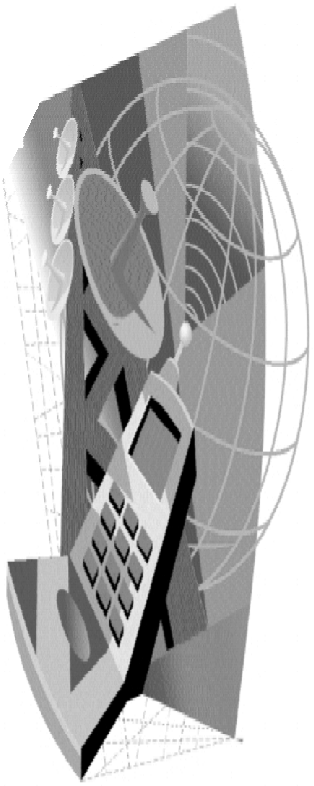
Pemrograman Aplikasi Wireless dengan Java/J2ME



Java 2 Standard Edition
- Pengantar Java
- Java dasar
- Pemrograman berorientasi objek dalam Java
- Java lanjut
- I/O Stream

Java 2 Micro Edition
- Pengantar J2ME
- Pengembangan aplikasi dengan J2ME
- MIDlet dasar
- Event handling
- User interface
- Pemrograman database dengan RMS
- Pemrograman client server dengan socket
- Installasi aplikasi pada perangkat wireless

Pemrograman Aplikasi Wireless dengan Java/J2ME

Versi:1.0

Disusun oleh:
Puji Hartono, ST <puji@perpustakaan-islam.com>

DAFTAR ISI

Bagian 1

J2SE

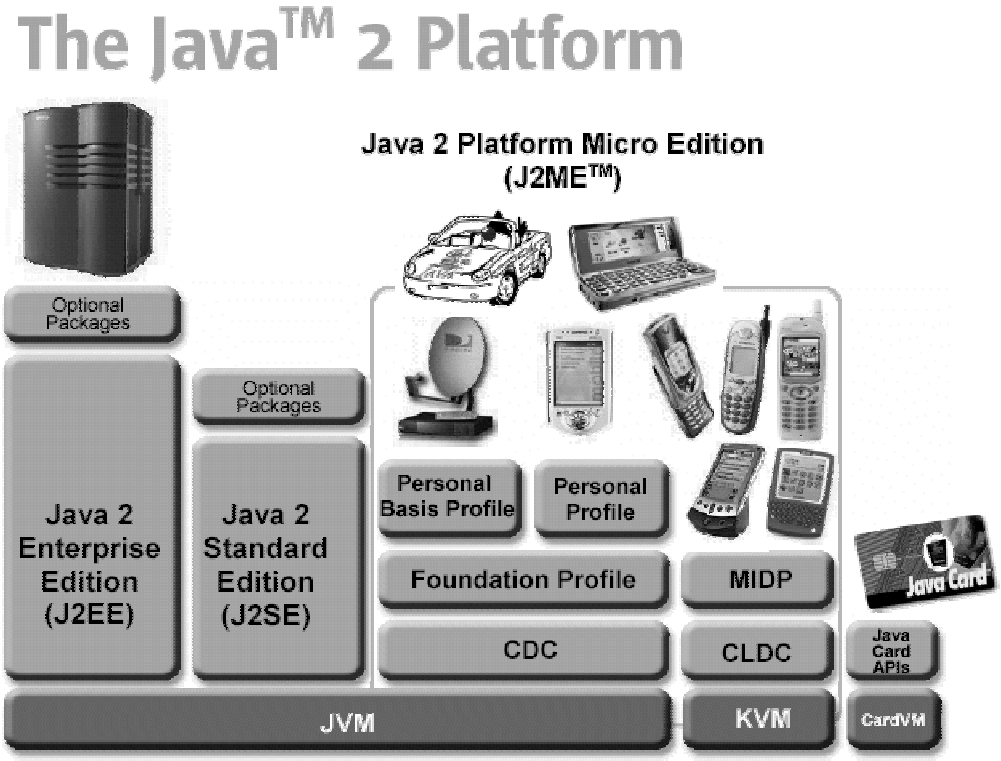
- Pengantar Java
- Java dasar
- Pemrograman berorientasi objek dalam Java
- Java lanjut
- I/O Stream

1

Pengantar JAVA

Java merupakan bahasa pemrograman "general purpose", Java bisa digunakan untuk pembuatan aplikasi web, database, grafis dan berbagai aplikasi lainnya seperti bahasa pemrograman lainnya. Mengapa Java menjadi andalan bagi para programmer ? Selain dukungan API yang begitu lengkap, hasil program Java akan bisa dijalankan di semua platform selama disitu terdapat JVM (Java Virtual Machine). Kalau seorang programmer memilih Java sebagai bahasa untuk pengembangan aplikasinya, maka dia tidak perlu pusing memikirkan apakah sistem operasi usernya mendukung atau tidak sehingga Java mempunyai slogan *"write once, run every where"*. Memang kalau hanya sekedar program helloWord jika ditulis dengan bahasa C akan bisa dijalankan di berbagai platform, akan tetapi untuk untuk program-program yang sudah menyentuh operasi I/O, komunikasi network dan berbagai hal yang spesifik terhadap platform tertentu maka kebutuhan bahasa yang "*cross platform*" menjadi penting dan Java adalah solusinya.

