

# Java ME (Java Micro Edition)

## Java ME Yapısı

Java ME (Eski adıyla J2ME) yani Java Micro Edition, Java'nın belirli kütüphanelerinin alınması ve bulunduğu ortama göre ek kütüphanelerin eklenmesi ile oluşturulmuş bir ortamdır. Bu sayede farklı yapılara ve güce sahip cihazlar arasında değişik Java versiyonları kullanılabilir. Bu ortam cihazlar arasında da belirli konfigürasyonlara göre farklılık gösterebilir. Örnek olarak bir PDA üzerindeki JAVA ME ile bir cep telefonundaki JAVA ME özellikleri aynı olmayabilir. Bunlara konfigürasyonlar diyoruz.

JAVA ME içerisinde cep telefonu üzerinde gerekli bir çok kütüphane mevcuttur sadece belirli ek özellikleri kullanamayacağımız kısıtlamalar yapılmıştır. Bunun nedeni zaten kısıtlı belleğe sahip cihazların Java'nın standart versiyonunu destekleyecek kapasiteye sahip olmamasıdır. JAVA ME'nin çıkış amacı da budur.

Standart Java bilgisine sahip olan bir yazılımcının JAVA ME öğrenmesi zor olmayacaktır. Basit bir kaç kütüphaneye sahip bu platformda Nokia, Sony-Ericsson gibi şirketler olaya esneklik ve kolaylık katacak bir çok ek kütüphane geliştirmişlerdir. Bu kütüphanelere üreticilerin developer sayfalarından ulaşabilirsiniz. Bunların popülerleri arasında aşağıdaki siteler sayılabilir.

[http://developer.sonyericsson.com/site/global/home/p\\_home.jsp](http://developer.sonyericsson.com/site/global/home/p_home.jsp)

<http://forum.nokia.com>

<http://www.benqmobile.com/developer>

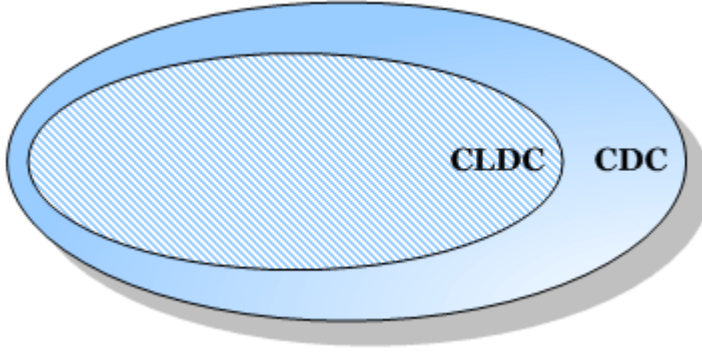
<http://developer.motorola.com>

## Konfigürasyonlar

Cihazlar arasında bağlantı, bellek ve işlemci kapasiteleri konusunda farklılıklar vardır. Bu ufak cihazlar arasında daha büyük farklar demektir. Örnek olarak bir akıllı kart (Smart Card) çok düşük bir konfigürasyona sahipken PDA'lar bunun binlerce katı belleğe sahip olabilirler.

Mobil aygıtlarda iki tip konfigürasyon vardır. Bunlar CLDC (Connected Limited Device Configuration) ve CDC (Connected Device Configuration). CLDC belli bir bağlantı yeteneğine sahip bellek ve işlemci kapasitesi daha düşük cihazlardır. Cep telefonları bunu sınıfa girer. CDC ise daha yüksek bağlantı kapasitesine sahip işlemci ve bellek durumu daha yüksek cihazlardır.

Günümüz JAVA ME dünyasında popüler olan konu CLDC yani cep telefonlarının bulunduğu konfigürasyondur. Sonuç olarak bir kaç sene içerisinde 1 milyar gibi bir rakama ulaşacak ve bu pazar yazılım şirketlerinin iştahını kabartıyor. Kitap üzerinde genel olarak CLDC üzerinde çalışacağız.



Yukarıdaki resimde görüldüğü gibi CDC daha kapsamlı kütüphane mimarisine sahiptir. Ancak CLDC içerisinde CDC den bağımsız kütüphanelerde bulunmaktadır bunları ileriki adımlarda göreceğiz.

### **CDC (Connected Device Configuration)**

Yüksek bellekli normal pc mimarisine yakın cihazlardır. Bu cihazlar üzerinde uygulama geliştirme imkanı daha genişleyebilir olup bellek kullanımı daha yüksektir. Bu yüzden standart Java' ya yakın bir kütüphane zenginliği vardır. Şu anda palm tarzı cihazlarda kullanılan bu mimari cep telefonu gibi cihazların daha da gelişmesiyle CLDC' nin yerini alacaktır. Bunun dışında pocket pc yada terminal tarzı cihazlarda da bu tarz bir yapı kullanılabilir.

