoster(){
      int i=0;
      LstMusteriler= new List("MÜSTERILER",List.IMPLICIT);
```

```
        for(i=1;i<=musteriSayisi-1;i++)
            LstMusteriler.append(musteriAdiDizi[i],null);
        LstMusteriler.addCommand(CmdLstMusterilerGeri);
        LstMusteriler.setCommandListener(this);
        display.setCurrent(LstMusteriler);
    }

    public void ProMusteriIslemleriMenuGoster(){
    LstMusteriIslemleri= new
List(secilenMusteriAdi,List.IMPLICIT,musteriIslemleriMenuList,null);
    LstMusteriIslemleri.addCommand(CmdMusteriIslemleriGeri);
    LstMusteriIslemleri.setCommandListener(this);
    display.setCurrent(LstMusteriIslemleri);
    }

    public void ProMusteriBilgileriGoster(){
        URL="http://"+SunucuAdi+":8080/servlet/MusteriBilgileri?ad="+secilenMusteriId;
        MusteriBilgileri MstrBilgileri=new MusteriBilgileri(this);
        MstrBilgileri.goster();
        while (FrmMusteriBilgileri.size()>0){
            FrmMusteriBilgileri.delete(FrmMusteriBilgileri.size()-1);
        }
        FrmMusteriBilgileri.append("Firma Adi    :"+inceleFirmaAdi+'\n');
        FrmMusteriBilgileri.append("Müsteri Adi   :"+inceleMusteriAdi+'\n');
        FrmMusteriBilgileri.append("Vergi Dairesi  :"+inceleVergiDairesi+'\n');
        FrmMusteriBilgileri.append("Vergi Numarasi  :"+inceleVergiNumarasi+'\n');
        FrmMusteriBilgileri.append("Telefonu        :"+inceleTelefon+'\n');
        FrmMusteriBilgileri.append("Adresi          :"+inceleAdres+'\n');
        FrmMusteriBilgileri.setCommandListener(this);
        display.setCurrent(FrmMusteriBilgileri);
    }

    public void ProLstUrunMenu(){
        UrunIdDizi=new String[200];
        UrunAdiDizi =new String[200];
        UrunBirimiDizi =new String[200];
        UrunFiyatiDizi =new String[200];
        UrunStokMiktariDizi =new String[200];
        HataUrunler=0;
```

```
        URL="http://"+SunucuAdi+":8080/servlet/UrunleriListele";
        UrunleriListele UrnlrLst = new UrunleriListele(this);
        UrnlrLst.listele();
        if (HataUrunler==1){
            mesajHata("Urunler Listelenemiyor. Sunucuya Ba?lanilamadi.");
        }else{
            ProLstUrunMenuGoster();
        }

    }

    public void ProLstUrunMenuGoster(){
        int i=0;
        LstUrunler = new List("URUNLER", List.IMPLICIT);
        LstUrunler.addCommand(CmdUrunlerGeri);
            for(i=1;i<=urunSayisi-1;i++)
                LstUrunler.append(UrunAdiDizi[i]+" Koli Miktari:"+UrunBirimiDizi[i]+'\n'+"Fiyati:"
                +UrunFiyatiDizi[i]+" Stok Miktari:"+UrunStokMiktariDizi[i],null);
        LstUrunler.setCommandListener(this);
        display.setCurrent(LstUrunler);
    }

    public void ProMusteriBugunSiparisKontrol(){
        HataSiparisKontrol=0;
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(new Date());
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH)+1;
        int date = calendar.get(Calendar.DATE);
        tarih=date+"."+month+"."+year;
        siparisTutari="0";

URL="http://"+SunucuAdi+":8080/servlet/SiparisKontrol?ad="+siraNo+":"+secilenMusteriNo+":
"+tarih;
        SiparisKontrol SprsKntrl = new SiparisKontrol(this);
        SprsKntrl.siparisVarmi();          // Sipari? varsa siparisNo al?n?yor yoksa olu?turuluyor.
        if (HataSiparisKontrol==1){
            mesajHata("Müsteri Siparisleri Listelenemiyor. Sunucuya Baglanilamadi.");
        }else{
```

```
        ProMusteriBugunSiparisleriGetir();
    }
}

public void ProMusteriBugunSiparisleriGetir(){
    SatisNoDizi= new String[200];
    SatisUrunIdDizi = new String[200];
    SatisUrunDizi = new String[200];
    SatisUrunFiyatiDizi =new String[200];
    SatisUrunMiktariDizi = new String[200];
    SatisUrunTutariDizi = new String[200];
    URL="http://"+SunucuAdi+":8080/servlet/SiparisListesi?ad="+siparisNo;
    SiparisListele SprsLst = new SiparisListele(this);
    SprsLst.listele();
    if (HataSiparisListele==1){
        mesajHata("Müsteri Siparisleri Listelenemiyor. Sunucuya Baglanilamadi.");
    }else{
        ProMusteriBugunSiparisGoster();
    }
}

public void ProMusteriBugunSiparisGoster(){
    int i=0;
    LstMusteriSatisUrunler = new List("Tutar : "+siparisTutari+" YTL", List.IMPLICIT);
    LstMusteriSatisUrunler.addCommand(CmdSiparisListesiGeri);
    LstMusteriSatisUrunler.addCommand(CmdSiparisListesiEkle);
    LstMusteriSatisUrunler.addCommand(CmdSiparisListesiSil);
    for(i=1;i<=siparisSayisi-1;i++)
        LstMusteriSatisUrunler.append(SatisUrunDizi[i]+'\n'+SatisUrunMiktariDizi[i]+"*"
        +SatisUrunFiyatiDizi[i]+"="+SatisUrunTutariDizi[i]+" YTL",null);
    LstMusteriSatisUrunler.setCommandListener(this);
    display.setCurrent(LstMusteriSatisUrunler);
}

public void ProListSiparisUrunler(){
    UrunIdDizi=new String[200];
    UrunAdiDizi =new String[200];
    UrunBirimiDizi =new String[200];
    UrunFiyatiDizi =new String[200];
```

```
        UrunStokMiktariDizi =new String[200];
        HataUrunler=0;
        URL="http://"+SunucuAdi+":8080/servlet/UrunleriListele";
        UrunleriListele UrnlrLst = new UrunleriListele(this);
        UrnlrLst.listele();
        if (HataUrunler==1){
           mesajHata("Urunler Listelenemiyor. Sunucuya Ba?lanilamadi.");
        }else{
           ProLstUrunMenuGoster2();
        }
     }

     public void ProLstUrunMenuGoster2(){
        int i=0;
        LstUrunlerSiparis = new List("URUNLER", List.IMPLICIT);
        LstUrunlerSiparis.addCommand(CmdUrunlerSiparisGeri);
        LstUrunlerSiparis.addCommand(CmdUrunlerSiparisEkle);
           for(i=1;i<=urunSayisi-1;i++)
              LstUrunlerSiparis.append(UrunAdiDizi[i]+" Koli
Miktari:"+UrunBirimiDizi[i]+'\n'+"Fiyati:"
              +UrunFiyatiDizi[i]+" Stok Miktari:"+UrunStokMiktariDizi[i],null);
        LstUrunlerSiparis.setCommandListener(this);
        display.setCurrent(LstUrunlerSiparis);
     }

     public void ProFrmSiparisAlmaGoster(){
        String urun_adi,urun_fiyati,urun_miktari;
        secilenUrun=-1;
        if (LstUrunlerSiparis.size()>0){
           secilenUrun=LstUrunlerSiparis.getSelectedIndex();
           urun_adi=UrunAdiDizi[secilenUrun+1];
           urun_fiyati=UrunFiyatiDizi[secilenUrun+1];
           urun_miktari=UrunStokMiktariDizi[secilenUrun+1];
           if (FrmSiparis.size()>=4){
              while(FrmSiparis.size()>1)
                 FrmSiparis.delete(FrmSiparis.size()-1);
           }
           FrmSiparis.append("Urun Adi :"+urun_adi+'\n');
           FrmSiparis.append("Urun Fiyati  :"+urun_fiyati+'\n');
```

