Bab 1

Pengembangan Aplikasi Mobile

1.1 Tujuan

Dalam bab ini, kita akan mendiskusikan karateristik dari perangkat mobile, dan bagaimana hal tersebut mempengaruhi pengembangan program untuk perangkat ini. Kita akan diperkenalkan kepada Java 2 Mobile Edition (J2ME), termasuk pentingnya configuration dan profilenya.

- Pada akhir bab ini, pelajar diharapkan dapat menguasai :
- Mengidentifikasi karakteristik dari perangkat mobile
- Menjelaskan arsitektur J2ME
- Mengetahui peran atau aturan configuration dan profile
- Mengidentifikasi API yang disediakan oleh MIDP
- Menjelaskan daur hidup MIDlet

1.2 Perangkat Mobile

Perangkat mobile memiliki banyak jenis dalam hal ukuran, desain dan layout, tetapi mereka memiliki kesamaan karakteristik yang sangat berbeda dari sistem desktop.

- Ukuran yang kecil

  Perangkat mobile memiliki ukuran yang kecil. Konsumen menginginkan perangkat yang terkecil untuk kenyamanan dan mobilitas mereka.

- Memory yang terbatas

  Perangkat mobile juga memiliki memory yang kecil, yaitu primary (RAM) dan secondary (disk). Pembatasan ini adalah salah satu faktor yang mempengaruhi penulisan program untuk berbagai jenis dari perangkat ini. Dengan pembatasan jumlah dari memory, pertimbangan-pertimbangan khusus harus diambil untuk memelihara pemakaian dari sumber daya yang mahal ini.

- Daya proses yang terbatas

  Sistem mobile tidaklah setangguh rekan mereka yaitu desktop. Ukuran, teknologi dan biaya adalah beberapa faktor yang mempengaruhi status dari sumber daya ini. Seperti harddisk dan RAM, Anda dapat menemukan mereka dalam ukuran yang pas dengan sebuah kemasan kecil.

- Mengkonsumsi daya yang rendah

  Perangkat mobile menghabiskan sedikit daya dibandingkan dengan mesin desktop. Perangkat ini harus menghemat daya karena mereka berjalan pada keadaan dimana daya yang disediakan dibatasi oleh baterai-baterai.

- Kuat dan dapat diandalkan

Karena perangkat mobile selalu dibawa kemana saja, mereka harus cukup kuat untuk menghadapi benturan-benturan, gerakan, dan sesekali tetesan-tetesan air.

- Konektivitas yang terbatas

  Perangkat mobile memiliki bandwith rendah, beberapa dari mereka bahkan tidak tersambung. Kebanyakan dari mereka menggunakan koneksi wireless.

- Masa hidup yang pendek

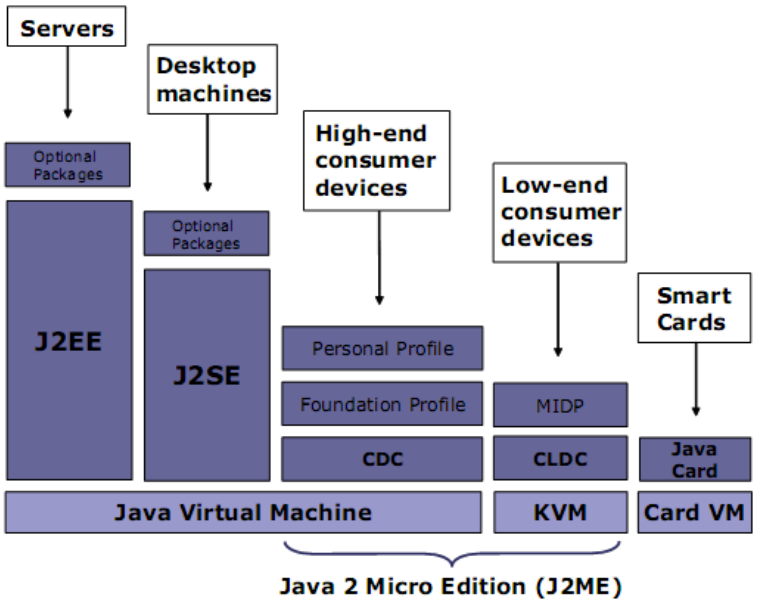
  Perangkat-perangkat konsumen ini menyala dalam hitungan detik kebanyakan dari mereka selalu menyala. Coba ambil kasus sebuah handphone, mereka booting dalam hitungan detik dan kebanyakan orang tidak mematikan handphone mereka bahkan ketika malam hari. PDA akan menyala jika anda menekan tombol power mereka.

1.3 Gambaran J2ME

1.3.1 Platform JAVA

Java dibuat pada tahun 1991 oleh James Gosling. Pada awalnya diberi nama Oak, dimana untuk menghormati pohon yang ada di luar jendela Gosling. Kemudian namanya diubah ke Java karena telah ada sebuah bahasa yang diberi nama Oak.Motivasi sesungguhnya dari Java adalah kebutuhan akan sebuah bahasa yang bisa digunakan pada berbagai platform yang bisa dimasukkan ke dalam berbagai produk elektronik seperti pemanggang roti dan lemari es. Salah satu dari proyek pertama yang dikembangkan menggunakan JAVA sebuah remote kontrol yang diberi nama Star 7. Pada saat yang sama, World Wide Web dan Internet berkembang sangat cepat. Gosling menyadari bahwa Java dapat digunakan untuk pemrograman Internet.

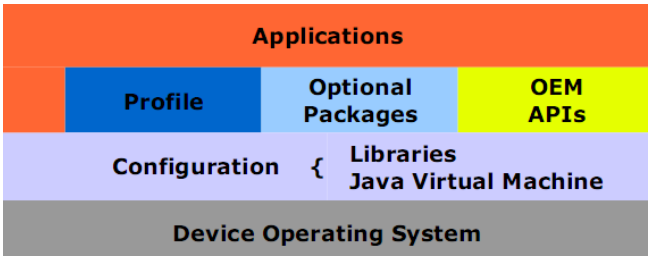
J2SE-Java 2 Platform, Standard Edition	Aplikasi Desktop
J2EE-Java 2 Platform, Enterprise Edition	Aplikasi enterprise dengan fokus pada pengembangan sisi webserver, termasuk servlet,JSP,EJB, dan XML
J2ME-Java 2 Platform, Micro Edition	Perangkat Mobile
JavaCard	Smart Cards



Gambar 1: Platform Java

1.3.2 Gambaran J2ME

J2ME adalah satu set spesifikasi dan teknologi yang fokus kepada perangkat konsumen. Perangkat ini memiliki jumlah memori yang terbatas, menghabiskan sedikit daya dari baterei, layar yang kecil dan bandwith jaringan yang rendah.Dengan perkembangbiakan perangkat mobile konsumer dari telepon, PDA, kotak permainan ke peralatan-peralatan rumah, Java menyediakan suatu lingkungan yang portable untuk mengembangkan dan menjalankan aplikasi pada perangkat ini. Program J2ME, seperti semua program JAVA adalah diterjemahkan oleh VM. Program-program tersebut dikompile ke dalam bytecode dan diterjemahkan denga Java Virtual Machine(JVM).Ini berarti bahwa program-program tersebut tidak berhubungan langsung dengan perangkat. J2ME menyediakan suatu interface yang sesuai dengan perangkat. Aplikasi-aplikasi tersebut tidak harus dikompile ulang supaya mampu dijalankan pada mesin yang berbeda.Inti dari J2ME terletak pada configuration dan profile-profile. Suatu configuration menggambarkan lingkungan runtime dasar dari suatu sistem J2ME. Ia menggambarkan core library, virtual machine, fitur keamanan dan jaringan.



Gambar 2: Arsitektur J2ME

Sebuah profile memberikan library tambahan untuk suatu kelas tertentu pada sebuah perangkat. profile-profile menyediakan user interface(UI) API,persistence, messaging library, dan sebagainya. Satu set library tambahan atau package tambahan menyediakan kemampuan program tambahan.

Pemasukan package ini ke dalam perangkat J2ME dapat berubah-ubah karena tergantung pada kemampuan sebuah perangkat. Sebagai contoh, beberapa perangkat MIDP tidak memiliki Bluetooth built-in, sehingga Bluetooth API tidak disediakan dalam perangkat ini.

1.3.3 Configuration

Suatu configuration menggambarkan fitur minimal dari lingkungan lengkap Java runtime. Untuk menjamin kemampuan portabilitas dan interoperabilitas optimal diantara berbagai macam perangkat yang dibatasi sumber dayanya(memory, prosesor, koneksi yang dibatasi), configuration tidak menggambarkan fitur tambahan. Suatu configuration J2ME menggambarkan suatu komplemen yang minimum dari teknologi JAVA. Adalah merupakan tugas profile-profile untuk menggambarkan tambahan library untuk suatu kategori perangkat tertentu.

configuration menggambarkan:

- Subset bahasa pemrograman JAVA
- Kemampuan Java Virtual Machine(JVM)
- Core platform libraries
- Fitur sekuriti dan jaringan

1.3.4 Profile

Suatu profile menggambarkan set-set tambahan dari API dan fitur untuk pasar tertentu, kategori perangkat atau industri. Sementara configuration menggambarkan library dasar, profile-profile menggambarkan library yang penting untuk membuat aplikasi-aplikasi efektif. Library ini memasukkan user interface, jaringan dan penyimpanan API.

1.4 CLDC

The Connected Limited Device Configuration (CLDC) menggambarkan dan menunjuk pada area berikut ini:

- Fitur Bahasa Java dan Virtual Machine(VM)
- Library dasar(java.lang.*,java.util.*)
- Input/Output(java.io.*)
- Kemanan
- Jaringan
- Internationalization

1.4.1 Fitur yang hilang

Fitur tertentu dari J2SE yang dipindahkan dari CLDC adalah :

- Finalization of class instances
- Asynchronous exceptions

- Beberapa error classes
- User-defined class loaders
- Reflection
- Java Native Interface (JNI)
- Thread groups dan daemon threads

Reflection, Java Native Interface (JNI) dan user-defined class loaders potensial menjadi lubang keamanan. JNI juga membutuhkan memory yang intensif sehingga dimungkinkan untuk tidak mendapat dukungan dari memory rendah sebuah perangkat mobile.

1.4.2 Karakteristik perangkat CLDC

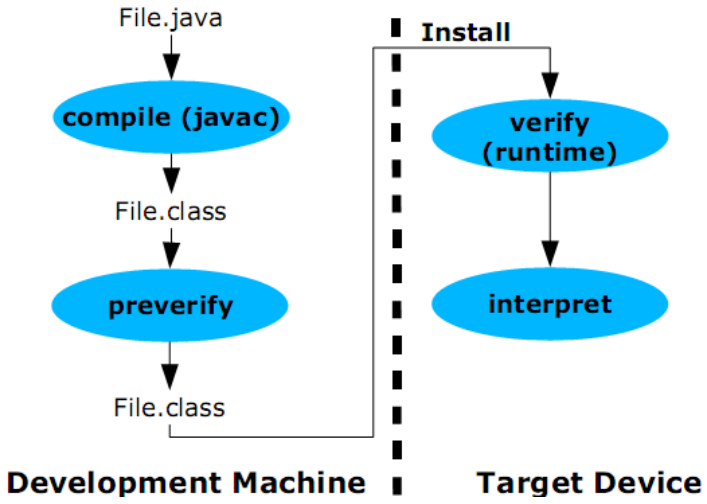
Perangkat yang diincar oleh CLDC mempunyai karateristik sebagai berikut:

- Memory minimal 192kb untuk platform Java.
- Prosesor dengan 16 atau 32 bit.
- Mengkonsumsi sedikit daya.
- Terbatas, koneksi jaringan yang sementara dengan pembatasan bandwith (biasanya wireless).

CLDC tidak menggambarkan instalasi dan daur hidup sebuah aplikasi, antarmuka(UI) dan penanganan peristiwa(event handling). Adalah merupakan tugas profile yang berada di bawah CLDC untuk menggambarkan area ini. Secara khusus, spesifikasi MIDP menggambarkan daur hidup aplikasi MIDP (MIDlet), library UI dan event handling(javax.microedition.lcdui.*).