CDC yapısının kullanılacağı mimarilerde JVM (Java Virtual Machine – Java Sanal Makinesi) yerine CVM yani Compact Virtual Machine kullanılır. Bu sanal makineyi yüklediğimiz her ortamda CDC tabanlı J2ME uygulamaları geliştirebiliriz. Buna örnek olarak son zamanlarda popüler olan smartphone' ların (Akıllı Telefonlar) gelişmiş hallerini gösterebiliriz.

CDC üzerinde kullanılan kütüphanelerin bazıları aşağıdaki gibidir.

```
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
```

### **CLDC (Connected Limited Device Configuration)**

Bu kitapta daha çok işleyeceğimiz mimari CLDC yani Connected Limited Device Configuration' dır. Şu anın popüler konusu olan cep telefonu uygulamalarında kullanılan konfigürasyon genellikle budur. Ancak CDC konusunda da belirttiğimiz gibi cep telefonlarının kapasiteleri git gide artıyor ve gelecek bir kaç sene içerisinde çok

daha iyi seviyelere gelecek bu durumda bellek ve bağlantı sorunları da yaşanmayacaktır ve artık uygulamalarımıza zenginlik katmak için bu tarz yüksek konfigirasyonlara ihtiyaç duyacağımız kesindir. Zaten şu anda Nokia ve Sony-Ericsson tarafında CDC konfigirasyonuna sahip cihazlar üretilmeye başladı zamanla bunlar artacak ve yerlerini artık smartphone tarzı cihazlara bırakacaktır.

CLDC konfigirasyonunda Java' nın temel kütüphanelerinden çok azı alınmıştır. Başta belirttiğimiz gibi bunlar sınırlı kapasite sahip cihazlar için JVM den özel olarak alınmış kütüphanelerdir. Bunlardan bazıları aşağıdaki gibidir.

```
java.io  
java.lang  
java.util
```

Ancak buradaki kütüphanelerde JVM de bulunan tüm sınıflar bulunmamaktadır. Örnek olarak standart Java da java.util paketi içerisinde Currency adında bir sınıf varken CLDC içerisinde böyle bir sınıf bulunmamaktadır. Bunun gibi onlarca sınıf CLDC java.util paketinde yok. Bunlar dışında microedition' la gelen bazı kütüphanelerde mevcut. Bunlardan bazıları javax.microedition.io yada javax.microedition gibi kütüphanelerdir. Bu kütüphaneler standart Java da bulunmayan ancak J2ME ortamında gerekli olan sınıfları içerir. Örnek olarak javax.microedition.io standart Java' da olan fakat J2ME ortamında kullanmadığımız bazı sınıfları içerir. CLDC adınında anlaşılacağı gibi limitli bir bağlantıya sahip bir konfigirasyon buna uygun olarak düzenlenmiş bazı giriş çıkış sınıfları düzenlenmiştir. Buna daha bir çok örnek verebiliriz ileriki bölümlerde bunların neden böyle kullanıldığını daha iyi anlamış olacağız.

### **MIDP (Mobile Information Device Profile)**

CLDC kendi içerisinde profile ' lara ayrılır. Bunlar birinci nesil MIDP 1.0 ve daha sonradan çıkarılmış biraz daha gelişmiş versiyonu olan MIDP 2.0 dir. MIDP 2.0, 1.0 versiyonuna göre bir çok gelişmiş özelliğe sahiptir. Hala 1.0 destekli bir çok telefon satılmaktadır bu yüzden J2ME destekli telefon alırken 2.0 destekli olmasına dikkat etmenizi öneririm.

Peki MIDP nedir ? Tam olarak açılımı Mobile Information Device Profile yani mobil cihazınızın versiyonudur diyebiliriz. Bu versiyon ile her üretici kendi mimarisine uygun birer profile seçiyor ve bunun için gerekli ortamı sağlıyor. Siz uygulamanızı geliştirirken cihazın bulunduğu ortama uygun profile' a göre geliştirme yapıyorsunuz. Uygulama yüklenirken ilgili ortama bakıyor ve tüm konfigirasyonlar ve profile' lar uygunsa uygulamanızın sistem üzerinde kurulumunu tamamlıyor.

Peki MIDlet ne demektir ? Aslında MIDlet J2ME ortamında çalıştırılabilir her bir uygulamaya verilen isimdir. Bu isim her MIDlet' in bir MIDlet sınıfından türemesi dolayısıyla birer MIDlet sınıfı olmasından kaynaklanıyor. İleriki konularda uygulamalarımızdan MIDlet olarak bahsedeceğiz gerçek anlamda bu isme alışsak iyi olur.