```
        FrmSiparis.append("Stok Miktari :"+urun_miktari);
        TxtSiparisMiktari.setString("");
        ProFormGoster(FrmSiparis);
    }
}

public void ProSiparisEkle(){
    siparisEklendi=0;
    HataSiparisEkleme=0;
    try{
        if (Integer.parseInt(TxtSiparisMiktari.getString())==0){
        mesajHata("0 dan farkli bir sayi giriniz");
        }
        else{
            URL="http://"+SunucuAdi+":8080/servlet/SiparisEkle?ad="+siparisNo+":"
            +UrunIdDizi[secilenUrun+1]+":"+Integer.parseInt(TxtSiparisMiktari.getString());

            SiparisEkle SprsEkle = new SiparisEkle(this);
            SprsEkle.yolla();
            if (HataSiparisEkleme==1){
                mesajHata("Siparis Eklenemedi. Sunucuya Baglanilamiyor! ");
            }
            else{
                if (siparisEklendi==1){
                    ProMesaj("Siparis Eklendi");
                }
                if (siparisEklendi==0){
                    ProMesaj("Secilen Üründen Stoklarda Yeterli Miktarda Yok");
                }
            }
        }
    }catch(Exception e){
        System.out.println(e);
        mesajHata("Siparis Miktarini Giriniz");
    }
}

public void ProSiparisSilOnay(){
    HataSiparisSil=0;
```

```
    URL="http://"+SunucuAdi+":8080/servlet/SiparisSil?ad="
              +SatisNoDizi[sil]+":"+SatisUrunIdDizi[sil]+":"+SatisUrunMiktariDizi[sil]+":"+
               SatisUrunTutariDizi[sil]+":"+siparisNo;
    SiparisSil sprsSil=new SiparisSil(this);
    sprsSil.yolla();
    System.out.println(HataSiparisSil);
    ProMusteriBugunSiparisKontrol();
}

public void ProMesaj(String ileti){
    while (FrmMesaj.size()>0)
       FrmMesaj.delete(FrmMesaj.size()-1);
    FrmMesaj.append(ileti);
    ProFormGoster(FrmMesaj);
}

public void ProSiparisSil(){
  ProFormGoster(FrmSilOnay);
}

public void ProMusteriEskiSiparisleriGoster(){
HataEskiSiparis=0;
eskiSiparisNoDizi =new String[15];
eskiSiparisTarihDizi=new String[15];
eskiSiparisTutarDizi=new String[15];
URL="http://"+SunucuAdi+":8080/servlet/EskiSiparisListele?ad="+secilenMusteriId;
EskiSiparisListele EskiSprsLst = new EskiSiparisListele(this);
EskiSprsLst.listele();
if (HataEskiSiparis==1){
    mesajHata("Eski Siparisler Listelenemiyor. Sunucuya Baglanilamadi");
}
else{
    ProLstEskiSiparisler();
}
}

public void ProLstEskiSiparisler(){
    int i=0;
    eskiSiparisler = new List("Onceki Siparisler", List.IMPLICIT);
```

```
        eskiSiparisler.addCommand(CmdEskiSiparislerGeri);
        for(i=1;i<=eskiSiparisSayisi-1;i++)
            eskiSiparisler.append(eskiSiparisTarihDizi[i],null);
        eskiSiparisler.setCommandListener(this);
        display.setCurrent(eskiSiparisler);
    }

    public void ProEskiSiparisAyrintilar(){
        SatisNoDizi= new String[200];
        SatisUrunIdDizi = new String[200];
        SatisUrunDizi = new String[200];
        SatisUrunFiyatiDizi =new String[200];
        SatisUrunMiktariDizi = new String[200];
        SatisUrunTutariDizi = new String[200];

URL="http://"+SunucuAdi+":8080/servlet/SiparisListesi?ad="+eskiSiparisNoDizi[eskiSiparisler.g
etSelectedIndex()+1];;
        SiparisListele SprsLst = new SiparisListele(this);
        SprsLst.listele();
        if (HataSiparisListele==1){
            mesajHata("Müsteri Siparisleri Listelenemiyor. Sunucuya Baglanilamadi.");
        }else{
            ProMusteriEskiSiparisGoster();
        }
    }

    public void ProMusteriEskiSiparisGoster(){
        int i=0;
        LstMusteriSatisUrunler = new List("Eski SIPARIS Toplam :
"+eskiSiparisTutarDizi[eskiSiparisler.getSelectedIndex()+1]+" YTL
Tarih:"+eskiSiparisTarihDizi[eskiSiparisler.getSelectedIndex()+1] , List.IMPLICIT);
        LstMusteriSatisUrunler.addCommand(CmdSiparisListesiGeri);
        for(i=1;i<=siparisSayisi-1;i++)
            LstMusteriSatisUrunler.append(SatisUrunDizi[i]+'\n'+SatisUrunMiktariDizi[i]+"*"
            +SatisUrunFiyatiDizi[i]+"="+SatisUrunTutariDizi[i]+" YTL",null);
        LstMusteriSatisUrunler.setCommandListener(this);
        display.setCurrent(LstMusteriSatisUrunler);
    }
```

```
public void commandAction(Command c, Displayable d) {
    if (c==CmdSunucuKaydet){
        try{
            if (KayitSunucu.getNumRecords()==0){
                ClassSunucu Snc = new ClassSunucu(this);
                Snc.writeRecord(TxtSunucu.getString());
            }
        }
        catch(Exception e){}
        try{
           if (KayitSunucu.getNumRecords()!=0){
               ClassSunucu Snc = new ClassSunucu(this);
               Snc.updateRecord(TxtSunucu.getString());
           }
        }
        catch(Exception e){}
        ProFormGoster(FrmGiris);
    }
    if (c==CmdSunucuCikis){
        ProCik();
    }
    if (c==CmdSifreCikis){
        ProCik();
    }

   if(c==CmdSifreAyarlar){
      TxtSunucu.setString(SunucuAdi);
      ProFormGoster(FrmSunucu);
   }

   if (c==CmdSifreGiris){
        String isim=TxtKullaniciAdi.getString();
        String sifre=TxtSifre.getString();
        siraNo=0;
        URL="http://"+SunucuAdi+":8080/servlet/Login?ad="+isim+":"+sifre;
        Giris grs = new Giris(this);
        grs.userFind();
   }
```

```
if (display.getCurrent()==LstAnaMenu){
 anaMenuSecilen=LstAnaMenu.getSelectedIndex();
 switch(anaMenuSecilen){
    case 0:ProLstMusteriMenuGoster();break;
    case 1:ProLstUrunMenu();break;
    case 2:ProCik();break;
 }
}
if (display.getCurrent()==LstMusteriMenu){
 musteriMenuSecilen=LstMusteriMenu.getSelectedIndex();
 switch(musteriMenuSecilen){
    case 0:ProMusteriListele(1);break;
    case 1:ProMusteriListele(2);break;
    case 2:ProMusteriListele(3);break;
    case 3:ProMusteriKaydetEkraniTemizle();ProFormGoster(FrmMusteriKaydet);break;
    case 4:ProLstAnaMenuGoster();break;
 }
}

if (display.getCurrent()==LstMusteriIslemleri){
   musteriIslemleriSecilen=LstMusteriIslemleri.getSelectedIndex();
   switch(musteriIslemleriSecilen){
      case 0:ProMusteriBugunSiparisKontrol();break;
      case 1:ProMusteriEskiSiparisleriGoster();break;
      case 2:ProMusteriBilgileriGoster();break;
   }
}

if (display.getCurrent()==LstMusteriler){
 secilenMusteriNo=LstMusteriler.getSelectedIndex();
 try{
 secilenMusteriId=Integer.parseInt(musteriIdDizi[secilenMusteriNo+1]);
 }
 catch(Exception e){}
 secilenMusteriAdi=musteriAdiDizi[secilenMusteriNo+1];
 ProMusteriIslemleriMenuGoster();
}

if (display.getCurrent()==eskiSiparisler){
```

```
    if (eskiSiparisler.size()>0){
        EskiSiparisTutar=(eskiSiparisTutarDizi[eskiSiparisler.getSelectedIndex()+1]);
        EskiSiparisNo=(eskiSiparisNoDizi[eskiSiparisler.getSelectedIndex()+1]);
        EskiSiparisTarih=(eskiSiparisTarihDizi[eskiSiparisler.getSelectedIndex()+1]);
        ProEskiSiparisAyrintilar();
    }
}
if (c==CmdMusteriKaydetGeri){
    ProLstMusteriMenuGoster();
}
if (c==CmdMusteriKaydetTemizle){
    ProMusteriKaydetEkraniTemizle();
}

if (c==CmdMusteriKaydet){
    ProMusteriKaydet();
}
if (c==CmdLstMusterilerGeri){
    ProLstMusteriMenuGoster();
}
if (c==CmdMusteriIslemleriGeri){
    ProMusteriListesiGoster();
}
if (c==CmdMusteriBilgileriGeri){
    ProMusteriIslemleriMenuGoster();
}
if (c==CmdUrunlerGeri){
    ProLstAnaMenuGoster();
}
if (c==CmdSiparisListesiGeri){
    ProMusteriIslemleriMenuGoster();
}
if (c==CmdSiparisListesiEkle){
    ProListSiparisUrunler();
}
if (c==CmdUrunlerSiparisGeri){
    ProMusteriBugunSiparisleriGetir();
}
if (c==CmdUrunlerSiparisEkle){
```