1.4.3 Verifikasi Class

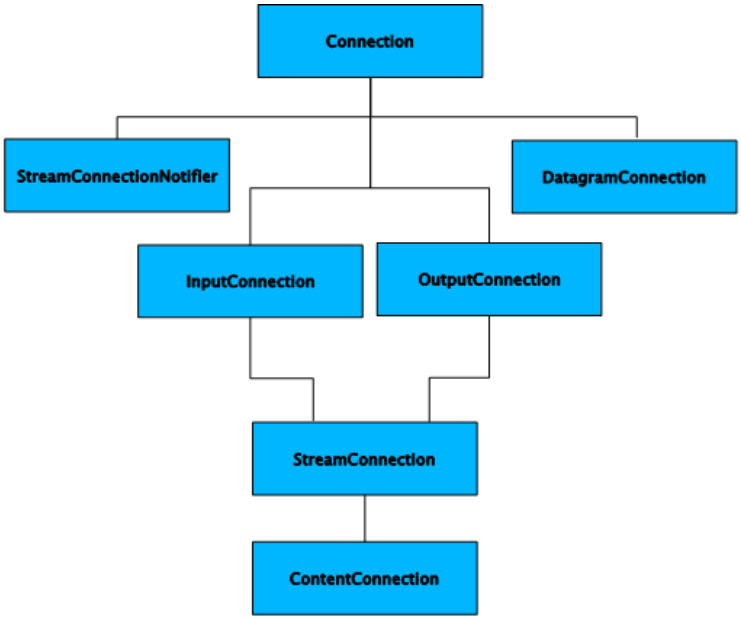
Spesifikasi CLDC memerlukan semua class untuk melewati proses verifikasi dua tingkat. Verifikasi pertama dilaksanakan diluar perangkat sebelum instalasi pada perangkat. Verifikasi kedua terjadi pada perangkat selama runtime dan dilaksanakan oleh KVM.



Gambar 3: Proses Verifikasi dua tingkat

1.4.4 Generic Connection Framework

The Generic Connection Framework menyediakan API dasar untuk koneksi dalam CLDC. Framework ini menyediakan suatu pondasi umum untuk koneksi seperti HTTP, Socket, dan Datagrams.GCF menyediakan suatu set API yang umum dan biasa yang memisahkan semua jenis koneksi. Perlu dicatat bahwa tidak semua jenis koneksi dibutuhkan untuk diterapkan oleh perangkat MIDP. Hirarki interface yang dapat diperluas dari GFC membuat proses penyamarataan menjadi mungkin. Jenis koneksi baru mungkin bisa ditambahkan ke dalamframework ini dengan memperluas hirarki ini.



Gambar 4: Hirarki koneksi GCF

1.5 CDC

Connected Device Configuration (CDC) adalah super set dari CLDC. CDC menyediakan lingkungan Java runtime yang lebih luas dibandingkan CLDC dan lebih dekat kepada lingkungan J2SE. CDC Java Virtual Machine (CVM) mendukung penuh Java Virtual Machine (JVM). CDC berisi semua API dari CLDC. CDC menyediakan suatu subset yang lebih besar dari semua class J2SE. Seperti CLDC, CDC tidak menggambarkan setiap class UI. Library UI digambarkan oleh profile-profile di bawah configuration ini.

Semua class yang terdapat dalam CDC datang dari package ini:

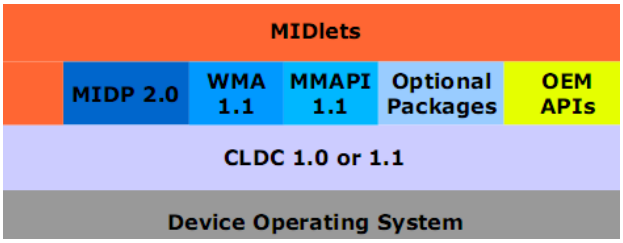
- java.io
- java.lang
- java.lang.ref
- java.lang.math
- java.net
- java.security
- java.security.cert
- java.text

- java.util
- java.util.jar
- java.util.zip

CDC juga memasukkan di dalamnya GCF. CDC memerlukan jenis koneksi tambahan seperti file dan dukungan datagram.

1.6 JTWI

The Java Technology for the Wireless Industry (JTWI) menetapkan satu set jasa dan spesifikasi standar. Berdasar spesifikasi JTWI, kata kuncinya adalah "untuk memperkecil fragmentasi API di dalam pasar telepon mobile, dan untuk mengirim spesifikasi yang dapat diprediksi,spesifikasi yang jelas untuk perangkat pabrik, operator, dan pengembang aplikasi".Dengan penyesuaian kepada JTWI, banyak aplikasi akan berjalan di suatu set yang lebih luas pada perangkat. Perangkat pabrik juga akan beruntung karena sebuah aplikasi yang besar akan tersedia untuk perangkat mereka.



Gambar 5: Komponen JTWI

1.7 MIDP

The Mobile Information Device Profile (MIDP) berada di atas dari CLDC. Anda tidak bisa menulis aplikasi mobile hanya dengan menggunakan CLDC API. Anda harus tetap memanfaatkan MIDP yang mendefinisikan UI. Spesifikasi MIDP, kebanyakan seperti CLDC dan API lainnya sudah digambarkan melalui Java Community Process (JCP). JCP melibatkan sebuah kelompok ahli berasal dari lebih dari 50 perusahaan, yang terdiri atas pabrik perangkat mobile, pengembang software. MIDP terus berkembang, dengan versi-versi masa depan yang telah lulus dari proses ketat JCP.

Berikut adalah perbandingan MIDP 1.0 dan MIDP 2.0 :

Spesifikasi	MIDP 1.0	MIDP 2.0
Display	96 x 54	96 x 54
Kedalaman Display	1-bit	1-bit
Bentuk piksel	Mendekati 1:1	Mendekati 1:1
Input	Keyboard dan touch screen	Keyboard dan touch screen
Memori	128 KB memori non-volatile untuk komponen MIDP	256 KB memori non-volatile untuk komponen MIDP

	8 KB memori non-volatile untuk data persistence yang dibuat oleh aplikasi 32 KB memori volatile untuk JRE	8 KB memori non-volatile untuk data persistence yang dibuat oleh aplikasi 128 KB memori volatile untuk JRE
Jaringan	Dua arah tanpa kabel (wireless)	Dua arah tanpa kabel (wireless)
Library J2ME	javax.microedition.lcdui javax.microedition.midlet javax.microedition.rms	javax.microedition.lcdui javax.microedition.midlet javax.microedition.rms javax.microedition.lcdui.game javax.microedition.media javax.microedition.pki
Multimedia	-	Support suara dan video

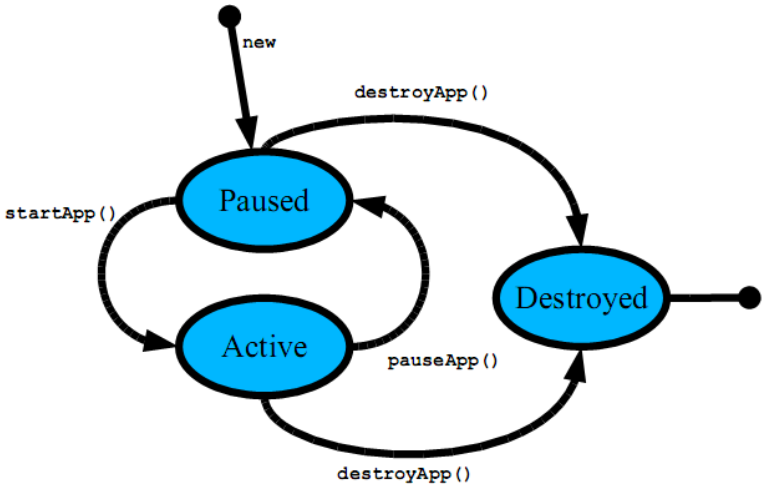
MIDP menggambarkan model aplikasi, UI API, penyimpanan dan jaringan yang kuat, permainan dan media API, kebijakan keamanan, penyebaran aplikasi dan ketetapan over-the-air.

1.8 MIDlet

Suatu aplikasi MIDP disebut MIDlet. Perangkat application management software (AMS) berinteraksi langsung dengan MIDlet dengan method MIDlet create, start, pause, dan destroy.MIDlet adalah bagian dari package javax.microedition.midlet. Sebuah MIDlet harus di-extend dengan class MIDlet. Dan dapat meminta parameter dari AMS seperti dirumuskan dalam application descriptor (JAD). Suatu MIDlet tidak harus memiliki (dan memang harus tidak mempunyai) sebuah method public static void main(String[] argv). Method tersebut tidak akan dikenal lagi oleh AMS sebagai titik awal sebuah program.

1.8.1 Siklus MIDlet

kehidupan MIDlet dimulai ketika di-instantiate oleh AMS. MIDlet pada awalnya masuk status “Pause” setelah perintah baru dibuat. AMS memanggil constructor public tanpa argumen dari MIDlet. Jika sebuah exception terjadi dalam constructor, MIDlet memasuki status “Destroyed” dan membuangnya segera. MIDlet masuk ke dalam status “Active” atas pemanggilan method startUp() oleh AMS. MIDlet masuk ke dalam status “Destroyed” ketika AMS memanggil method destroyApp(). Status ini juga kembali diakses ketika method notifyDestroyed() kembali dengan sukses kepada aplikasi. Dengan catatan bahwa MIDlet hanya bisa memasuki status “Destroyed” sekali dalam masa hidupnya.



Gambar 6: Daur hidup MIDlet

1.8.2 MIDlet suites

Aplikasi-aplikasi MIDlet dibungkus dan dikirim kedalam perangkat sebagai MIDlet suites. Sebuah MIDlet suite terdiri dari Java Archive (JAR) dan sebuah tambahan Java Application Descriptor (JAD). File JAD adalah suatu file teks yang berisi satu set atribut-atribut, beberapa dibutuhkan.

Bab 2

Memulai Pemrograman Mobile

2.1 Tujuan

Pada bagian ini, kita akan menggali tentang menulis, membangun, menggunakan emulator dan melakukan packaging aplikasi J2ME. Integrated Programming Environment yang akan kita gunakan adalah NetBeans (www.netbeans.org) dan NetBeans Mobility Pack.

Setelah menyelesaikan bagian ini, siswa diharapkan mampu:

- Membuat MIDlet sederhana
- Membuat sebuah project di NetBeans
- Membuat sebuah MIDlet menggunakan NetBeans Mobility Pack
- Menjalankan MIDlet di emulator

2.2 Pengenalan

IDE (Integrated Development Environment) adalah sebuah lingkungan pemrograman (programming environment) yang memiliki GUI builder, text atau code editor, compiler dan/atau interpreter dan debugger. Dalam hal ini, NetBeans Mobility Pack juga memiliki device emulator. Fasilitas ini bisa membuat kita melihat program kita pada device yang sesungguhnya.

2.3 "Hello, world!" MIDlet

Kita sudah mempelajari pada bagian sebelumnya tentang daur hidup MIDlet (MIDlet's life cycle). MIDlet mulai hidup ketika MIDlet dibuat oleh Application Management System (AMS) pada device. Agar kita dapat membuat MIDlet, kita harus membuat subclass dari MIDlet class dari javax.microedition.midlet package. Kita juga harus melakukan override atau implement pada method: startApp(), destroyApp() dan pauseApp(). Method-method tersebut adalah method yang diperlukan oleh AMS untuk menjalankan dan mengkontrol MIDlet.Tidak seperti program Java pada umumnya dimana method main() hanya digunakan sekali pada jalannya program, method startApp() mungkin akan dipanggil lebih dari sekali dalam daur hidup MIDlet. Sehingga Anda diharuskan tidak membuat satu inisialisasi code pada method startApp(). Daripada, anda dapat membuat MIDlet consturctor dan melakukan inisialisasi di situ.