MIDlet 1.0 versiyonunda bulunan kütüphaneler aşağıdaki gibidir.

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

MIDlet 2.0 da ise durum aşağıdaki gibi

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

Göründüğü gibi 2.0 versiyonunda kütüphane sayısında bazı değişiklikler gelmiştir. Ancak bu versiyon geçişi sadece yeni gelen kütüphaneler le değil var olan kütüphaneler içerisinde de değişikliğe neden olmuştur. MIDlet 2.0 da gelen en büyük yenilikler media ve game tarafında olmuştur. Bunlar özellikle oyun yazımı ve cihazların media aygıtlarının kontrolü konusunda bir çok yenilik ve kolaylık getirmiştir. Örnek olarak javax.microedition.lcdui.game içerisinde oyun yazmamızı kolaylaştıran bir çok kütüphane bulunuyor.

Uygulamamızda kullanacağımız NetBeans gibi geliştirme ortamları sayesinde bu tarz konfigürasyon ve versiyon farklılıklarından kaynaklanan sınıf değişimlerini rahatlıkla görebiliriz.

MIDlet 2.0 ile gelen bir çok yenilikle birlikte örnek olarak media aygıtları yönetimi gibi şeylerin kolaylaştığını söylemiştik. Peki MIDlet 2.0 dan önce yada 1.0 destekli cihazlarda bu tarz kamera yada ses kontrolü gibi işlemler yapılamıyormuydu ? Aslında yapılabiliyordu ancak her üretici hatta her üretici her serisi yada modeli için ayrı kütüphaneler üretmek ve bunları yayınlamak zorundaydı. İşte bu durum bizim platform bağımsızlığımıza engel teşkil edecek bir ortam yaratıyordu. Eğer kamera kontrolü yapacağımız bir uygulamamız varsa farklı telefonlar için versiyonlar çıkarmak zorunda kalıyorduk. Ancak MIDlet 2.0 tüm bunlar için ortak bir kütüphane yaratıp üreticilerin bu standarta uymalarını sağlamıştır.

Tüm bunlar dışında ekstra kütüphaneler için Nokia, Sony-Ericsson, Motorola ve Siemens gibi üreticilerin sitelerinde ilginç örnekler bulabilirsiniz.

### **JSR (Java Specification Requests)**

Tüm bu versiyonların ve standartların dışında versiyonların ayı olmasına karşın bazı farklı kütüphane destekleri vardır. Örnek olarak bazı telefonlar bluetooth desteklerken

## Java Micro Edition – Java ME

---

bazıları aynı konfigürasyona sahip olmasına karşın böyle bir özelliğe sahip olamayabilirler. İşte bu tarz durumlarda cihazın belirli JSR lara sahip olması gerekliliği yatar.

Peki JSR yani Java Specification Requests nedir ? JSR Java' yı destekleyen belirli gruplarca oluşturulmuş Java' nın ortaklaşa, ortak kararlarla geliştirilebilmesi için oluşturulmuş bir oluşumdur. Bu oluşumun adı JCP yani Java Community Process' tir. JCP içerisinde Sun, IBM, Nokia, Ericsson, Siemens, HP gibi bir çok güçlü grup vardır ve üretilecek olan ürünlerde ihtiyaçlara göre standartları belirlerler. Örnek olarak JSR-82 bluetooth paketi olarak çıkarılmıştır. Bluetooth desteği verecek her üretilen pakete uymak zorundadır. Bunlar dışında geliştirilen ve çıkarılmaya hazırlanan bir çok JSR standardı bulunmaktadır.

Örnek olarak JSR-30 standardında yani CLDC üzerinde çalışan bazı şirketler.

3COM

Bull

Ericsson

Matsushita

Mitsubishi Electric

Motorola

Nokia

NTT DoCoMo

Siemens

Bunlar dışında çıkarılan bazı JSR ların listesi aşağıdaki gibidir.

<b>JSR Name</b>	<b>Package Name</b>
1 Realtime	javax.realtime
30 CLDC 1.0	javax.microedition.io
36 CDC 1.0	javax.microedition.io
37 MIDP 1.0	javax.microedition.io javax.microedition.lcdui javax.microedition.midlet javax.microedition.rms
62 Personal Profile	javax.microedition.xlet javax.microedition.xlet.ixc
75 PDA Optional	javax.microedition.pim javax.microedition.file
80 <sup>1</sup> USB API	java.usb
82 Bluetooth	javax.bluetooth javax.obex
113 <sup>2</sup> Speech API	javax.speech
118 MIDP 2.0	javax.microedition.io javax.microedition.lcdui javax.microedition.game javax.microedition.media javax.microedition.media.control