```
            ProFrmSiparisAlmaGoster();
        }
        if (c==CmdSiparisEkle){
            ProSiparisEkle();
        }
        if (c==CmdSiparisGeri){
            ProListSiparisUrunler();
        }
        if (c==CmdTamam){
            ProMusteriBugunSiparisKontrol();
        }
        if (c==CmdSiparisListesiSil){
            if (LstMusteriSatisUrunler.size()>0)
              sil=LstMusteriSatisUrunler.getSelectedIndex()+1;
              ProSiparisSil();
        }
        if (c==CmdSilEvet){
           ProSiparisSilOnay();
        }
        if (c==CmdSilHayir){
            ProMusteriBugunSiparisKontrol();
        }
        if (c==CmdEskiSiparislerGeri){
            ProMusteriIslemleriMenuGoster();
        }
    }

}
```

ClassSunucu.java

```
import java.lang.String;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
import javax.microedition.io.*;
import java.io.*;
import java.util.*;
public class ClassSunucu {
```

```
SiparisTakip SprsTkp=null;
public ClassSunucu(SiparisTakip ST) {
        SprsTkp=ST;
}
public void openRecStore(){
        try{
        SprsTkp.KayitSunucu = RecordStore.openRecordStore(SprsTkp.REC_STORE,
        true );
        }
        catch (Exception e){}
}
public void sunucuKontrol(){
        try{
        if (SprsTkp.KayitSunucu.getNumRecords()==0){
                SprsTkp.ProFormGoster(SprsTkp.FrmSunucu);
        }
                else readRecords();
        }
        catch(Exception e){}
}

public void readRecords(){
        String yedek;
        try{
                byte[] recData = new byte[40];
                int len;
                len = SprsTkp.KayitSunucu.getRecord( 1, recData, 0 );
                yedek = new String(recData,0,len);
                SprsTkp.SunucuAdi = yedek;
                SprsTkp.ProFormGoster(SprsTkp.FrmGiris);
        }catch (Exception e){}
}
public void closeRecStore(){
        try{
        SprsTkp.KayitSunucu.closeRecordStore();
        }
        catch (Exception e){}
}
public void writeRecord(String str){
```

```
                byte[] rec = str.getBytes();
                try{
                SprsTkp.KayitSunucu.addRecord(rec, 0, rec.length);
                SprsTkp.SunucuAdi=str;
                }
                catch (Exception e){}
        }
        public void updateRecord(String str){
                byte[] rec = str.getBytes();
                try{
        SprsTkp.KayitSunucu.setRecord(1,rec, 0, rec.length);
        SprsTkp.SunucuAdi=str;
                }
                catch (Exception e){}
        }
}
```

EskiSiparisListele.java

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
import javax.microedition.io.*;
import java.io.*;
class EskiSiparisListele{
        SiparisTakip SprsTkp;
        public EskiSiparisListele(SiparisTakip ST){
                SprsTkp=ST;
        }
        public void listeAl (){
                HttpConnection c=null;
                InputStream is=null;
                OutputStream os=null;
                StringBuffer b=new StringBuffer();
                StringBuffer yazi=new StringBuffer();
                try{
                        c=(HttpConnection)Connector.open(SprsTkp.URL);
                        c.setRequestMethod(HttpConnection.GET);
                        c.setRequestProperty("IF-Modified-Since", "20 Jan 2001
```

```
                    16:19:14 GMT");
                    c.setRequestProperty("User-Agent","Profile/MIDP-1.0
                    Configuration/CLDC-1.0");
                    c.setRequestProperty("Content-Language", "en-CA");
                    os=c.openOutputStream();
                    is=c.openDataInputStream();

                    int ch,sayac0,sayac1,sayac2,i,durum;
                    sayac0=1;sayac1=1;durum=0;
                    sayac2=1;
                    while((ch=is.read())!=-1) {
                    if (ch==13){
                if (durum==0){
                       SprsTkp.eskiSiparisNoDizi[sayac0++]=yazi.toString();
                    durum=1; }
                    else if (durum==1){
                       SprsTkp.eskiSiparisTarihDizi[sayac1++]=yazi.toString();
                    durum=2;
                    }else if (durum==2){
                       SprsTkp.eskiSiparisTutarDizi[sayac2++]=yazi.toString();
                    durum=0;
                    }
                    yazi.delete(0,yazi.length());
                    is.read();
                    }else{
                    yazi.append((char)ch);
                    b.append((char) ch);
                    }
                    }
                    SprsTkp.eskiSiparisSayisi=sayac2;
                    SprsTkp.HataEskiSiparis=0;
                    is.close();
                    os.close();
                    c.close();
                }catch(Exception exp){
                SprsTkp.HataEskiSiparis=1;
                System.out.println("Baglanti hatasi");
                }
}
```

```
public void listele(){
        Thread t=new Thread() {
                public void run() {
                        try {
                        listeAl();
                }catch(Exception e) {
                                SprsTkp.HataEskiSiparis=1;
                }
               }
        };
        t.start();
        while (t.isAlive()){}
}
}
```

Giris.java

```
import javax.microedition.rms.*;
import javax.microedition.io.*;
import java.io.*;
class Giris{
        SiparisTakip SprsTkp=null;
        public Giris(SiparisTakip ST){
                SprsTkp=ST;
        }
        public void login (){
                HttpConnection c=null;
                InputStream is=null;
                OutputStream os=null;
                StringBuffer b=new StringBuffer();
                try{
                        c=(HttpConnection)Connector.open(SprsTkp.URL);
                        c.setRequestMethod(HttpConnection.GET);
                        c.setRequestProperty("IF-Modified-Since", "20 Jan 2001
                        16:19:14 GMT");
                        c.setRequestProperty("User-Agent","Profile/MIDP-1.0
                        Configuration/CLDC-1.0");
                        c.setRequestProperty("Content-Language", "en-CA");
                        os=c.openOutputStream();
```

```
                    is=c.openDataInputStream();
                  int ch;
                    while((ch=is.read())!=-1) {
                            b.append((char) ch);
                    }
                    is.close();
                    os.close();
                    c.close();
                }catch(Exception exp){
                SprsTkp.siraNo=-1;
                System.out.println("Baglanti hatasi");
                }
                SprsTkp.kullaniciId=b.toString();
                        if (SprsTkp.siraNo!=-1){
                        try{
                        SprsTkp.siraNo=Integer.parseInt(SprsTkp.kullaniciId);
                        }catch(Exception exp){
                        SprsTkp.siraNo=-2;
                        }
                        }
 switch(SprsTkp.siraNo){
        case -1:SprsTkp.mesajHata("Sunucuya Baglanilamiyor. Lütfen Tekrar
Deneyin");break;
        case -2:SprsTkp.mesajHata("Hatali Kullanici Adi veya Sifre. Lütfen
                Tekrar Deneyin");break;
        case  0:System.out.println("veri gelmedi");break;
        default : SprsTkp.ProLstAnaMenuGoster();break;
 }
}

public void userFind(){
   Thread t=new Thread() {
        public void run() {
       try {
        login();
       }
       catch(Exception ef)          {
          SprsTkp.siraNo=-1;
          System.out.println("d?? hata");
```

```
        }
         }
   };
   t.start();
}
}
```