Berikut ini adalah code program MIDP pertama kita:

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

public class HelloMidlet extends MIDlet implements CommandListener {
  Display display;
```

```
  Command exitCommand = new Command("Exit", Command.EXIT, 1);
  Alert helloAlert;
  public HelloMidlet(){
    helloAlert = new Alert(
      "Hello MIDlet", "Hello, world!", null, AlertType.INFO
    );
    helloAlert.setTimeout(Alert.FOREVER);
    helloAlert.addCommand(exitCommand);
    helloAlert.setCommandListener(this);
  }

  public void startApp() {
    if (display == null){
      display = Display.getDisplay(this);
    }
    display.setCurrent(helloAlert);
  }

  public void pauseApp() {
  }

  public void destroyApp(boolean unconditional) {
  }

  public void commandAction(Command c, Displayable d){
    if (c == exitCommand){
      destroyApp(true);
      notifyDestroyed(); // Exit
    }
  }
}
```

Selanjutnya kita akan mempelajari MIDlet pertama kita, difokuskan pada line yang penting dari code diatas:

```
public class HelloMidlet extends MIDlet implements CommandListener {
```

Seperti yang sudah kita katakan sebelumnya, kita harus membuat subclass dari MIDlet untuk membuat MIDP program. Pada line ini, kita sudah membuat subclass dari MIDlet dengan memberikan turunan kelas induk dan menamakannya HelloMIDlet.

```
  Display display;
  Command exitCommand = new Command("Exit", Command.EXIT, 1);
  Alert helloAlert;
```

Line diatas ini adalah variabel properties dari MIDlet. Kita membutuhkan object Display (hanya ada satu diplay per MIDlet) untuk melakukan fungsi menggambar pada layar. exitCommand adalah perintah yang akan kita taruh pada layar agar kita dapat keluardari program. Jika kita tidak memiliki perintah keluar, maka kita tidak memiliki cara untuk keluar dari MIDlet dengan benar.

```
  public HelloMidlet(){
    helloAlert = new Alert(
      "Hello MIDlet", "Hello, world!", null, AlertType.INFO
    );
    helloAlert.setTimeout(Alert.FOREVER);
    helloAlert.addCommand(exitCommand);
    helloAlert.setCommandListener(this);
  }
```

Consturctor melakukan inisialisasi dari object Alert. Kita akan mempelajari lebih lanjut dari Alert class pada bab berikutnya. Method addCommand() pada object Alert memberikan perintah "Exit" pada layar. Method setCommandListener() memberikan informasi kepada sistem untuk memberikan semua command events ke MIDlet.

```
public class HelloMidlet extends MIDlet implements CommandListener {
```

Code "implements CommandListener" adalah untuk command/key presses, sehingga program kita mampu menghandle "command" events. Jika kita melakukan implement CommandListener, kita harus membuat method commandAction().

```
  public void commandAction(Command c, Displayable d){
    if (c == exitCommand){
      destroyApp(true);
      notifyDestroyed(); // Exit
    }
  }
```

commandAction() diatas hanya menghandle request untuk perintah "Exit". Method diatas akan menghentikan program menggunakan notifyDestroyed() jika perintah "Exit" dijalankan atau ditekan.

```
  public void startApp() {
    if (display == null){
      display = Display.getDisplay(this);
    }
    display.setCurrent(helloAlert);
  }
```

Code diatas adalah bagian awal dari program kita ketika program kita sudah siap untuk ditampilkan oleh AMS. Perlu diingat bahwa method startApp() mungkin/bisa dimasukkan lebih dari sekali seperti pada daur hidup MIDlet. Jika MIDlet berhenti/dihentikan, seperti bila ada telepon masuk, program akan masuk ke state berhenti (pausedApp). Jika panggilan sudah selesai AMS akan kembali ke program dan memanggil method startApp() lagi. Method display.setCurrent() memberikan informasi ke sistem bahwa kita menginginkan object Alert untuk dimunculkan ke layar. Kita dapat mendapat tampilah object dengan memanggil method statis Display.getDisplay().

NetBeans Mobility Pack secara otomatis membuat Java Application Descriptor (JAD) untuk program Anda. NetBeans Mobility Pack menaruh file JAD pada folder "dist" dari folder project. Berikut ini adalah contoh file JAD yang dibuat oleh NetBeans Mobility Pack:

```
MIDlet-1: HelloMidlet, , HelloMidlet
MIDlet-Jar-Size: 1415
MIDlet-Jar-URL: ProjectHello.jar
MIDlet-Name: ProjectHello
MIDlet-Vendor: Vendor
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.0
```

Sekarang kita siap untuk mengcompile, melakukan pemaketan (package) pada aplikasi MIDlet pertama kita.

2.4 Compilation dan Packaging MIDlets

Sebelum kita menggunakan integrated tools untuk mengcompile dan melakukan packaging aplikasi MIDlet (MIDlet suite), kita akan mencoba menggunakan command line. Aplikasi MIDlet biasanya dipaketkan ke dalam sebuah file yaitu file JAR. File ini adalah file terkompres, seperti file ZIP. Pada implementasinya, Anda dapat membuka file JAR menggunakan program dekompresor file ZIP.

Aplikasi MIDlet terdiri dari:

- File JAR
- File Java Application Descriptor (JAD)

File JAR memiliki:

- File class
- Manifest file describing the contents of the archive
- File manifest yang menjelaskan isi dari arsip
- Sumber: image/icon, video, data, dll. Digunakan oleh aplikasi

File manifest, manifest.mf adalah seperti file JAD. File ini digunakan oleh appication manager dari device. Beberapa field yang diperlukan oleh file manifest adalah:

- MIDlet-Name

- MIDlet-Version
- MIDlet-Vendor
- MIDlet-<n> (dimana n adalah angka dari 1, untuk setiap MIDlet di file JAR)
- MicroEdition-Profile
- MicroEdition-Configuration

Selanjutnya kita mengcompile file source java:

```
javac –bootclasspath C:\WTK23\lib\cldcapi11.jar;C:\WTK23\lib\midpapi20.jar *.java
```

Program Compiler Java, "javac", harus berada pada path Anda. Jika anda melihat error seperti "cannot find file" atau "not an executable", Anda bisa mengkonsultasikan dengan panduan instalasi untuk distribusi Java development kit Anda tentang bagaimana memasukkan executable PATH dari lokasi tools yang ada di Java.

Selanjutnya kita melakukan pre-verify dari file class:

```
preverify
-classpath C:\WTK23\lib\cldcapi11.jar;C:\WTK23\lib\midpapi20.jar;.
-d . HelloMidlet
```

Preverify sudah berada di wireless toolkit dari java.sun.com. Masukkan perintah ini pada sebuah baris. Langkah terakhir adalah membuat file JAR tersebut:

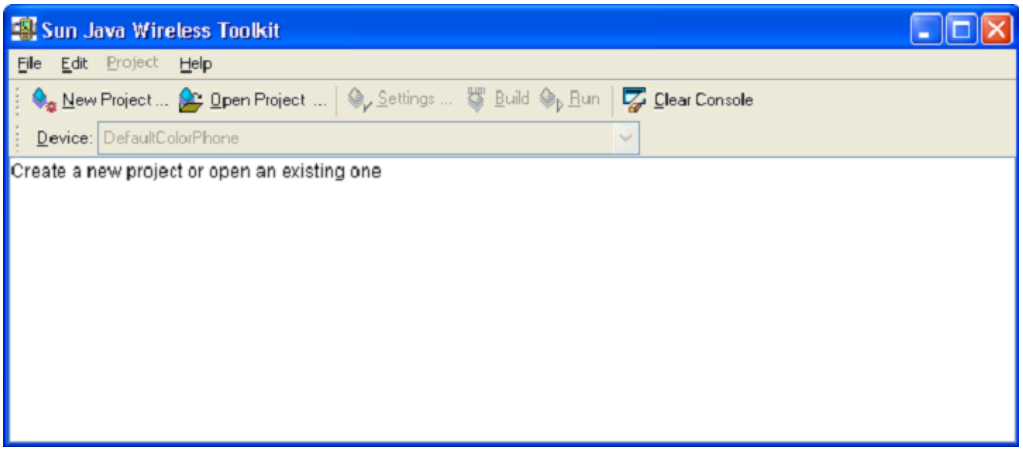
```
jar cvfm HelloMidlet.jar manifest.txt HelloMidlet.class
```

Program jar sudah berada di Java Development Kit, dan lokasinya harus dimasukkan pada executeable path Anda. Perintah ini akan membuat file JAR dengan nama file HelloMidlet.jar. File manifest.txt namanya diganti dengan manifest.mf pada file JAR.

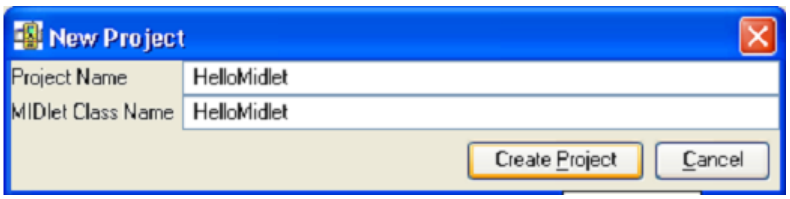
2.5 Menggunakan Sun Wireless Toolkit

Sekarang kita menggunakan Sun Wireless Toolkit untuk mengcompile dan memaketkan aplikasi MIDlet / MIDlet suite (mengandung satu MIDlet)

Buka ktoolbar (dari Wireless Toolkit distribution):

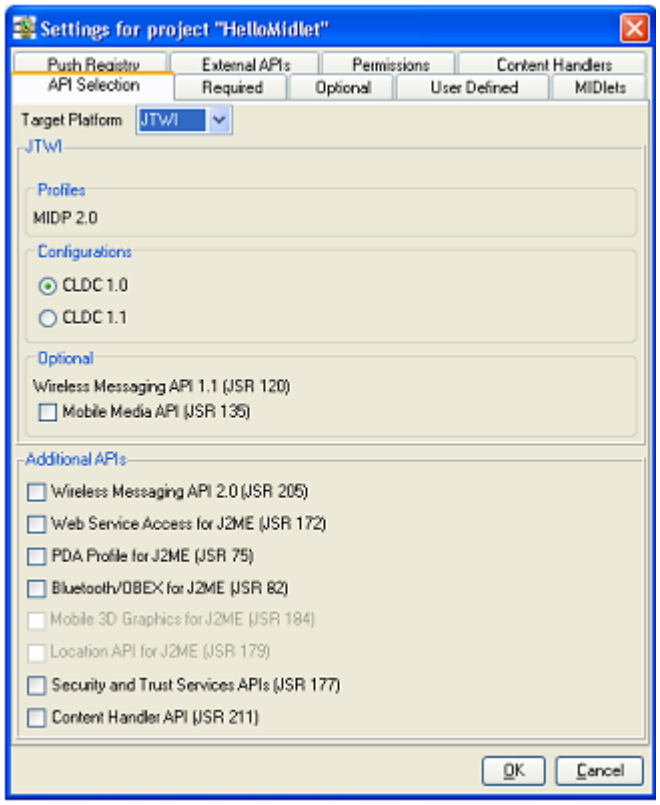


Buat sebuah project:

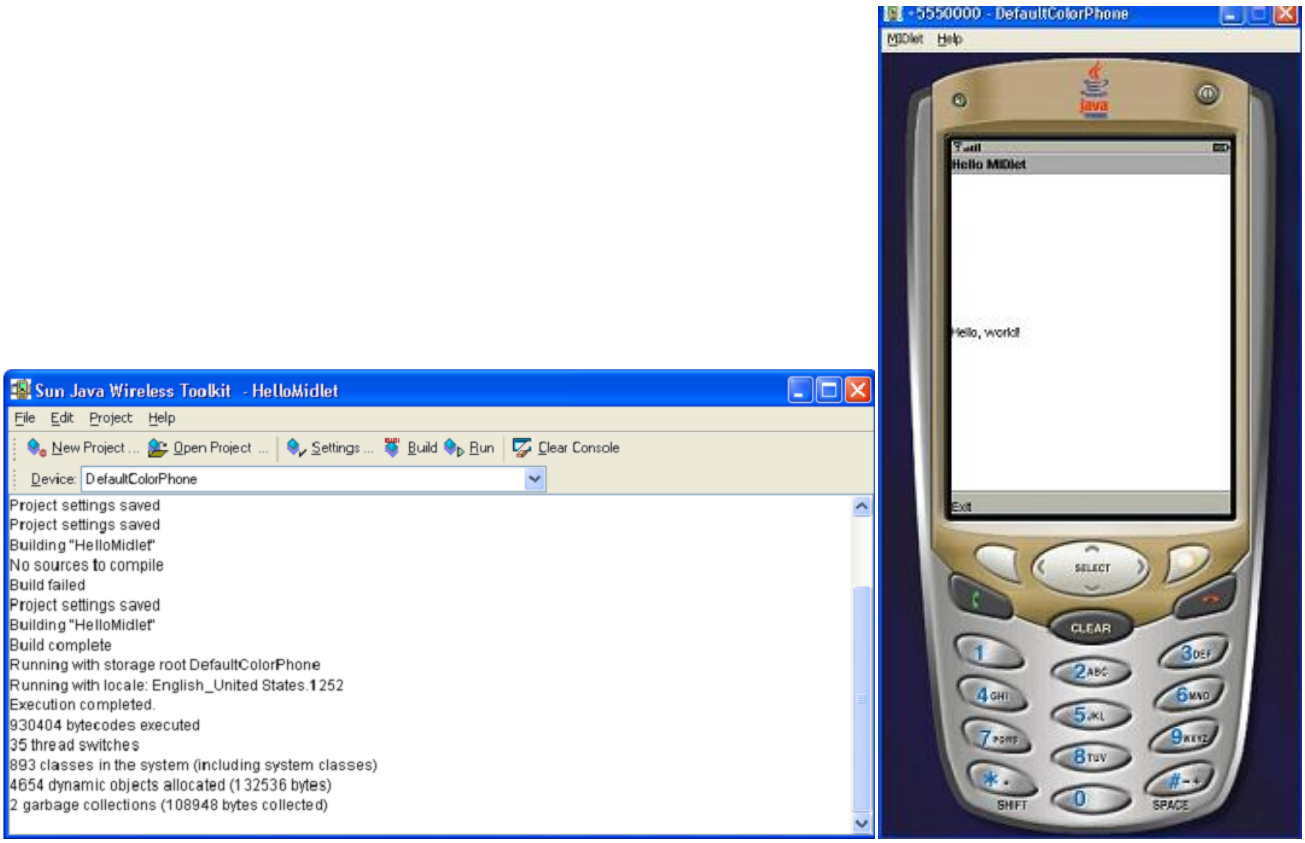


Pada Setting window, anda dapat merubah banyak pilihan-pilihan dari beberapa opsi konfigurasi untuk project Anda. Anda dapat memilih konfigurasi yang akan bekerja, package/API yang diperlukan, konfigurasi Push Registry dan yang lain. Untuk tujuan kita kali ini, kita akan menggunakan konfigurasi default project. Click "OK" untuk selesai

membuat project.



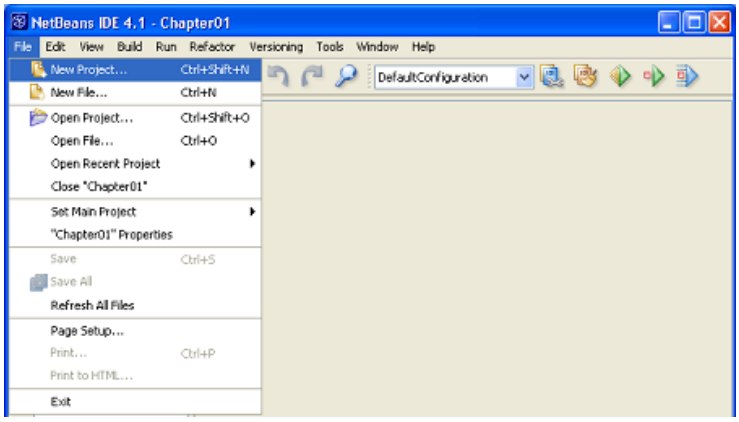
Copy HelloMidlet.java kedalam direktori "src": Pada jendela ini berada di direktori: C:\WTK23\apps\HelloMidlet\src (dimana C:\WTK23 adalah lokasi Anda menginstall wireless toolkit). Click "Build" dan "Run":



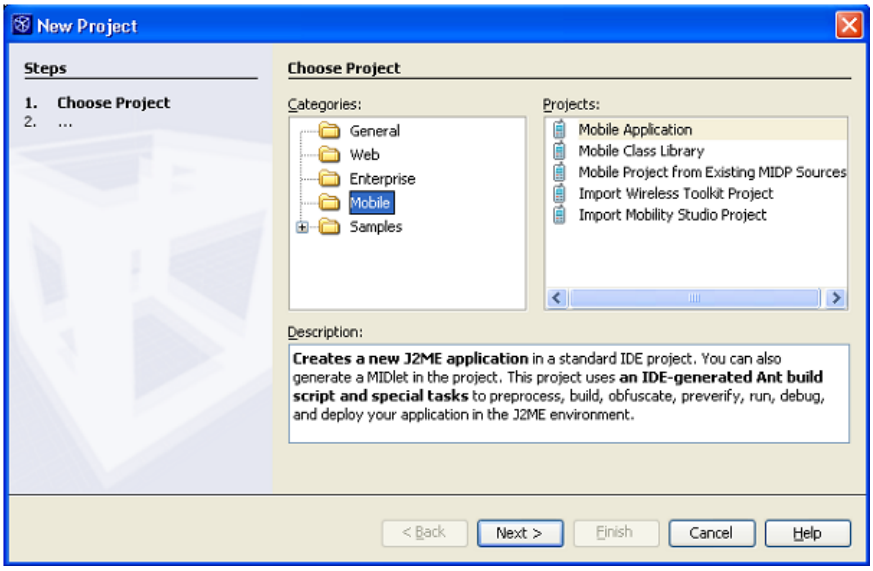
2.6 Menggunakan NetBeans Mobility Pack

Seperti yang telah dijelaskan pada awal bab ini tentang hal yang diperlukan, NetBeans dan NetBeans Mobility Pack harus sudah terinstall di komputer Anda.

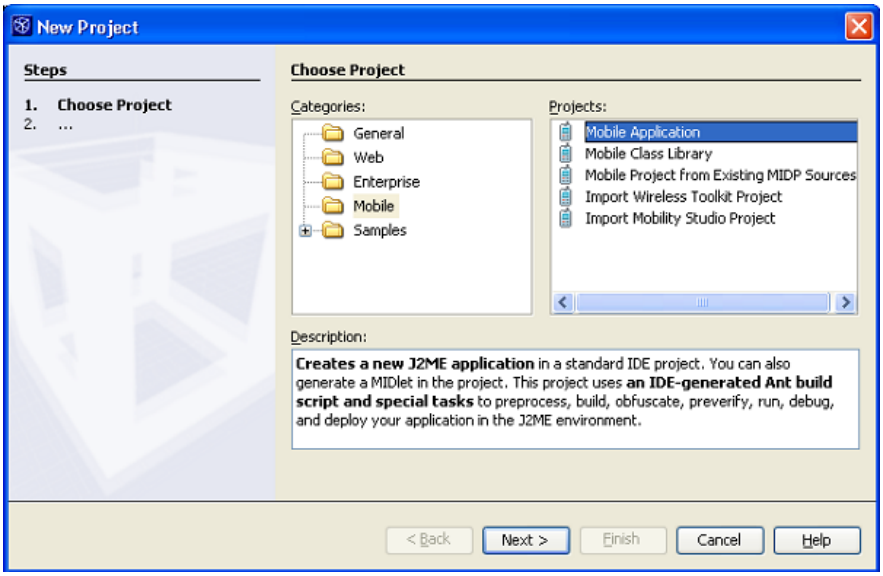
Langkah 1: Membuat project baru



Langkah 2: Memilih kategori "Mobile"

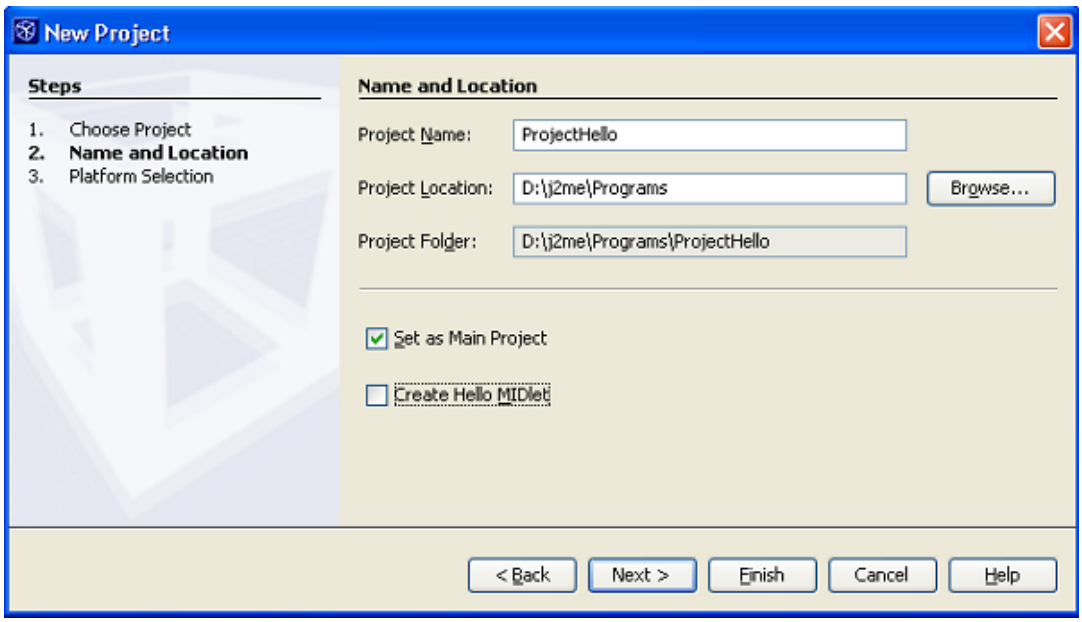


Langkah 3: Memilih "Mobile Application"

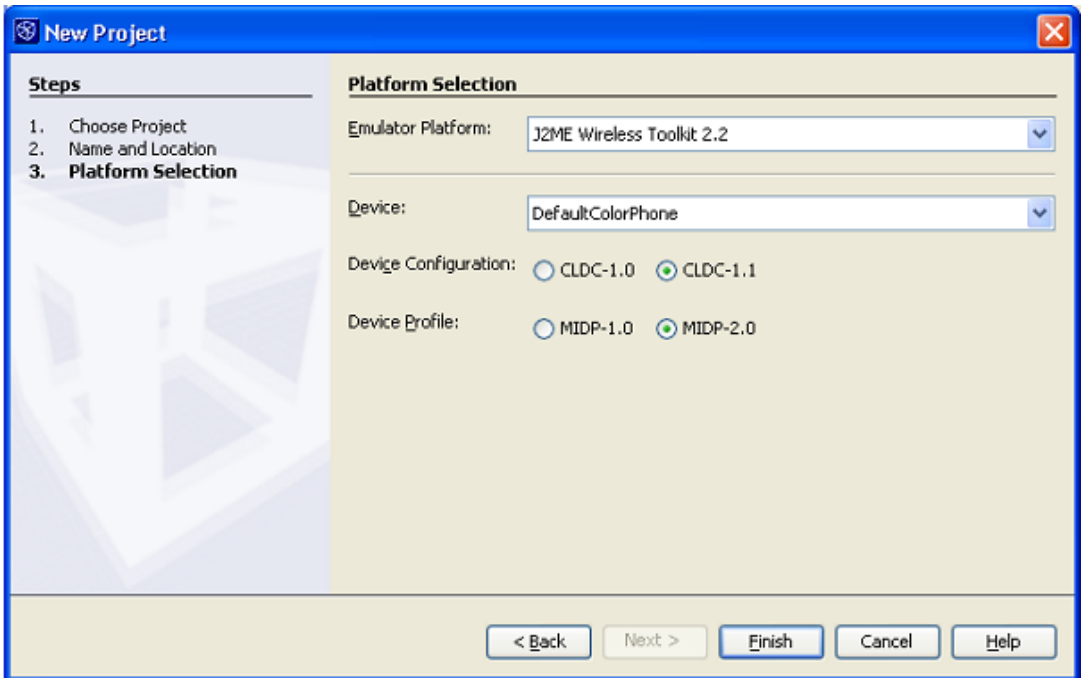


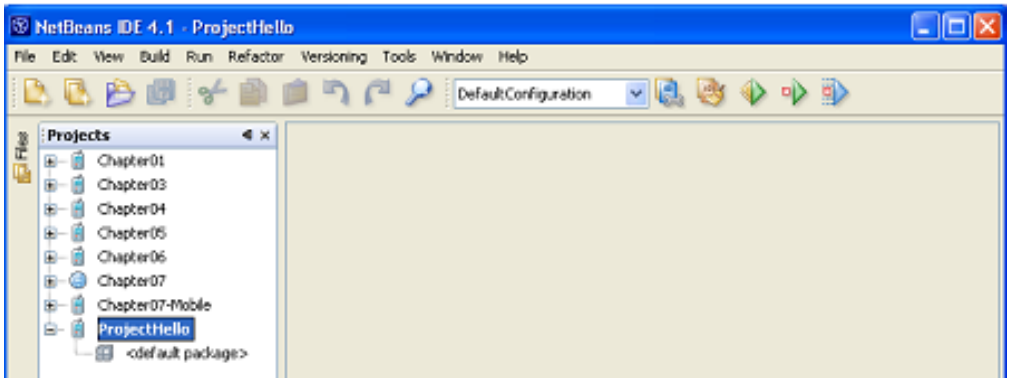
Langkah 4: Beri nama project dan tentukan lokasinya

(Hilangkan tanda pada "Create Hello MIDlet", kita akan membuat MIDlet kita sendiri nantinya)



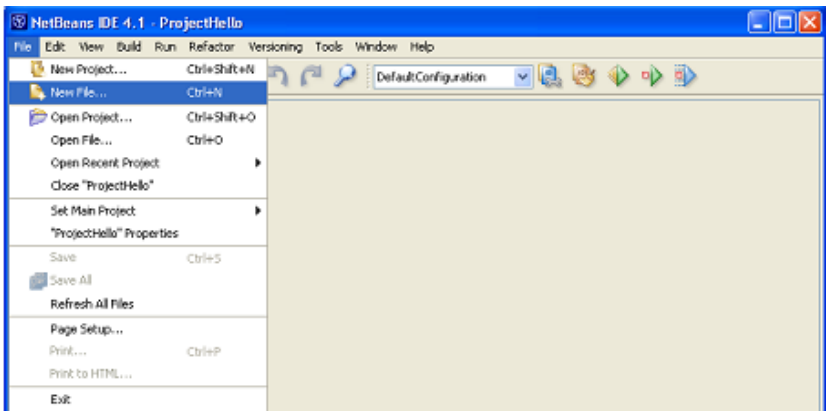
Memilih Platform (optional)



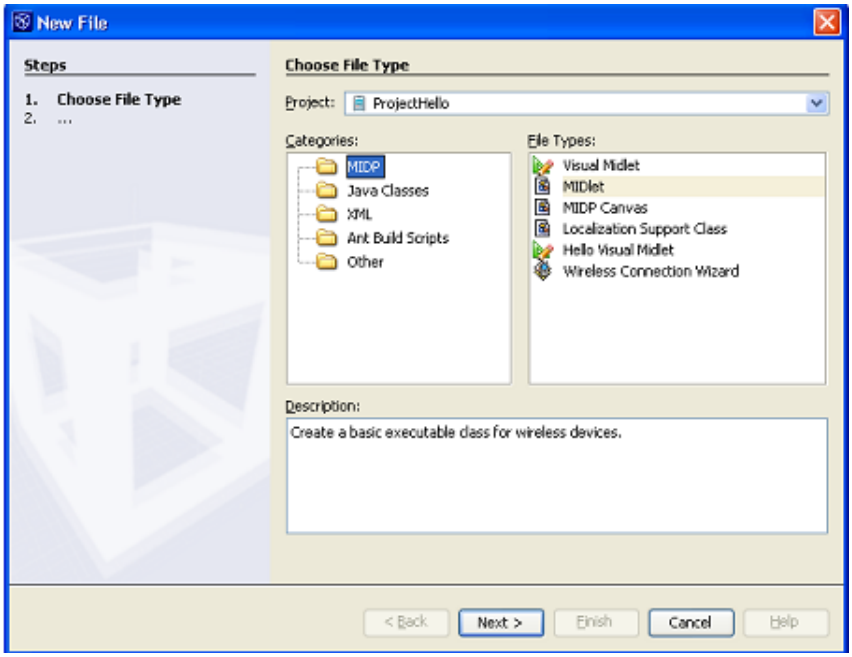


Gambar 1: Mobile Project yang baru dibuat (NetBeans Mobility Pack)

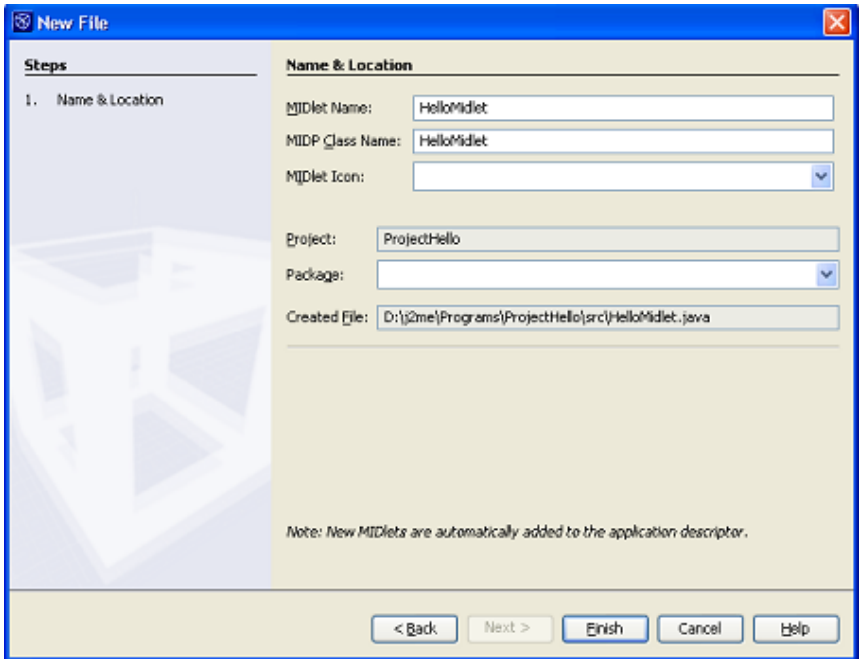
Langkah 6: Membuat sebuah MIDlet baru



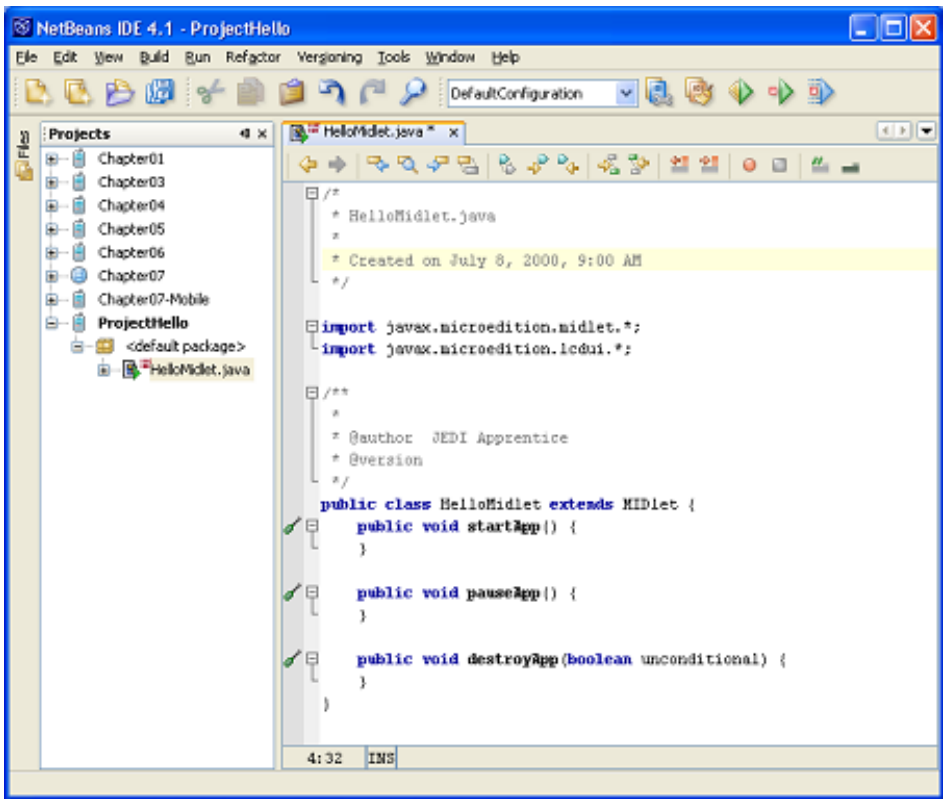
Langkah 7: Memilih MIDP "Category" dan MIDlet "File Type"



Langkah 8: Memberi nama MIDlet

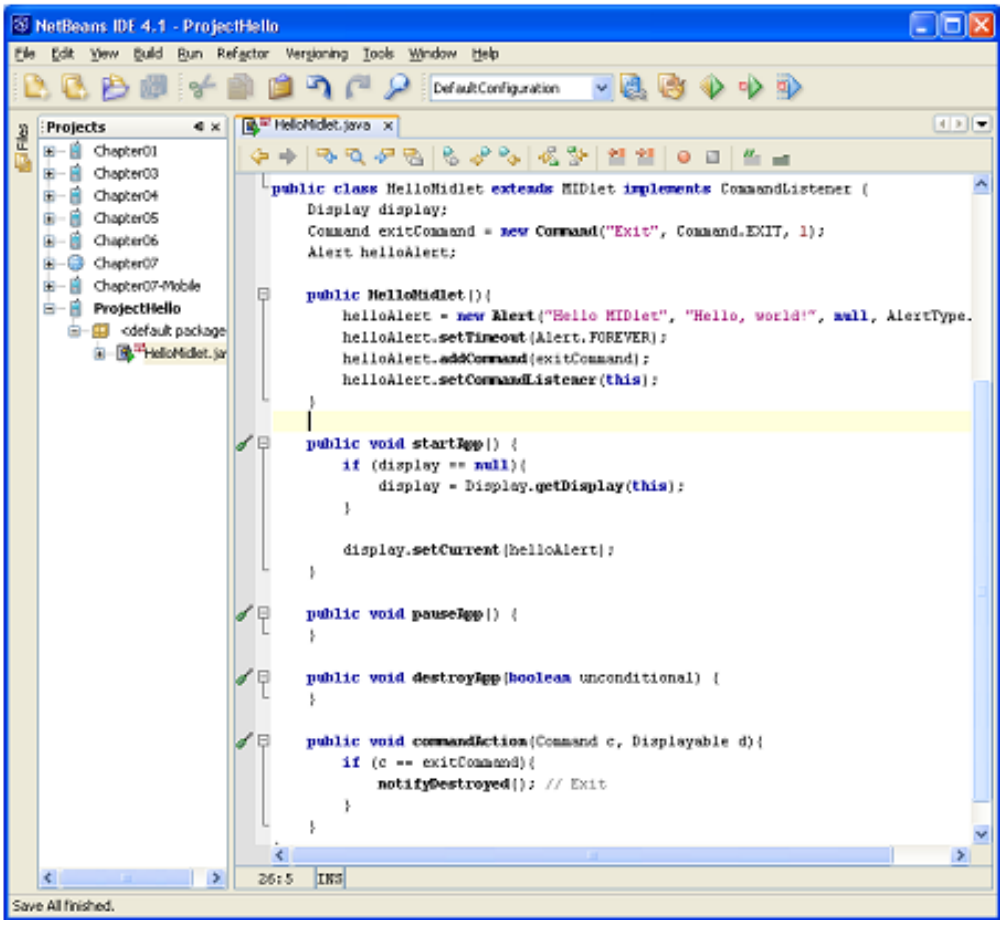


Langkah 9: Tampil beberapa sintaks secara otomatis



Gambar 2: Membuat MIDlet secara otomatis membuat method MIDlet yang diperlukan

Langkah 10: Mengganti code yang dibuat secara otomatis dengan code program kita



Langkah 11: Mengcompile dan Menjalankan (Run) MIDlet di Emulator

Langkah 12: Menjalankan MIDlet kita di Emulator

Gambar 3: Hello World MIDlet

Bab 3

High Level User Interface

3.1 Tujuan

Pada akhir pembahasan, para pembaca diharapkan dapat:

- Mengetahui keuntungan dan kerugian dengan menggunakan high-level dan low-level UI classes
- Mengetahui desain MIDlets menggunakan komponen high-level UI
- Mengidentifikasi perbedaan sub-classes pada screen
- Mengetahui perbedaan item-item yang dapat dimasukkan kedalam sebuah object Form

3.2 MIDP User Interface

MDIP user interface didesain untuk peralatan mobile. Aplikasi MDIP ditunjukan pada area limited screen. Peralatan memory juga menjadi faktor penting jika perlengkapan mobile hanya memiliki kapasitas memory yang kecil.Dengan berbagai macam peralatan mobile, dari berbagai model mobile phones sampai PDAs, MIDP user interface telah didesain untuk lebih fleksibel dan mudah digunakan dalam berbagai macam peralatan ini. MIDP mempunyai class yang dapat menangani fungsi high-level dan low-level user interface. High-level UI interfaces didesain secara fleksibel. Penampilan dari komponen ini tidak didefinisikan secara spesifik. Penampilan screen yang sebenarnya dari berbagai macam komponen ini digunakan dari satu peralatan ke peralatan yang lain. Tetapi para programmer telah teryakinkan oleh kegunaan dari high-level komponen UI interfaces memiliki persamaan dalam berbagai spesifikasi-pengimplementasi secara keseluruhan.

Berikut perbedaan High-Level UI dengan Low-Level UI

High Level UI	Low-Level UI
highly portable across devices	Memungkinkan semua peralatan
look dan feel sama dengan peralatannya	Spesifik aplikasi look and feel