	javax.microedition.midlet
	javax.microedition.pki
	javax.microedition.rms
120 WMA 1.0	javax.microedition.io
	javax.wireless.messaging
129 Personal Basis Profile	javax.microedition.xlet
	javax.microedition.xlet.ixc
135 MMAPAPI 1.0	javax.microedition.media
	javax.microedition.media.control
	javax.microedition.media.protocol
139 CLDC 1.1	javax.microedition.io
177 Security and Trust Services API	javax.microedition.apdu
	javax.microedition.jcrmi
	javax.microedition.pki
	javax.microedition.securityservice
	javax.microedition.io
	javacard.framework
	javacard.framework.service
	javacard.security
179 Location API	javax.microedition.location
180 SIP API	javax.microedition.sip
184 Mobile 3D Graphics API	javax.microedition.m3g
195 Information Module Profile	javax.microedition.io
	javax.microedition.midlet
	javax.microedition.rms
205 WMA 2.0	javax.microedition.io
	javax.wireless.messaging
211 <sup>3</sup> Content handler API	javax.microedition.content
226 <sup>1</sup> Scalable 2D Vector Graphics API	javax.microedition.m2d
234 <sup>1</sup> Advanced Multimedia Supplements	javax.microedition.media
	javax.microedition.media.control
	javax.microedition.media.control.audio3d
	javax.microedition.media.control.audioeffect
	javax.microedition.media.control.camera
	javax.microedition.media.control.imageeffect
	javax.microedition.media.control.tuner

### **MIDlet Sınıf Mimarisi**

MIDlet lerimiz çalışan uygulamalarımızdır diye belirtmiştik. Bizim her MIDlet' imiz MIDlet sınıfında türeme birer sınıftır. MIDlet sınıfı abstract bir sınıf olup üç adet abstract metodu ezmek zorundadır. Abstract sınıflar içerisinde abstract sınıflar barındırır ve tekbaşlarına nesnelere yaratılamaz. Yani MIDlet m= new MIDlet() diye bir nesne oluşturamayız.

MIDlet sınıf yapısındaki abstract sınıflar aşağıdaki gibidir.

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

startApp uygulamamız başlatıldığı anda çalışan metodumuzdur. Bu kod bloğu içerisinde MIDlet açıldığı anda neler yapmak istediğimizi yazabiliriz. Buna örnek olarak formumuzun oluşturulması ve gösterilmesi diyebiliriz.

pauseApp uygulamamız durdurulduğunda çalışan metodumuzdur. Bunları birer blok olarak düşünebiliriz.

destroyApp(boolean unconditional) ise uygulamamız sonlandırıldığında çalışır. Örnek olarak oyundan çıktığımız anda verilerimi kaydet diyebiliriz.

Örneğimizi bir sınıf olarak nitelendirecek olursak aşağıda Test adında bir adet MIDletimiz bulunmaktadır. Görüldüğü gibi bu sınıf bir MIDlet sınıfını miras almıştır ve üç adet metodu bulunmaktadır.

```
import javax.microedition.midlet.*;  
  
public class Test extends MIDlet {  
    public void startApp() {  
    }  
  
    public void pauseApp() {  
    }  
  
    public void destroyApp(boolean unconditional) {  
    }  
}
```

Bunun dışında javax.microedition.midlet kütüphanesi uygulamamıza import edilmiştir. Bu sayede MIDlet sınıflarımızı bu kütüphane içerisindeki sınıflardan yaratabiliriz.

Peki abstract metodlarımızdan herhangi birini kullanmadığımızda ne olur ? Aslında mantık olarak uygulama yöneticisinin uygulamanın çalıştığı anda ne yapacağını bilmesi gerekmektedir. Standart Java da bunlar için main metodlarını kullanabiliriz ancak MIDlet mimarisinde bu biraz daha basitleştirilmiş. Peki şimdi startApp metodumuzu kadırıp uygulamamızı çalıştırmayı deneyelim.

Bir hata aldık ve hatamız aşağıdaki gibi.

C:\Kitap\DemoKitap\src\Test.java:3: Test is not abstract and does not override abstract method startApp() in javax.microedition.midlet.MIDlet

```
public class Test extends MIDlet {
```

1 error

C:\Sabanci\DemoKitap\nbproject\build-impl.xml:183: Compile failed; see the compiler error output for details.

BUILD FAILED (total time: 0 seconds) □

Burada Őu notaya dikkat edelim “*Test is not abstract and does not override abstract method startApp() in javax.microedition.midlet.MIDlet*” burada sınıfımızın bir MIDlet abstract sınıfından türemiŐ olduĐunu ve startApp metodunun kullanılmadığını söylüyor. Bu durumda bizim her Őekilde bu üç metodu kullanmamız gerektiĐi net bir Őekilde çıkıyor.