MusteriBilgileri.java

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
import javax.microedition.io.*;
import java.io.*;

class MusteriBilgileri{
SiparisTakip SprsTkp;
public MusteriBilgileri(SiparisTakip ST){
  SprsTkp=ST;
}

public void listeAl (){
                    HttpConnection c=null;
                    InputStream is=null;
                    OutputStream os=null;
                    StringBuffer b=new StringBuffer();
            StringBuffer yazi=new StringBuffer();

            try{
                    c=(HttpConnection)Connector.open(SprsTkp.URL);
                    c.setRequestMethod(HttpConnection.GET);
                    c.setRequestProperty("IF-Modified-Since", "20 Jan 2001 16:19:14 GMT");
             c.setRequestProperty("User-Agent","Profile/MIDP-1.0 Configuration/CLDC-1.0");
                    c.setRequestProperty("Content-Language", "en-CA");
                    os=c.openOutputStream();
                    is=c.openDataInputStream();

             int ch,durum;
             durum=1;
```

```
        while((ch=is.read())!=-1) {
 if (ch==13){
            if (durum==1){
                    SprsTkp.inceleFirmaAdi=yazi.toString();
                    durum=2;
            }else
               if (durum==2){
                    SprsTkp.inceleMusteriAdi=yazi.toString();
                    durum=3;
            }else
               if (durum==3){
                    SprsTkp.inceleVergiDairesi=yazi.toString();
                    durum=4;
            }else
               if (durum==4){
                    SprsTkp.inceleVergiNumarasi=yazi.toString();
                    durum=5;
            }else
               if (durum==5){
                    SprsTkp.inceleTelefon=yazi.toString();
                    SprsTkp.inceleTelefon="+ "+SprsTkp.inceleTelefon;
                    durum=6;
            }else
               if (durum==6){
                    SprsTkp.inceleAdres=yazi.toString();
                    durum=1;
            }

            yazi.delete(0,yazi.length());
            is.read();
 }else{
            yazi.append((char)ch);
            b.append((char) ch);
 }
}
SprsTkp.HataMusteriBilgileri=0;
is.close();
os.close();
c.close();
```

```
            }
        catch(Exception exp)
        {
           SprsTkp.HataMusteriBilgileri=1;
           System.out.println("Baglanti hatasi");
        }
}

public void goster(){
Thread t=new Thread() {
            public void run() {
                                    try {
                        listeAl();
                        }
                    catch(Exception e)
                        {
                           SprsTkp.HataMusteriBilgileri=1;
                        }
                      }
                    };
t.start();
while (t.isAlive()){
}
}

}
```

MusteriEkle.java

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
import javax.microedition.io.*;
import java.io.*;
import java.lang.*;

class MusteriEkle{
   SiparisTakip SprsTkp=null;
```

```
public MusteriEkle(SiparisTakip ST){
   SprsTkp=ST;
}

public void kayitEkle (String str){

                    HttpConnection hc=null;
                    OutputStream out=null;

          try {
                              hc=(HttpConnection)Connector.open(str);

                              hc.setRequestMethod(HttpConnection.GET);
                              hc.setRequestProperty("IF-Modified-Since", "20 Jan 2001 16:19:14
GMT");
                    hc.setRequestProperty("User-Agent","Profile/MIDP-1.0 Configuration/CLDC-
1.0");
                              hc.setRequestProperty("Content-Language", "en-CA");
                              out=hc.openOutputStream();
                 SprsTkp.HataMusteriEkle=0;
                              out.close();
                              hc.close();

                    }
                    catch(Exception e) {
              SprsTkp.HataMusteriEkle=1;
              System.out.println("Hata Connection da");
          }
}

public void gonder(){
   Thread t=new Thread() {
           public void run() {
         try {
            kayitEkle(SprsTkp.URL);
            }
         catch(Exception ef)            {
            SprsTkp.HataMusteriEkle=1;
            System.out.println("Hata Connection de");
```

```
        }
         }
   };
   t.start();
   while (t.isAlive()){}
}
}
```

MusteriListele.java

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
import javax.microedition.io.*;
import java.io.*;

class MusteriListele{
SiparisTakip SprsTkp;
public MusteriListele(SiparisTakip ST){
  SprsTkp=ST;
}

public void listeAl (){
                    HttpConnection c=null;
                    InputStream is=null;
                    OutputStream os=null;
                    StringBuffer b=new StringBuffer();
            StringBuffer yazi=new StringBuffer();

           try{
                    c=(HttpConnection)Connector.open(SprsTkp.URL);
                    c.setRequestMethod(HttpConnection.GET);
                    c.setRequestProperty("IF-Modified-Since", "20 Jan 2001 16:19:14 GMT");
             c.setRequestProperty("User-Agent","Profile/MIDP-1.0 Configuration/CLDC-1.0");
                    c.setRequestProperty("Content-Language", "en-CA");
                    os=c.openOutputStream();
                    is=c.openDataInputStream();

             int ch,sayac,sayac2,i,durum;
```

```
            sayac=1;sayac2=1;durum=1;
                    while((ch=is.read())!=-1) {
             if (ch==13){
                    if (durum==1){
                            SprsTkp.musteriIdDizi[sayac++]=yazi.toString();
                            durum=0;
                    }else{
                            SprsTkp.musteriAdiDizi[sayac2++]=yazi.toString();
                            durum=1;
                    }
                    yazi.delete(0,yazi.length());
                    is.read();
             }else{
                    yazi.append((char)ch);
                    b.append((char) ch);
             }
            }
            SprsTkp.HataMusteriListele=0;
            is.close();
            os.close();
            c.close();
            SprsTkp.musteriSayisi=sayac2;
           }
      catch(Exception exp)
      {
         SprsTkp.HataMusteriListele=1;
         System.out.println("Baglanti hatasi");
      }
}

public void listele(){
Thread t=new Thread() {
          public void run() {
                          try {
                  listeAl();
                  }
                catch(Exception e)
                  {
                    SprsTkp.HataMusteriListele=1;
```

```
                    }
                  }
                };
t.start();
while (t.isAlive()){
}
}

}
```

SiparisEkle.java

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
import javax.microedition.io.*;
import java.io.*;
import java.lang.*;

class SiparisEkle{
SiparisTakip SprsTkp;
int ch;
public SiparisEkle(SiparisTakip ST){
  SprsTkp=ST;
}

public void kayitlariEkle (){

                    HttpConnection hc=null;
                    OutputStream os=null;
                    InputStream is=null;
          try {
                        hc=(HttpConnection)Connector.open(SprsTkp.URL);

                        hc.setRequestMethod(HttpConnection.GET);
                        hc.setRequestProperty("IF-Modified-Since", "20 Jan 2001 16:19:14
GMT");
```

```
                    hc.setRequestProperty("User-Agent","Profile/MIDP-1.0 Configuration/CLDC-
1.0");
                                hc.setRequestProperty("Content-Language", "en-CA");
                                os=hc.openOutputStream();
                        is=hc.openDataInputStream();
                            ch=is.read();
                    SprsTkp.HataSiparisEkleme=0;
                    os.close();
                    is.close();
                            hc.close();
                            }
                            catch(Exception e) {
                        SprsTkp.HataSiparisEkleme=1;
                        System.out.println("Hata Connection da");
                    }
}

public void yolla(){
   Thread t=new Thread() {
            public void run() {
          try {
             kayitlariEkle();
          }
               catch(Exception e){
             SprsTkp.HataSiparisEkleme=1;
             System.out.println("Hata Connection de");
          }
           }
   };
   t.start();
   while (t.isAlive()){}
if(ch=='H'){
   SprsTkp.siparisEklendi=0;
}
if (ch=='E'){
   SprsTkp.siparisEklendi=1;
}
}
}
```

SiparisKontrol.java

```
import javax.microedition.rms.*;
import javax.microedition.io.*;
import java.io.*;

class SiparisKontrol{
   SiparisTakip SprsTkp;
public SiparisKontrol(SiparisTakip ST){
   SprsTkp=ST;
}

public void kontrol (){
                    HttpConnection c=null;
                    InputStream is=null;
                    OutputStream os=null;
            StringBuffer yazi=new StringBuffer();
           try{
                    c=(HttpConnection)Connector.open(SprsTkp.URL);
                    c.setRequestMethod(HttpConnection.GET);
                    c.setRequestProperty("IF-Modified-Since", "20 Jan 2001 16:19:14 GMT");
            c.setRequestProperty("User-Agent","Profile/MIDP-1.0 Configuration/CLDC-1.0");
                    c.setRequestProperty("Content-Language", "en-CA");
                    os=c.openOutputStream();
                    is=c.openDataInputStream();

             int ch;
                    while((ch=is.read())!=13) {
                    yazi.append((char)ch);
            }
            SprsTkp.siparisNo=Integer.parseInt(yazi.toString());
            yazi.delete(0,yazi.length());
            is.read();
            while((ch=is.read())!=13) {
                    yazi.append((char)ch);
            }
            SprsTkp.siparisTutari=yazi.toString();
            SprsTkp.HataSiparisKontrol=0;
```

```
            is.close();
            os.close();
            c.close();
           }
      catch(Exception exp)
      {
          SprsTkp.HataSiparisKontrol=1;
         System.out.println("Baglanti hatasi");
      }
}

public void siparisVarmi(){
   Thread t=new Thread() {
           public void run() {
         try {
            kontrol();
         }
         catch(Exception e){
             SprsTkp.HataSiparisKontrol=1;
            System.out.println("d?? hata");
         }
          }
   };
   t.start();
   while(t.isAlive()){}
}
}
```