Memiliki interaksi seperti scrolling yang dienkapsulasi	Pengimplementasiannya harus dengan petunjuk sendiri
Penampilannya tidak dapat digambarkan secara aktual	Penampilannya tidak dapat digambarkan dalam satuan pixel
Tidak memiliki akses untuk peralatan dengan feature yang spesifik	Mengakses masukkan low-level hanya dengan menekan

Kapan menggunakan High-Level UI

- Saat membangun aplikasi text-based yang mudah
- Saat Anda ingin aplikasi Anda dapat dengan mudah dipertukarkan dengan berbagai macam peralatan (Portabilitas)

- Saat Anda ingin aplikasi Anda memiliki tampilan yang sama dengan komponen UI yang lain dari berbagai peralatan
- Saat Anda ingin kode Anda dapat menjadi sesedikit mungkin, ketika sebuah interaksi ditangani oleh API

Kapan menggunakan Low-Level UI

- Saat Anda memerlukan sebuah high-level untuk mengkontrol tampilan dari suatu aplikasi
- Saat aplikasi Anda membutuhkan tempat yang tepat dari elemen-elemen yang ada pada screen
- Saat membuat game secara grafik; meskipun Anda tetap dapat menggunakan high-level UI pada menu game, hal tersebut lebih disarankan untuk membuat menu UI Anda sendiri untuk menghindari seamless atmosphere bagi para user
- Saat sebuah aplikasi membutuhkan akses ke low-level yang memiliki inputan seperti key presses
- Jika aplikasi Anda akan diimplementasikan pada layar navigasi Anda sendiri

3.2.1 Display

Inti dari MIDP user interfaces adalah display. Yang merupakan satu-satunya kemudahan dari Display per MIDlet. MIDlet dapat mendapatkan referensi Display object dengan menggunakan method static Display.getDisplay(), melewatkan referensi tersebut ke MIDlet instance. MIDlet dijamin dengan display object tidak akan berubah dengan adanya eksistensi instance MIDlet. Hal ini berarti bahwa variebel dikembalikan (returned) ketika Anda memanggil getDisplay() dan tidak akan berpengaruh jika anda memenggilnya dengan startApp() atau destroyApp() (Lihat pada gambar Midlet Life Cycle).

3.2.2 Displayable

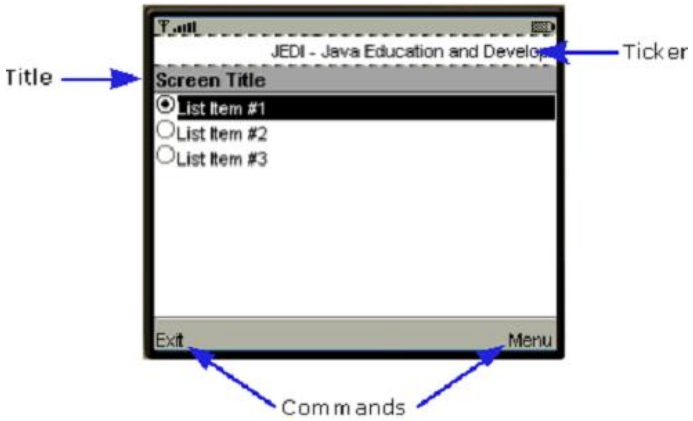
Hanya satu displayable yang ditampilkan pada satu waktu. Secara langsung, displayable tidak ditampilkan pada layar. Seuabh displayable dapat ditampilkan dengan memanggil method setCurrent() dari Display instance. Method setCurrent() harus dipanggil pada saat memulai aplikasi, dengan kata lain sebuah screen kosong akan ditampilkan atau aplikasi tersebut tidak akan dijalankan.

Method startApp dari MIDlet merupakan suatu tempat dimana Anda dapat menaruh method pemanggil setCurrent(). Tetapi Anda harus mempertimbangkan bahwa dalam MIDlet startApp() dapat dipanggil lebih dari satu kali. Untuk memberhentikan MIDlet sementara waktu dapat dipause dengan memanggil fungsi pauseApp(), dengan adanya incoming call, memungkinkan startApp() dipanggil lagi (setelah ada telepon masuk). Maka dengan memanggil setCurrent() pada method pada startApp(), dan ada kemungkinan layar akan menjadi gelap (blank) pada screen displayed yang sebelumnya, sampai adanya penghentian sementara (pause by the phone call). Sebuah displayable dapat memiliki nama, beberapa perintah(command), commandListener dan Ticker.

3.2.3 Title

Sebuah Displayable memiliki title yang berhubungan dengan dirinya sendiri. Posisi dan penampilan dari title tersebut merupakan piranti spesifik yang hanya dapat ditentukan oleh peralatan dari aplikasi yang sedang dijalankan. Sebuah title ditampilkan pada Displayable dengan memanggil setTitle().Dengan memanggil method ini maka seketika akan meng-update title pada Displayable. Jika pada saat Displayable ditampilkan pada layar, MIDP specification states menyebutkan bahwa title harus dirubah dengan implementasi "Memungkinkan untuk dilakukan dengan cepat".

Memberi parameter null pada setTitle() berarti menghapus title pada Displayable. Merubah atau menghapus sebuah title dari Displayable dapat mempengaruhi ukuran area untuk isi dari Displayable tersebut.Jika terjadi perubahan ukuran area terjadi, MIDlet akan diberitahu dengan memanggil kembali method sizeChanged().



Gambar Ticker, Title dan Commands

3.2.4 Command

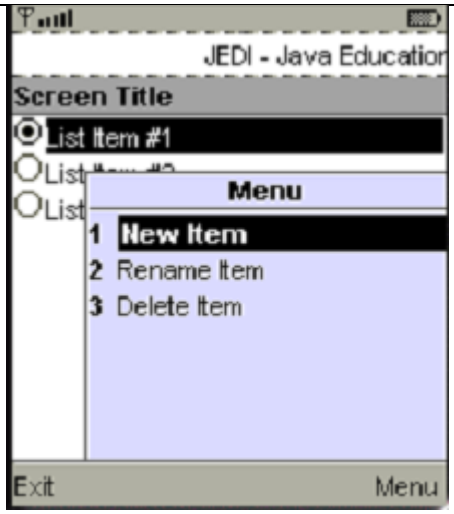
Dengan adanya kekurangan ukuran pada screen, MIDP tidak menggambarkan sebuah menu bar. Untuk menggantikan menu bar, MIDlet memiliki Commands. Biasanya Command diimplementasikan sebagai soft key atau item dalam sebuah menu. Object Command hanya berisi informasi tentang action yang harus dikerjakan pada saat Command diaktifkan. Dia tidak berisikan kode yang akan dieksekusi pada saat Command tersebut dipilih. Properti CommandListener dari Displayable berisi action yang akan dieksekusi saat Command diaktifkan. CommandListener merupakan interface yang spesifik pada single method :