SiparisListele.java

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
import javax.microedition.io.*;
import java.io.*;

class SiparisListele{
   SiparisTakip SprsTkp;
```

```
public SiparisListele(SiparisTakip ST){
   SprsTkp=ST;
}

public void listeAl (){
                    HttpConnection c=null;
                    InputStream is=null;
                    OutputStream os=null;
                    StringBuffer b=new StringBuffer();
            StringBuffer yazi=new StringBuffer();

           try{
                    c=(HttpConnection)Connector.open(SprsTkp.URL);
                    c.setRequestMethod(HttpConnection.GET);
                    c.setRequestProperty("IF-Modified-Since", "20 Jan 2001 16:19:14 GMT");
            c.setRequestProperty("User-Agent","Profile/MIDP-1.0 Configuration/CLDC-1.0");
                    c.setRequestProperty("Content-Language", "en-CA");
                    os=c.openOutputStream();
                    is=c.openDataInputStream();

            int ch,sayac0,sayac,sayac2,sayac3,sayac4,sayac5,i,durum;
            sayac=1;sayac2=1;durum=0;
            sayac3=1;sayac4=1;
            sayac5=1;
            sayac0=1;
                    while((ch=is.read())!=-1) {
             if (ch==13){
                      if (durum==0){
                              SprsTkp.SatisNoDizi[sayac0++]=yazi.toString();
                              durum=1; }
                      else if (durum==1){
                              SprsTkp.SatisUrunIdDizi[sayac++]=yazi.toString();
                              durum=2;
                      }else if (durum==2){
                              SprsTkp.SatisUrunDizi[sayac2++]=yazi.toString();
                              durum=3;
                      }else if (durum==3){
                              SprsTkp.SatisUrunFiyatiDizi[sayac3++]=yazi.toString();
                              durum=4;
```

```
                    }else if (durum==4){
                            SprsTkp.SatisUrunMiktariDizi[sayac4++]=yazi.toString();
                            durum=5;
                    }else if (durum==5){
                            SprsTkp.SatisUrunTutariDizi[sayac5++]=yazi.toString();
                            durum=0;
                    }
                    yazi.delete(0,yazi.length());
                    is.read();
              }else{
                    yazi.append((char)ch);
                    b.append((char) ch);
              }
             }
             is.close();
             os.close();
             c.close();
             SprsTkp.HataSiparisListele=0;
             SprsTkp.siparisSayisi=sayac2;
            }
      catch(Exception exp)
      {
         SprsTkp.HataSiparisListele=1;
         System.out.println("Baglanti hatasi");
      }
}

public void listele(){
Thread t=new Thread() {
          public void run() {
                        try {
                  listeAl();
                  }
                catch(Exception e)
                  {
                    SprsTkp.HataSiparisListele=1;
                  }
                 }
                };
```

```
t.start();
while (t.isAlive()){
}
}

}
```

SiparisSil.java

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
import javax.microedition.io.*;
import java.io.*;
import java.lang.*;

class SiparisSil{
   SiparisTakip SprsTkp;
public SiparisSil(SiparisTakip ST){
   SprsTkp=ST;
}
public void sprsSil (){

                    HttpConnection hc=null;
                    OutputStream out=null;

          try {
                              hc=(HttpConnection)Connector.open(SprsTkp.URL);



                              hc.setRequestMethod(HttpConnection.GET);
                              hc.setRequestProperty("IF-Modified-Since", "20 Jan 2001 16:19:14
GMT");
                    hc.setRequestProperty("User-Agent","Profile/MIDP-1.0 Configuration/CLDC-
1.0");
                              hc.setRequestProperty("Content-Language", "en-CA");
                              out=hc.openOutputStream();
                SprsTkp.HataSiparisSil=0;
                              out.close();
```

```
                    hc.close();
                }
                catch(Exception e) {
            SprsTkp.HataSiparisSil=1;
            System.out.println("Hata Connection da");
          }
}

public void yolla(){
   Thread t=new Thread() {
          public void run() {
        try {
           sprsSil();
        }
        catch(Exception ef)          {
           SprsTkp.HataSiparisSil=1;
           System.out.println("Hata Connection de");
        }
          }
   };
   t.start();
   while (t.isAlive()){}
}
}
```

UrunleriListele.java

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.rms.*;
import javax.microedition.io.*;
import java.io.*;

class UrunleriListele{
SiparisTakip SprsTkp;
public UrunleriListele(SiparisTakip ST){
  SprsTkp=ST;
}
```

```
public void listeAl (){

                HttpConnection c=null;
          InputStream is=null;
                OutputStream os=null;
                StringBuffer b=new StringBuffer();
          StringBuffer yazi=new StringBuffer();

         try {
          c=(HttpConnection)Connector.open(SprsTkp.URL);
                c.setRequestMethod(HttpConnection.GET);
                c.setRequestProperty("IF-Modified-Since", "20 Jan 2001 16:19:14 GMT");
          c.setRequestProperty("User-Agent","Profile/MIDP-1.0 Configuration/CLDC-1.0");
                c.setRequestProperty("Content-Language", "en-CA");
                os=c.openOutputStream();
                is=c.openDataInputStream();

          int ch,sayac,sayac2,sayac3,sayac4,sayac5,i,durum;
          sayac=1;sayac2=1;durum=1;
          sayac3=1;sayac4=1;sayac5=1;
                while((ch=is.read())!=-1) {
           if (ch==13){
                  if (durum==1){
                        SprsTkp.UrunIdDizi[sayac++]=yazi.toString();
                        durum=2;
                  }else if (durum==2){
                        SprsTkp.UrunAdiDizi[sayac2++]=yazi.toString();
                        durum=3;
                  }else if (durum==3){
                        SprsTkp.UrunBirimiDizi[sayac3++]=yazi.toString();
                        durum=4;
                  }else if(durum==4){
                        SprsTkp.UrunFiyatiDizi[sayac4++]=yazi.toString();
                        durum=5;
                  }else if (durum==5){
                        SprsTkp.UrunStokMiktariDizi[sayac5++]=yazi.toString();
                        durum=1;
                  }
                  yazi.delete(0,yazi.length());
```

```
                    is.read();
                }else{
                    yazi.append((char)ch);
                    b.append((char) ch);
                }
              }
              SprsTkp.urunSayisi=sayac2;
              SprsTkp.HataUrunler=0;
              is.close();
              os.close();
              c.close();
             }
        catch(Exception exp)
        {
          SprsTkp.HataUrunler=1;
          System.out.println("Baglanti hatasi");
        }
}

public void listele(){
   Thread t=new Thread() {
            public void run() {
          try {
             listeAl();
          }
              catch(Exception ef)       {
             SprsTkp.HataUrunler=1;
             System.out.println("Hata Connection de");
          }
            }
   };
   t.start();
   while (t.isAlive()){}
}
}
```

Ek – 3 Yönetim Konsolu Programının Kodları

Module1.bas

```
Attribute VB_Name = "Module1"
Global localCNN As New ADODB.Connection
Global localRST As New ADODB.Recordset
Global LocalRST2 As New ADODB.Recordset
Global LocalRST3 As New ADODB.Recordset
Global PersonelId As Integer
Global gname, gsurname As String
Global fis As Integer


 Public Function DBOpen() As Long
   Set localCNN = New ADODB.Connection
   Set localRST = New ADODB.Recordset
   Set LocalRST2 = New ADODB.Recordset
   Set LocalRST3 = New ADODB.Recordset

   dbopened = False
   On Error GoTo ErrorHandler1

    With localCNN
      .ConnectionTimeout = 30
      .CommandTimeout = 60
      .Mode = adModeReadWrite Or adModeShareDenyNone
      .CursorLocation = adUseClient
      .IsolationLevel = adXactIsolated
      .Open "DSN=veri;", , , adConnectUnspecified
   End With

   dbopened = True

   With localRST
      .CacheSize = 301
      .CursorType = adOpenForwardOnly
      .LockType = adLockReadOnly
      .ActiveConnection = localCNN
   End With

      With LocalRST2
      .CacheSize = 301
      .CursorType = adOpenForwardOnly
      .LockType = adLockReadOnly
      .ActiveConnection = localCNN
   End With

   With LocalRST3
      .CacheSize = 301
      .CursorType = adOpenForwardOnly
      .LockType = adLockReadOnly
      .ActiveConnection = localCNN
   End With
```

```
    Exit Function
ErrorHandler1:
    DBOpen = ErrorWorks(localCNN, "DBOpen")
    If Not (localCNN Is Nothing) Then
        If localCNN.State = adStateOpen Then
            localCNN.Close
        End If
    End If
    dbopened = False
    On Error GoTo 0

    Exit Function

End Function

Public Sub DBClose()
On Error Resume Next

If localRST.State = adStateOpen Then
    localRST.Close
End If

If LocalRST2.State = adStateOpen Then
    LocalRST2.Close
End If

If LocalRST3.State = adStateOpen Then
    LocalRST3.Close
End If

If localCNN.State = adStateOpen Then
    localCNN.Close
End If

Set localRST = Nothing
Set LocalRST2 = Nothing
Set LocalRST3 = Nothing
Set localCNN = Nothing

On Error GoTo 0

dbopened = False


End Sub

Private Function ErrorWorks(errObj As ADODB.Connection, procname As String, Optional bno
As Byte = 255) As Long

   Dim objError As ADODB.Error

    If errObj Is Nothing Then
        Call HataYaz(CInt(bno), procname & " " & Err.Number & ": " & "[" & Err.Source & "]" &
Err.Description)
        ErrorWorks = Err.Number
    ElseIf errObj.Errors.Count > 0 Then
     For Each objError In errObj.Errors
        With objError
```

```
        Call HataYaz(CInt(bno), procname & " " & Err.Number & ": " & "[" & Err.Source & "]" &
Err.Description)
        ErrorWorks = .Number
      End With
   Next objError
   Else
      Call HataYaz(CInt(bno), procname & " " & Err.Number & ": " & "[" & Err.Source & "]" &
Err.Description)
      ErrorWorks = Err.Number
   End If


End Function
Public Sub HataYaz(a As String, b As String)
MsgBox ("ODBC Ayarlarından veri isimli bir veri kaynağı oluşturun" & vbCrLf & "Hata için bu
klasordeki hata.txt dosyasinin sonuna bakiniz")
Open "hata.txt" For Append As #11
 Print #11, a, b, " dbopened"
 Close #11
End Sub

Public Sub vtkapat()
  If localRST.State = adStateOpen Then
      localRST.Close
  End If
End Sub

Public Sub vtkapat2()
  If LocalRST2.State = adStateOpen Then
      LocalRST2.Close
  End If
End Sub
Public Sub vtkapat3()
  If LocalRST3.State = adStateOpen Then
      LocalRST3.Close
  End If
End Sub
```

Form1.frm

```
Dim sira As String
Dim urunsira As String
Dim musteri_id(200) As String
Dim siparis_no(200) As String
Dim siparis_tutar(200) As String
Dim user_id(200) As String
Dim user_password(200) As String
Dim gun, ay, yil As Integer
Dim tarih As String

Private Sub clear_Click()
temizle
Text1.SetFocus
update.Enabled = False
delete.Enabled = False
save.Enabled = True
clear.Enabled = True
```

```
End Sub

Private Sub clear2_Click()
Text7 = ""
Text8 = ""
Text9 = ""
Text10 = ""
update2.Enabled = False
delete2.Enabled = False
save2.Enabled = True
clear2.Enabled = True
Text7.SetFocus
End Sub

Private Sub clear3_Click()
Text11 = ""
Text12 = ""
update3.Enabled = False
delete3.Enabled = False
save3.Enabled = True
clear3.Enabled = True
End Sub

Private Sub Combo1_KeyPress(KeyAscii As Integer)
If KeyAscii > 1 And KeyAscii < 128 Then
KeyAscii = 0
End If
End Sub

Private Sub Combo2_KeyPress(KeyAscii As Integer)
If KeyAscii > 1 And KeyAscii < 128 Then
KeyAscii = 0
End If
End Sub

Private Sub Combo3_KeyPress(KeyAscii As Integer)
If KeyAscii > 1 And KeyAscii < 128 Then
KeyAscii = 0
End If
End Sub

Private Sub Combo4_KeyPress(KeyAscii As Integer)
If KeyAscii > 1 And KeyAscii < 128 Then
KeyAscii = 0
End If
End Sub

Private Sub Combo5_KeyPress(KeyAscii As Integer)
If KeyAscii > 1 And KeyAscii < 128 Then
KeyAscii = 0
End If
End Sub

Private Sub Command1_Click()
Frame5.Visible = False
Frame4.Visible = False
Frame1.Visible = True
Frame2.Visible = False
```

```
Text1.SetFocus
update.Enabled = False
delete.Enabled = False
save.Enabled = True
clear.Enabled = True
musteriListele
plasiyerEkle
temizle
End Sub

Private Sub temizle()
Text1 = ""
Text2 = ""
Text3 = ""
Text4 = ""
Text5 = ""
Text6 = ""
Combo1 = ""
Combo2 = ""
End Sub
Private Sub plasiyerEkle()
vtkapat3
LocalRST3.Open ("select * from user order by user_name")
Combo1.clear
Do While Not (LocalRST3.EOF)
   Combo1.AddItem (LocalRST3("user_name").Value)
   LocalRST3.MoveNext
Loop
End Sub

Private Sub musteriListele()
vtkapat
localRST.Open ("select musteri_id,firma,musteri,telefon from musteri order by " + sira)
Set DataGrid1.DataSource = localRST
End Sub

Private Sub Command10_Click()
DataGrid2.Visible = True
If Combo4 <> "" Then
vtkapat
localRST.Open ("select * from satis where siparis_no=" & siparis_no(Combo4.ListIndex) & "")
Set DataGrid2.DataSource = localRST
End If
Label11.Caption = "TOPLAM TUTAR " + siparis_tutar(Combo4.ListIndex) + " YTL"
End Sub

Private Sub bugun()
gun = Day(Date)
ay = Month(Date)
yil = Year(Date)
tarih = Trim(Str(gun)) + "." + Trim(Str(ay)) + "." + Trim(Str(yil))
End Sub


Private Sub Command11_Click()
urunsira = "urun_id"
urunlerListele
End Sub
```

```
Private Sub Command12_Click()
urunsira = "urun_adi"
urunlerListele
End Sub

Private Sub Command13_Click()
urunsira = "birimi"
urunlerListele
End Sub

Private Sub Command14_Click()
urunsira = "fiyati"
urunlerListele
End Sub

Private Sub Command15_Click()
urunsira = "stok"
urunlerListele
End Sub

Private Sub Command16_Click()
If Not localRST.EOF Then
   Text7 = localRST("urun_adi").Value
   Text8 = localRST("birimi").Value
   Text9 = localRST("fiyati").Value
   Text10 = localRST("stok").Value
End If
update2.Enabled = True
delete2.Enabled = True
clear2.Enabled = True
save2.Enabled = False
End Sub

Private Sub Command17_Click()
Form1.Enabled = False
Form2.Text1 = ""
Form2.Text7 = localRST("urun_adi").Value
Form2.Text8 = localRST("birimi").Value
Form2.Text9 = localRST("fiyati").Value
Form2.Text10 = localRST("stok").Value
Form2.Show
End Sub

Private Sub Command18_Click()

DTPicker1.Value = Now
update3.Enabled = False
delete3.Enabled = False
save3.Enabled = True
clear3.Enabled = True
Frame1.Visible = False
Frame2.Visible = False
Frame4.Visible = False
Frame5.Visible = True
clear3_Click
plasiyerlistele
End Sub
```