```
public void commandAction(Command command, Displayable displayable)
```

Mapping dari Commands pada peralatan bergantung pada nomer yang telah ditetapkan atau programable button pada peralatan. Jika nomer dari Command tidak benar pada semua button, maka memungkinkan peralatan menaruh beberapa atau semua Command pada menu dan peta pada menu dan button akan diberi label "Menu".

```
Command exitCommand = new Command("Exit", Command.EXIT, 1);
Command newCommand = new Command("New Item", Command.OK, 1);
Command renameCommand = new Command("Rename Item", Command.OK, 1);
Command deleteCommand = new Command("Delete Item", Command.OK, 1);
```

```
...
list.addCommand(exitCommand);
list.addCommand(newCommand);
list.addCommand(renameCommand);
list.addCommand(deleteCommand);
```



Gambar: Listing program untuk mapping Commands kedalam Displayable

Command Label

Diasumsikan bahwa screen yang berukuran kecil dari target sebuah peralatan, selalu menjadi faktor ketika membangun aplikasi MIDP. Asumsi ini juga dapat diterapkan untuk Command label. Command label harus singkat, namun deskriptif, maka hal itu harus benar pada screen dan tetap dapat dipahami oleh user. Ketika long label ditentukan, hal tersebut akan ditampilkan kapan saja pada saat sebuah implementasi sistem dilihat secara sesuai. Tidak ada pemanggilan API yang menetapkan label yang akan ditampilkan. Hal tersebut juga memungkinkan bahwa sebuah Command akan menampilkan short label pada saat Command lain pada screen yang sama menampilkan long labels.

Command Type

Sebuah Command yang diperkenalkan pada peralatan sering disebut device-dependent. Seorang programer dapat mengetahui spesifikasi tipe dari Command. Tipe ini akan ditampilkan sebagai hint pada tempat Command diletakkan.

Berbagai macam tipe Command:

```
Command.OK, Command.BACK,
Command.CANCEL, Command.EXIT,
Command.HELP, Command.ITEM,
Command.SCREEN, Command.STOP
```

Gambar: Tampilan Command yang berbeda pada implementasi telepon yang berbeda

Command Priority

Aplikasi dapat menetapkan spesifikasi Command yang penting pada priority property. Hal ini merupakan integer property dan nilai rendah yang sangat penting. Priority property juga hanya sebuah hint pada tempat dimana seharusnya Command ditempatkan. Biasanya implementasi menentukan posisi dari Command oleh tipenya. Jika terdapat lebih dari satu Command dari tipe yang sama, secara normal priority akan mempertimbangkan penempatan Command.

3.2.5 CommandListener

CommandListener merupakan interface dengan single method:

```
void commandAction(Command command, Displayable displayable)
```

Method commandAction() akan dipanggil jika Command dipilih. Variabel Command merupakan referensi Command yang telah dipilih. Tampilan merupakan Displayable (atau screen) dimana Command ditempatkan dan saat action “pilih” terjadi. CommandAction() harus dikembalikan dengan seketika, jika tidak maka pengeksekusian aplikasi akan diblock. Hal ini dikarenakan, spesifikasi MIDP tidak memerlukan implementasi untuk membuat sebuah pembatas untuk pengiriman event.

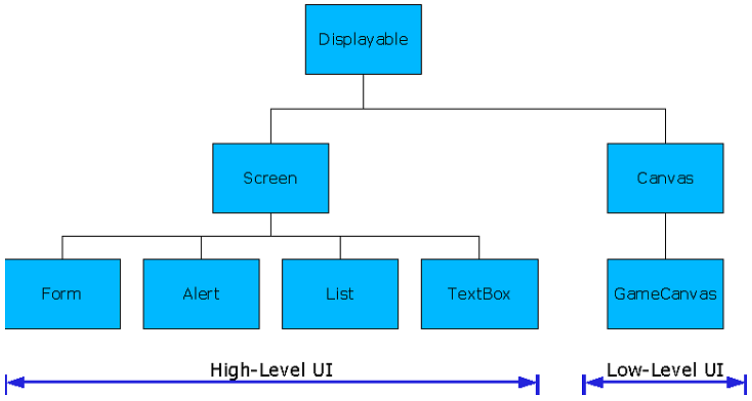
3.2.6 Ticker

Ticker adalah sebuah baris dari text yang dapat discrolling secara terus-menerus pada display. Method konstruktor dari ticker menerima text string untuk ditampilkan. Hal tersebut hanya memiliki dua method lain, yaitu getter dan setter untuk text ini: String getString() dan void setString(String text). Tidak ada cara lain pada sebuah aplikasi untuk mengontrol kecepatan dan arah dari scrolling text. Scrolling tidak dapat dipause atau distop. Jika spasi diletakkan pada text, hal tersebut tidak akan ditampilkan pada layar. Semua baris text akan ditampilkan sebagai single line dari scrolling text. Sebuah ticker dapat dipasang pada Displayable dengan memanggil setTicker(). Jika ticker telah ada pada Displayable, maka akan diganti oleh ticker yang baru yang terdapat dalam parameter. Memberi parameter null pada setTicker akan mengganti semua ticker yang telah dimasukkan pada Displayable. Menghapus ticker dari Displayable dapat menyebabkan perubahan ukuran

area dari isi Displayable tersebut. Jika perubahan ukuran area terjadi, maka MIDlet akan memanggil sebuah ukuran dengan method sizeChanged(). Pada ticker object Displayable boleh berbagi suatu kejadian(action).

3.2.7 Screen

Screen merupakan inti abstrak class yang digunakan untuk high-level UI ketika canvas merupakan Displayable abstrak class untuk low-level UI. Berikut ini empat subclasses dari abstract class screen : Form, TextBox, List dan Alert.

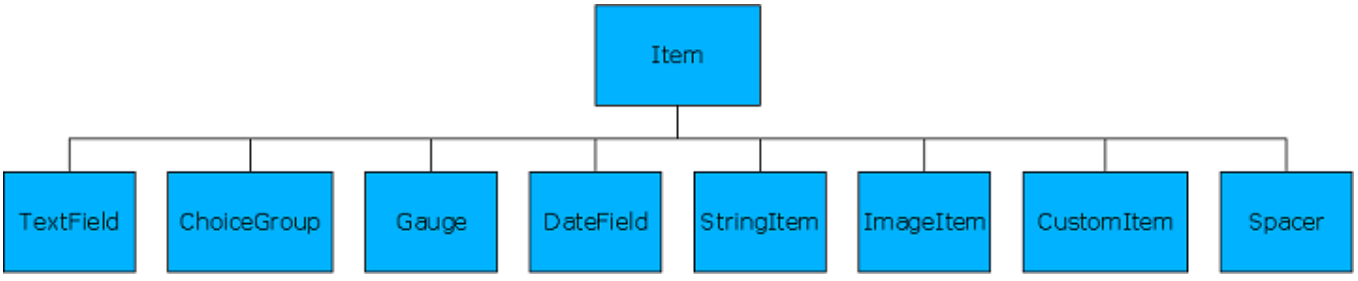


Gambar: Displayable Class Heirarchy

3.2.8 Item

Items merupakan komponen yang dapat diletakan kedalam container, seperti Form atau Alert. Sebuah item dapat memiliki property seperti dibawah ini:

Property	Default Value
Label	Dikelompokan pada subclass konstruktor
Commands	-
defaultCommand	null
ItemCommandListener	null
Layout directive	LAYOUT_DEFAULT
Preferred width and height	-1 (unlocked)



Gambar: Item Class Heirarchy

Spesifikasi layout dari item dengan Form. Direktif layout dapat dikombinasikan menggunakan bitwise atau operasi (|). Bagaimanapun juga, beberapa direktif bersifat mutually exclusive. Berikut ini direktif horizontal alignment yang mutually exclusive:

```
LAYOUT_LEFT
LAYOUT_RIGHT
LAYOUT_CENTER
```

Berikut ini direktif vertical alignment yang juga mutually exclusive:

```
LAYOUT_TOP
LAYOUT_BOTTOM
LAYOUT_VCENTER
```

Berikut ini layout yang lain dari direktif (tidak mutually exclusive):

```
LAYOUT_NEWLINE_BEFORE
LAYOUT_NEWLINE_AFTER
LAYOUT_SHRINK
LAYOUT_VSHRINK
LAYOUT_EXPAND
LAYOUT_VEXPAND
LAYOUT_2
```

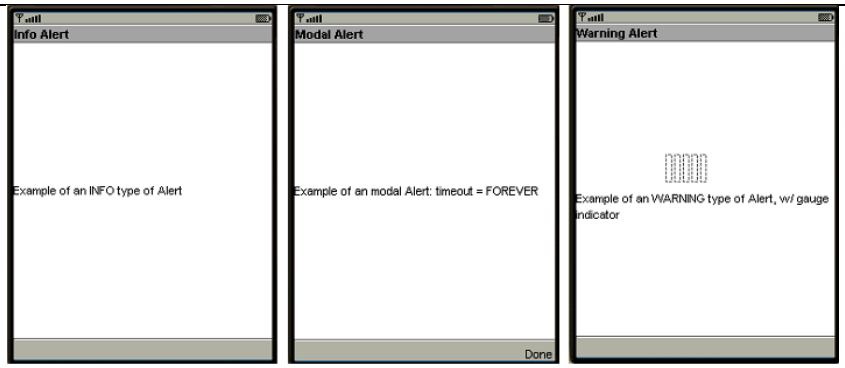
3.3 Alert

Alert merupakan sebuah screen yang dapat menampilkan text dan gambar. Alert merupakan komponen untuk menampilkan error dan warning, display text dan informasi gambar atau untuk mendapatkan informasi dari user. Alert ditampilkan untuk spesifikasi peeriode dari waktu. Waktu di-set menggunakan method setTimeout() dan method tersebut dispesifikasikan dalam unit millliseconds. Hal tersebut dapat dibuat untuk ditampilkan hingga user mengaktifkan perintah (“Done”) dengan menspesifikasikan spesial timeout dari Alert.FOREVER. Alert juga dapat menampilkan komponen Gauge (Lihat pada Gauge item) sebagai indikator. Ketika alert berisi text yang tidak sesuai dengan screenful dan harus discroll, maka secara otomatis alert menge-set ke modal(timeout di set kepada Alert.FOREVER).

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class AlertExample extends MIDlet implements CommandListener {
  Display display;
  Form mainForm;
  Command exitCommand = new Command("Exit", Command.EXIT, 0);
  Command okCommand = new Command("Ok", Command.OK, 0);
  Gauge gauge = new Gauge(null, false, 5, 0);
  Command[] commands = {
```

```
    new Command("Alarm", Command.OK, 0),
    new Command("Confirmation", Command.OK, 0),
    new Command("Info", Command.OK, 0),
    new Command("Warning", Command.OK, 0),
    new Command("Error", Command.OK, 0),
    new Command("Modal", Command.OK, 0)
  };
  Alert[] alerts = { new Alert("Alarm Alert", "Example of an Alarm type of
Alert", null, AlertType.ALARM), new Alert("Confirmation Alert", "Example of
an CONFIRMATION type of Alert", null,  AlertType.CONFIRMATION) new
Alert("Info Alert", "Example of an INFO type of Alert", null,
AlertType.INFO), new Alert("Warning Alert", "Example of an WARNING type of
Alert, w/ gauge indicator", null, AlertType.WARNING), new Alert("Error
Alert", "Example of an ERROR type of Alert, w/ an 'OK' Command", null,
AlertType.ERROR), new Alert("Modal Alert", "Example of an modal Alert:
timeout = FOREVER", null, AlertType.ERROR),
  };

  public AlertExample(){
    mainForm = new Form("JEDI: Alert Example");
    mainForm.addCommand(exitCommand);
    for (int i=0; i< commands.length; i++)
      mainForm.addCommand(commands[i]);
    mainForm.setCommandListener(this);
    // Menambah sebuah gauge dan menge-set timeout (milliseconds)
    alerts[3].setIndicator(gauge);
    alerts[3].setTimeout(5000);
    // Menambah sebuah command untuk Alert
    alerts[4].addCommand(okCommand);
    // Menge-Set alert
    alerts[5].setTimeout(Alert.FOREVER);
  }

  public void startApp() {
    if (display == null){
      display = Display.getDisplay(this);
      display.setCurrent(mainForm);
    }
```

```
  }

  public void pauseApp() {}

  public void destroyApp(boolean unconditional) {}

  public void commandAction(Command c, Displayable d){
    if (c == exitCommand){
      destroyApp(true);
      notifyDestroyed(); // Exit
    }
    for (int i=0; i<commands.length; i++)
      if (c == commands[i])
        display.setCurrent(alerts[i]);
  }
}
```



Gambar: Perbedaan tipe-tipe Alert.

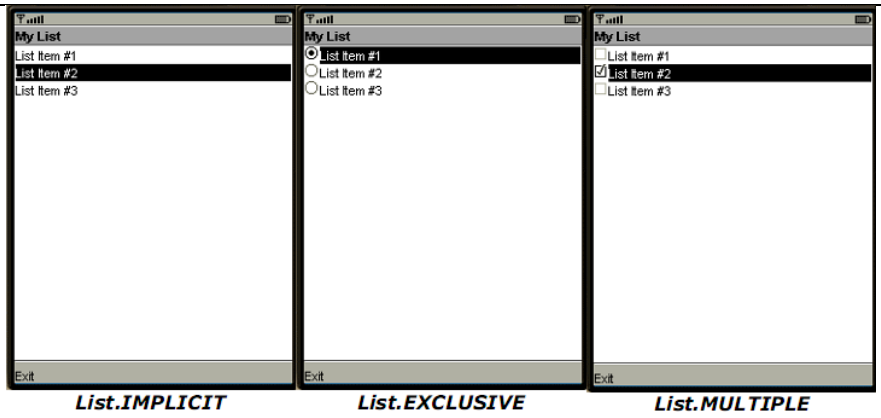
3.4 List

List merupakan subclass dari screen yang berisi sebuah daftar dari suatu pilihan. Sebuah list dapat dibagi menjadi tiga tipe: IMPLICIT, EXCLUSIVE atau MULTIPLE. Jika List bertipe IMPLICIT dan user mengeksekusi tombol "select", commandAction() dari list commandListener akan dipanggil. Default perintahnya adalah List.SELECT_COMMAND. Untuk tipe IMPLICIT dan EXCLUSIVE, GetSelectedIndex() mengembalikan index dari element yang dipilih. Untuk tipe MULTIPLE, getSelectedFlags() mengembalikan sebuah array dari boolean yang berisi state dari elemen-elemen. isSelected(int index) mengembalikan state dari elemen dalam pemberian posisi index.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class ListExample extends MIDlet implements CommandListener {
```