```
Private Sub plasiyerlistele()
vtkapat
localRST.Open ("select * from user order by user_name")
i = 0
Combo5.clear
Do While Not (localRST.EOF)
   Combo5.AddItem (localRST("user_name").Value)
   user_id(i) = (localRST("user_id").Value)
   user_password(i) = (localRST("user_password").Value)
   i = i + 1
   localRST.MoveNext
Loop
End Sub

Private Sub Command19_Click()
If Combo5.ListIndex <> -1 Then
   Text11 = Combo5.List(Combo5.ListIndex)
   Text12 = user_password(Combo5.ListIndex)
   update3.Enabled = True
   delete3.Enabled = True
   save3.Enabled = False
   clear3.Enabled = True
End If
End Sub

Private Sub Command2_Click()
Frame5.Visible = False
Frame4.Visible = False
Frame3.Visible = False
Frame2.Visible = True
Frame1.Visible = False
DataGrid2.Visible = False
Combo3musteri
End Sub

Private Sub Combo3musteri()
vtkapat
localRST.Open ("select * from musteri order by firma")
vtkapat2
LocalRST2.Open ("select count(*) as firmaSayisi from musteri")
i = 0
Combo3.clear
Do While Not (localRST.EOF)
   Combo3.AddItem (localRST("firma").Value)
   musteri_id(i) = (localRST("musteri_id").Value)
   i = i + 1
   localRST.MoveNext
Loop

End Sub

Private Sub Command20_Click()
If Combo5.ListIndex = -1 Then
   MsgBox ("Lütfen Plasiyer Seçin")
Else
   Dim gun1, ay1, yil1 As Integer
   Dim tarih1 As String
   gun1 = DTPicker1.Day
```

```
   ay1 = DTPicker1.Month
   yil1 = DTPicker1.Year
   tarih1 = Trim(Str(gun1)) + "." + Trim(Str(ay1)) + "." + Trim(Str(yil1))
   vtkapat
   localRST.Open ("select siparis_no, firma,tutar from siparis,musteri where user_id=" &
user_id(Combo5.ListIndex) & " and tarih ='" & tarih1 & "' and musteri_id=firma_id ")
   Set DataGrid4.DataSource = localRST
End If
End Sub

Private Sub Command24_Click()
Frame1.Visible = False
Frame2.Visible = False
Frame3.Visible = False
Frame4.Visible = False
Frame5.Visible = False
Form1.Enabled = False
Form3.DTPicker1.Value = Date
Form3.Show
End Sub

Private Sub Command25_Click()
If Not (localRST.EOF) Then
   vtkapat2
   LocalRST2.Open ("select satis_no,urun_adi,urun_fiyati,urun_miktari,urun_tutari from satis
where siparis_no =" & localRST("siparis_no").Value & " ")
   Set DataGrid5.DataSource = LocalRST2
End If
End Sub

Private Sub Command3_Click()
update2.Enabled = False
delete2.Enabled = False
save2.Enabled = True
clear2.Enabled = True
Frame1.Visible = False
Frame4.Visible = True
Frame5.Visible = False
Frame2.Visible = False
Text7.SetFocus
clear2_Click
urunlerListele
End Sub

Private Sub urunlerListele()
vtkapat
localRST.Open ("Select * from urunler order by " + urunsira)
Set DataGrid3.DataSource = localRST
End Sub

Private Sub Command31_Click()
End
End Sub

Private Sub Command4_Click()
sira = "musteri_id"
musteriListele
End Sub
```

```
Private Sub Command5_Click()
sira = "firma"
musteriListele
End Sub

Private Sub Command6_Click()
sira = "musteri"
musteriListele
End Sub

Private Sub Command7_Click()
sira = "telefon"
musteriListele
End Sub

Private Sub Command8_Click()
update.Enabled = True
delete.Enabled = True
save.Enabled = False
clear.Enabled = True
vtkapat2
LocalRST2.Open ("select * from musteri where musteri_id=" +
Str(localRST("musteri_id").Value))
If Not (LocalRST2.EOF) Then
   Text1 = LocalRST2("Firma").Value
   Text2 = LocalRST2("musteri").Value
   Text4 = LocalRST2("telefon").Value
   Text3 = LocalRST2("adres").Value
   Text5 = LocalRST2("vergi_dairesi").Value
   Text6 = LocalRST2("vergi_numarasi").Value
   vtkapat3
   LocalRST3.Open ("select * from user where user_id=" + LocalRST2("kullanici").Value)
   Combo1.Text = LocalRST3("user_name").Value
   Select Case Int(LocalRST2("gun").Value)
      Case 1
         Combo2.Text = "PAZARTESİ"
      Case 2
         Combo2.Text = "SALI"
      Case 3
         Combo2.Text = "ÇARŞAMBA"
      Case 4
         Combo2.Text = "PERŞEMBE"
      Case 5
         Combo2.Text = "CUMA"
      Case 6
         Combo2.Text = "CUMARTESİ"
      Case 7
         Combo2.Text = "PAZAR"
   End Select
End If
End Sub

Private Sub delete_Click()
If MsgBox("Bu Müşteriyi gerçekten silmek istiyormusunuz", vbYesNo) = vbYes Then
   vtkapat2
   LocalRST2.Open ("delete from musteri where musteri_id=" & localRST("musteri_id").Value &
" ")
   Command1_Click
```

```
   MsgBox ("Müşteri Silindi")
   clear_Click
End If
End Sub

Private Sub delete2_Click()
If MsgBox("Bu ürünü gerçekten silmek istiyormusunuz", vbYesNo) = vbYes Then
   vtkapat2
   LocalRST2.Open ("delete from urunler where urun_id=" & localRST("urun_id").Value & " ")
   Command3_Click
   MsgBox ("Ürün Silindi")
End If
End Sub

Private Sub delete3_Click()
If MsgBox("Bu plasiyeri gerçekten silmek istiyormusunuz", vbYesNo) = vbYes Then
   vtkapat2
   LocalRST2.Open ("delete from user where user_id=" & user_id(Combo5.ListIndex) & " ")
   Command18_Click
   MsgBox ("Plasiyer Silindi")
End If
End Sub

Private Sub Form_Load()
sira = "musteri_id"
urunsira = "urun_adi"
If DBOpen <> 0 Then
   MsgBox ("Veri tabani acilamadi")
End If

End Sub

Private Sub Form_Unload(Cancel As Integer)
DBClose
End Sub

Private Sub save_Click()
If (Len(Text1) * Len(Text2) * Len(Text3) * Len(Text4) * Len(Text5) * Len(Text6) *
Len(Combo1) * Len(Combo2) = 0) Then
   MsgBox ("TÜM BİLGİLERİ EKSİKSİZ GİRİNİZ")
Else
   vtkapat3
   LocalRST3.Open ("select * from user where user_name='" & Combo1 & "'")
   localCNN.Execute ("insert into musteri
(firma,musteri,telefon,adres,vergi_dairesi,vergi_numarasi,kullanici,gun) Values('" & Text1 & "','"
& Text2 & "','" & Text4 & "','" & Text3 & "','" & Text5 & "','" & Text6 & "','" &
LocalRST3("user_id").Value & "','" & Combo2.ListIndex + 1 & "')")
   Command1_Click
   MsgBox ("KAYDEDİLDİ")
End If
End Sub

Private Sub save2_Click()
If (Len(Text7) * Len(Text8) * Len(Text9) * Len(Text10) = 0) Then
   MsgBox ("TÜM BİLGİLERİ EKSİKSİZ GİRİNİZ")
Else
   localCNN.Execute ("insert into urunler (urun_adi,birimi,fiyati,stok) Values('" & Text7 & "','" &
Text8 & "','" & Text9 & "','" & Text10 & "')")
```

```
   Command3_Click
   MsgBox (" ÜRÜN KAYDEDİLDİ")
End If
End Sub

Private Sub save3_Click()
If (Len(Text11) * Len(Text12) = 0) Then
   MsgBox ("TÜM BİLGİLERİ EKSİKSİZ GİRİNİZ")
Else
   localCNN.Execute ("insert into user (user_name,user_password) Values('" & Text11 & "','" &
Text12 & "')")
   Command18_Click
   MsgBox ("PLASİYER KAYDEDİLDİ")
End If
End Sub

Private Sub Text10_KeyPress(KeyAscii As Integer)
If (KeyAscii < 48 Or KeyAscii > 57) And Not KeyAscii = 8 Then
   KeyAscii = 0
End If
End Sub

Private Sub Text12_KeyPress(KeyAscii As Integer)
If (KeyAscii < 48 Or KeyAscii > 57) And Not KeyAscii = 8 Then
   KeyAscii = 0
End Sub

Private Sub Text9_KeyPress(KeyAscii As Integer)
If (KeyAscii < 48 Or KeyAscii > 57) And Not KeyAscii = 46 And Not KeyAscii = 8 Then
   KeyAscii = 0
End If

End Sub

Private Sub update_Click()
If (Len(Text1) * Len(Text2) * Len(Text3) * Len(Text4) * Len(Text5) * Len(Text6) *
Len(Combo1) * Len(Combo2) = 0) Then
   MsgBox ("TÜM BİLGİLERİ EKSİKSİZ GİRİNİZ")
Else
   vtkapat3
   LocalRST3.Open ("select * from user where user_name='" & Combo1 & "'")
   If Combo2.Text = "PAZARTESİ" Then gun = "1"
   If Combo2.Text = "SALI" Then gun = "2"
   If Combo2.Text = "ÇARŞAMBA" Then gun = "3"
   If Combo2.Text = "PERŞEMBE" Then gun = "4"
   If Combo2.Text = "CUMA" Then gun = "5"
   If Combo2.Text = "CUMARTESİ" Then gun = "6"
   If Combo2.Text = "PAZAR" Then gun = "7"
   localCNN.Execute ("update  musteri set firma='" & Text1 & "',musteri='" & Text2 &
"',telefon='" & Text4 & "',Adres='" & Text3 & "', vergi_dairesi='" & Text5 & "',vergi_numarasi='"
& Text6 & "', kullanici='" & LocalRST3("user_id").Value & "', gun='" & gun & "' where
musteri_id= " & localRST("musteri_id").Value & " ")
   Command1_Click
   MsgBox ("MÜŞTERİ BİLGİLERİ GÜNCELLENDİ")
End If
End Sub

Private Sub Command9_Click()
```

```
If Combo3.Text <> "" Then
      Frame3.Visible = True
      vtkapat
      localRST.Open ("select * from siparis where firma_id=" & musteri_id(Combo3.ListIndex) &
" order by siparis_no")
      Combo4.clear
      i = 0
      Do While Not (localRST.EOF)
         Combo4.AddItem (localRST("tarih").Value)
         siparis_no(i) = (localRST("siparis_no").Value)
         siparis_tutar(i) = (localRST("tutar").Value)
         i = i + 1
         localRST.MoveNext
      Loop
End If
End Sub

Private Sub update2_Click()
If (Len(Text7) * Len(Text8) * Len(Text9) * Len(Text10) = 0) Then
   MsgBox ("TÜM BİLGİLERİ EKSİKSİZ GİRİNİZ")
Else
   localCNN.Execute ("update  urunler set urun_adi='" & Text7 & "',birimi='" & Text8 &
"',fiyati='" & Text9 & "',stok='" & Text10 & "' where urun_id= " & localRST("urun_id").Value &
" ")
   Command3_Click
   MsgBox ("ÜRÜN BİLGİLERİ GÜNCELLENDİ")
End If
End Sub

Private Sub update3_Click()
If (Len(Text11) * Len(Text12) = 0) Then
   MsgBox ("TÜM BİLGİLERİ EKSİKSİZ GİRİNİZ")
Else
   localCNN.Execute ("update user set user_name='" & Text11 & "',user_password='" & Text12 &
"'where user_id= " & user_id(Combo5.ListIndex) & " ")
   Command18_Click
   MsgBox ("PLASİYER BİLGİLERİ GÜNCELLENDİ")
End If
End Sub
```

Form2.frm

```
Private Sub Command1_Click()
Dim toplam As String
If (Len(Text1) = 0) Then
   MsgBox ("LÜTFEN GELEN MİKTARI GİRİNİZ")
Else
   toplam = Str(Int(Text10) + Int(Text1))
   localCNN.Execute ("update  urunler set urun_adi='" & Text7 & "',birimi='" & Text8 &
"',fiyati='" & Text9 & "',stok='" & toplam & "' where urun_id= " & localRST("urun_id").Value &
" ")
   MsgBox ("ÜRÜN BİLGİLERİ GÜNCELLENDİ")
   Unload Me
End If
End Sub
```

```
Private Sub Command2_Click()
MsgBox (Form2.Left + " " + Form2.Top)
Unload Me
End Sub

Private Sub Form_Unload(Cancel As Integer)
Form1.Enabled = True
vtkapat
localRST.Open ("Select * from urunler order by urun_adi")
Set Form1.DataGrid3.DataSource = localRST
Form1.Text7 = ""
Form1.Text8 = ""
Form1.Text9 = ""
Form1.Text10 = ""
Form1.update2.Enabled = False
Form1.delete2.Enabled = False
Form1.save2.Enabled = True
Form1.clear2.Enabled = True
End Sub

Private Sub Text1_KeyPress(KeyAscii As Integer)
If (KeyAscii < 48 Or KeyAscii > 57) And Not KeyAscii = 8 Then
   KeyAscii = 0
End If
End Sub
```

Form3.frm

```
Private Sub Command1_Click()
   Dim gun1, ay1, yil1 As Integer
   Dim tarih1 As String
   gun1 = DTPicker1.Day
   ay1 = DTPicker1.Month
   yil1 = DTPicker1.Year
   tarih1 = Trim(Str(gun1)) + "." + Trim(Str(ay1)) + "." + Trim(Str(yil1))
   vtkapat
   localRST.Open ("select siparis_no, firma,tutar from siparis,musteri where  tarih ='" & tarih1 &
"' and musteri_id=firma_id ")
   Set DataGrid1.DataSource = localRST
End Sub

Private Sub Command2_Click()
If (localRST.State = adStateOpen) Then
If Not (localRST.EOF) Then
   vtkapat2
   LocalRST2.Open ("select satis_no,urun_adi,urun_fiyati,urun_miktari,urun_tutari from satis
where siparis_no =" & localRST("siparis_no").Value & " ")
   Set DataGrid2.DataSource = LocalRST2
Else
   vtkapat2
   Set DataGrid2.DataSource = LocalRST2
End If
End If
End Sub

Private Sub Command3_Click()
If (LocalRST2.State = adStateOpen) Then
```

```
   If Not LocalRST2.EOF Then
Dim firma As Integer
Dim i As Integer
vtkapat3
LocalRST3.Open ("select * from siparis where siparis_no=" & localRST("siparis_no").Value &
"")
firma = LocalRST3("firma_id").Value
vtkapat3
LocalRST3.Open ("select * from musteri where musteri_id=" & firma & "")

With Printer
.FontSize = 12
.FontBold = False
X1 = (.ScaleWidth - .TextWidth("FATURA")) / 2
.CurrentX = X1
Printer.Print " FATURA "
Printer.Print "Firma Adı", ": "; LocalRST3("firma").Value
Printer.Print "Müşteri Adı", ": "; LocalRST3("musteri").Value
Printer.Print "Telefon", ":"; LocalRST3("telefon").Value
Printer.Print "Adres", ": "; LocalRST3("adres").Value
Printer.Print "Vergi Dairesi", ": "; LocalRST3("vergi_dairesi").Value
Printer.Print "Vergi Nn", ": "; LocalRST3("vergi_numarasi").Value
Printer.Print "Tarih/Saat", ": "; Str(Now)
Printer.Print " "
Printer.FontUnderline = True
Printer.Print "Sıra No          Ürün Adı                              Ürün Miktari      Ürün Fiyatı
Tutar "
Printer.FontUnderline = False
i = 1
vtkapat3
LocalRST3.Open ("select satis_no,urun_adi,urun_fiyati,urun_miktari,urun_tutari from satis where
siparis_no =" & localRST("siparis_no").Value & " ")
Do While (Not LocalRST3.EOF)
   Printer.Print i, LocalRST3("urun_adi").Value, , LocalRST3("urun_miktari").Value,
LocalRST3("urun_fiyati").Value, LocalRST3("urun_tutari").Value
   i = i + 1
   LocalRST3.MoveNext
Loop
Printer.FontStrikethru = True
Printer.FontUnderline = True
Printer.Print "
"
Printer.FontStrikethru = False
Printer.FontUnderline = False
Printer.Print "                                                                    TOPLAM :",
localRST("tutar").Value
End With
Printer.EndDoc
End If
End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
vtkapat
vtkapat2
vtkapat3
Form1.Enabled = True
End Sub
```