```
  Display display;
  List list;
  Command exitCommand = new Command("Exit", Command.EXIT, 1);
  Command newCommand = new Command("New Item", Command.OK, 1);
  Command renameCommand = new Command("Rename Item", Command.OK, 1);
  Command deleteCommand = new Command("Delete Item", Command.OK, 1);
  Ticker ticker = new Ticker("JEDI-Java Education and Development
Initiative");

  public ListExample(){
    list = new List("JEDI: List Example", List.IMPLICIT);
    list.append("List Item #1", null);
    list.append("List Item #2", null);
    list.append("List Item #3", null);
    list.setTicker(ticker);
    list.addCommand(exitCommand);
    list.addCommand(newCommand);
    list.addCommand(renameCommand);
    list.addCommand(deleteCommand);
    list.setCommandListener(this);
  }

  public void startApp() {
    if (display == null){
      display = Display.getDisplay(this);
      display.setCurrent(list);
    }
  }

  public void pauseApp() {}

  public void destroyApp(boolean unconditional) {}

  public void commandAction(Command c, Displayable d){
    if (c == exitCommand){
      destroyApp(true);
      notifyDestroyed(); // Exit
    }
```

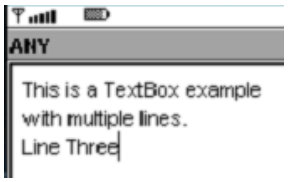
```
    if (c == List.SELECT_COMMAND){
      int index = list.getSelectedIndex();
      String currentItem = list.getString(index);
      // menjalankan suatu hal
    }
  }
}
```



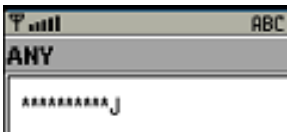
Gambar: Tipe-tipe List

3.5 Text Box

TextBox merupakan sub-class dari screen yang dapat digunakan untuk mendapatkan input text dari user. Hal ini memperbolehkan user untuk memasukan dan mengedit text. TextBox hampir sama dengan TextField(Lihat pada item TextField) karena dia dapat memiliki input constraint dan input modes. Perbedaannya dengan TextField adalah user dapat memasukan garis baru(ketika input constraint di-set untuk semua "ANY"). Isi dari TextBox dapat diambil kembali dengan menggunakan method getString().



Gambar:TextBox tipe ANY (multi-line)



Gambar: TextBox dengan modifikasi PASSWORD

3.6 Form

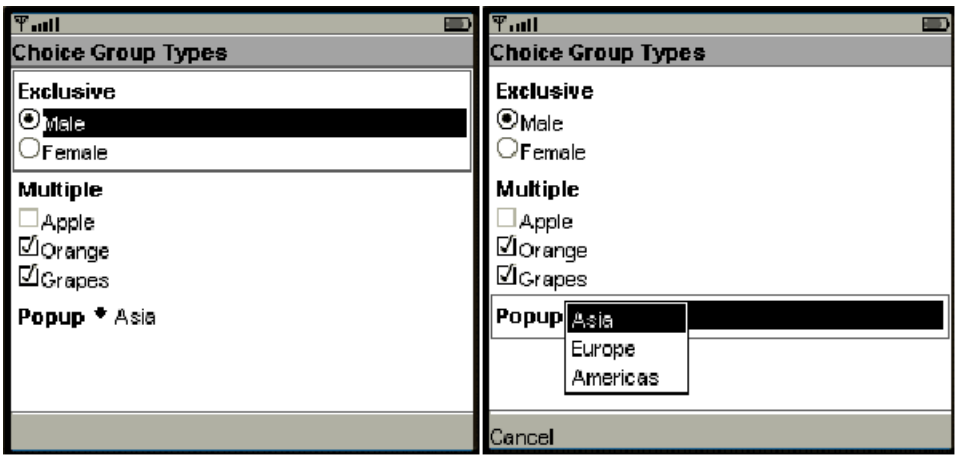
Form merupakan subclass dari Screen. Form merupakan conntainer untuk item subclass, seperti TextField, StringItem, ImageItem, DateField dan ChoiceGroup. Dia menghandle layout untuk komponen ini. Dan juga menghandle traversal antar komponen-komponen dan scrolling dari Screen. Item ditambahkan dan dimasukkan

ke dalam sebuah Form menggunakan method append() dan insert(), berturut-turut. Item direferensikan menggunakan index zero-based.

3.7 ChoiceGroup

Item Choicegroup merupakan group dari selectable choice. Sebuah choice boleh berisi sebuah text, gambar atau kedua-duanya. Choice boleh EXCLUSIVE (hanya satu pilihan yang dapat dipilih) atau MULTIPLE (banyak pilihan yang dapat dipilih pada suatu waktu). Jika ChoiceGroup bertipe POPUP, hanya satu choice yang ditampilkan. Popup selection akan ditampilkan ketika item ini dipilih. Dari popup seleksi ini, user diperbolehkan memilih pilihannya. Choice yang ditampilkan selalu choice yang dipilih.GetSelectedIndex() mengembalikan nilai index pada element dari ChoiceGroup yang dipilih. GetSelectedFlags() mengembalikan sebuah array dariboolean yang merespon elemen dari Choicegroup. isSelected(int index) mengembalikan state dari elemen yang diberikan oleh posisi index.

```
choiceForm = new Form("Choice Group Types");
  choiceForm.addCommand(exitCommand);
  choiceForm.setCommandListener(this);

  choiceExclusive = new ChoiceGroup("Exclusive", Choice.EXCLUSIVE);
  choiceExclusive.append("Male", null);
  choiceExclusive.append("Female", null);
  choiceForm.append(choiceExclusive);

  choiceMultiple = new ChoiceGroup("Multiple", Choice.MULTIPLE);
  choiceMultiple.append("Apple", null);
  choiceMultiple.append("Orange", null);
  choiceMultiple.append("Grapes", null);
  choiceForm.append(choiceMultiple);

  choicePopup = new ChoiceGroup("Popup", Choice.POPUP);
  choicePopup.append("Asia", null);
  choicePopup.append("Europe", null);
  choicePopup.append("Americas", null);
  choiceForm.append(choicePopup);
```

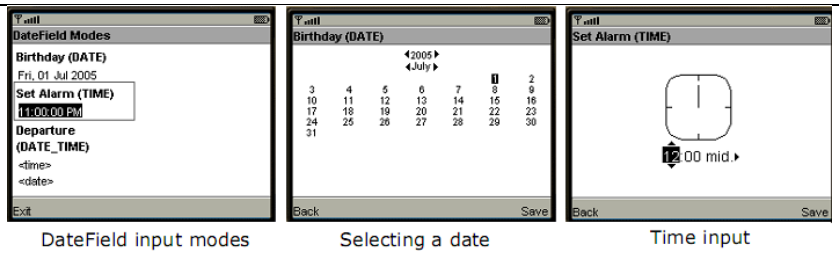


Gambar: Tipe dari Choice Group

3.8 Date Field

Komponen DateField digunakan untuk masukan tanggal dan waktu dari user. DateField dapat diisi dengan date entry(mode DATE), time entry (mode TIME) atau keduanya (mode DATE_TIME). Method getDate() mengembalikan nilai suatu item. Dia akan mengembalikan nilai null jika item tidak diinialisasi terlebih dahulu. Jika mode dari DateField adalah DATE, komponen time dari pengembalian nilai akan di-set menjadi nol. Jika modenya adalah TIME, komponen date akan di-set menjadi "Januari 1, 1970".

```
dateForm = new Form("DateField Modes");
dateForm.addCommand(backCommand);
dateForm.setCommandListener(this);
DateField dateonly =
new DateField("Birthday (DATE)", DateField.DATE);
DateField timeonly =
new DateField("Set Alarm (TIME)", DateField.TIME);
DateField datetime =
new DateField("Departure (DATE_TIME)", DateField.DATE_TIME);
dateForm.append(dateonly);
dateForm.append(timeonly);
dateForm.append(datetime);
```



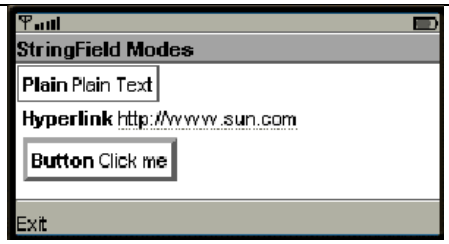
DateField input modes Selecting a date Time input

Gambar: mode DateField dan input screens

3.9 String Item

StringItem merupakan komponen read-only. Dia terdiri dari label dan text.Secara bebas StringItem menerima tampilan mode parameter. Tampilan dari mode dapat berupa Item.PLAIN, Item.HYPERLINK atau Item.BUTTON.Jika tampilan sebuah mode bertipe HYPERLINK atau BUTTON, default Command dan ItemCommandListener harus di-set didalam Item.

```
stringForm = new Form("StringField Modes");
stringForm.addCommand(exitCommand);
stringForm.setCommandListener(this);
StringItem plain = new StringItem("Plain", "Plain Text", Item.PLAIN);
StringItem hyperlink = new StringItem("Hyperlink", "http://www.sun.com",
Item.HYPERLINK);
hyperlink.setDefaultCommand(new Command("Set", Command.ITEM, 0));
hyperlink.setItemCommandListener(this);
StringItem button = new StringItem("Button", "Click me", Item.BUTTON);
button.setDefaultCommand(new Command("Set", Command.ITEM, 0));
button.setItemCommandListener(this);
stringForm.append(plain);
stringForm.append(hyperlink);
stringForm.append(button);
```



Gambar: StringItem

3.10 Image Item

ImageItem merupakan Image sederhana yang dapat dimasukan kedalam komponen, seperti Form. ImageItem menerima item layout sebagai parameter (Lihat pada bagian Item):

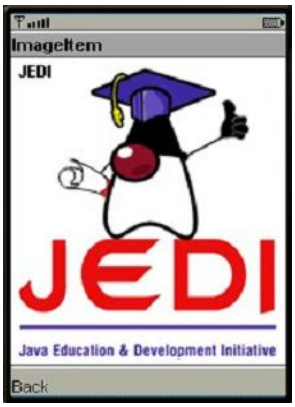
```
public ImageItem(String label, Image img, int layout, String altText)
```

Konstruktor yang lain menerima tampilan mode yang bertipe Item.PLAIN, Item.HYPERLINK atau Item.BUTTON (Lihat pada bagian StringItem):

```
public ImageItem(String label, Image image, int layout, String altText, int
appearanceMode)
imageForm = new Form("ImageItem");
imageForm.addCommand(backCommand);
imageForm.setCommandListener(this);
```

```
try {
  Image img = Image.createImage("/jeni.png");
  ImageItem image = new ImageItem("JENI", img, Item.LAYOUT_CENTER, "logo");
  imageForm.append(image);
} catch (Exception e){e.printStackTrace();}
```

File “jeni.png” sangat penting untuk dimasukan kedalam project dengan menggunakan operating system's manager dan menaruh image tersebut kedalam direktori project dibawah subdirektori “src”. Kemudian project direfresh dengan mengklik kanan nama project dan pilih “Refresh Folders”.



Gambar: ImageItem

3.11 Text Field

TextField merupakan Item dimana user dapat memasukan encode. Beberapa batasan exclusive yang dapat di-set yaitu:

```
TextField.ANY
TextField.EMAILADDR
TextField.NUMERIC
TextField.PHONENUMBER
TextField.URL
TextField.DECIMAL
```

Masukan tersebut juga dapat memiliki macam-macam modifikasi:

```
TextField.PASSWORD
TextField.UNEDITABLE
TextField.SENSITIVE
TextField.NON_PREDICTIVE
TextField.INITIAL_CAPS_WORD
TextField.INITIAL_CAPS_SENTENCE
```

Modifikasi dapat di-set dengan menggunakan bit-wise OR (|) operator (atau toggled menggunakan bit-wise XOR operator ^) pada input constraint. Sebagai konsekuensinya, modifikasi dapat diperoleh dari pengembalian nilai dari getConstraint() bit-wise operator AND(&). Sejak nilai modifikasi juga dikembalikan oleh getConstraint(), Masukan main constraint dapat diekstrak dengan menggunakan bit-wise operator AND

dengan TextBox.CONSTAINT_mask dan nilai pengembalian dari getConstaints(). GetString() mengembalikan isi dari TextField sebagai nilai sebuah String.

```
textForm = new Form("TextField Types");
textForm.addCommand(backCommand);
textForm.setCommandListener(this);
TextField ANY = new TextField("ANY", "", 64,  TextField.ANY);
TextField EMAILADDR =
new TextField("EMAILADDR", "", 64,  TextField.EMAILADDR);
TextField NUMERIC =
new TextField("NUMERIC", "", 64,  TextField.NUMERIC);
TextField PHONENUMBER =
new TextField("PHONENUMBER", "", 64,  TextField.PHONENUMBER);
TextField URL =
new TextField("URL", "", 64,  TextField.URL);
TextField DECIMAL =
new TextField("DECIMAL", "", 64,  TextField.DECIMAL);
textForm.append(ANY);
textForm.append(EMAILADDR);
textForm.append(NUMERIC);
textForm.append(PHONENUMBER);
textForm.append(URL);
textForm.append(DECIMAL);
```

Gambar: TextField Items