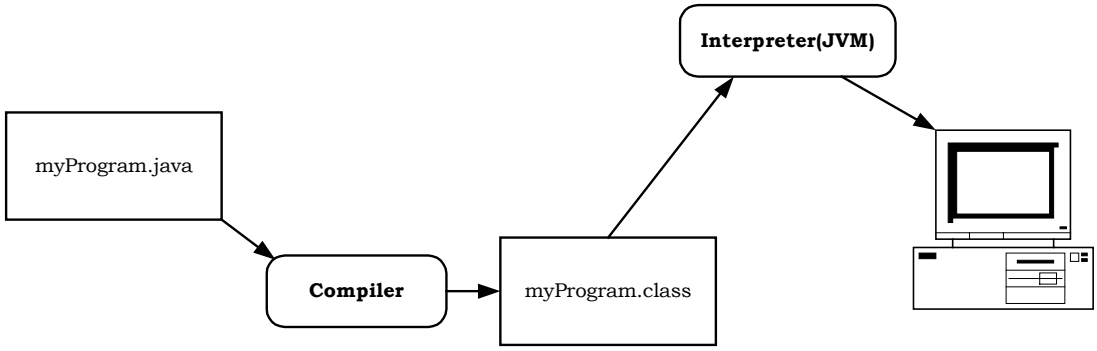
A. Keluarga Java



Gambar 1.1 Berbagai versi Java dilihat dari target platformnya

Sun Microsystem sebagai pengembang bahasa Java telah mengembangkan beberapa versi Java, seperti: J2EE (Java 2 Enterprise Edition), J2SE (Java 2 Standar Edition), J2ME (Java 2 Micro Edition), Java Card. J2EE dan J2SE dikembangkan untuk platform Server dan PC, J2ME dikembangkan untuk platform mobile device sedangkan Java Card untuk platform simcard. Pada dasarnya versi-versi tersebut sama, hanya penggunaan API yang berbeda sesuai dengan target platform yang akan digunakan dan kompatibilitas platform tersebut.

B. Pengembangan aplikasi dengan Java



Gambar 1.2 Alur pembuatan aplikasi dengan Java

Dalam pembuatan aplikasi dengan bahasa Java, pertama yang dilakukan adalah mengetik program dengan editor sembarang dan kemudian menyimpannya dengan ekstensi ".java". Setelah itu program dikompile sehingga mendapatkan class-classnya. File class inilah yang di*run* oleh interpreter (dalam hal ini JVM) menjadi aplikasi seperti program-program yang dibuat dengan bahasa-bahasa pemrograman lainnya.

2

Java Dasar

A. Kebutuhan software

Untuk membuat program Java diperlukan Java Development Kit yang berguna untuk mengkompile source kode sehingga didapatkan class-classnya. Java Development Kit bisa didownload di http://java.sun.com/

B. Memulai membuat program

Program Java bisa ditulis dengan sembarang editor, kode program tersebut kemudian dikompile sehingga dihasilkan file-file classnya. Class-class inilah yang akan dijalankan sebagai aplikasi program kita.

B.1. Menulis kode program

Penulisan kode program java bisa dilakukan dengan editor biasa seperti Notepad, Editplus, Ultraedit dan editor-editor lainnya. Setiap program Java disimpan dalam ekstensi ".java". Sebagai contoh program pertama kali, salinlah kode program berikut dan simpan dalam file "notHelloWorld.java"

```
1:  public class notHelloWorld
2:  {
3:          public static void main(String[] args)
4:          {
5:          System.out.println("Not Hello World");
6:          }
7:
8:  }
```

Kode program diatas akan menampilkan "Not Hello World"

B.2. Mengkompile program

Setelah kode program disimpan dalam ekstensi ".java", kode perlu dikompile, sintak kompilasi program Java adalah:

```
javac <options> <source files>
```

Sebagai contoh kompilasi program notHelloWorld yang berada di direktori D:\java adalah sebagai berikut

```
D:\java>javac notHelloWorld.java
```

Hasil kompilasi berupa file notHelloWorld.class seperti tampak pada output berikut.

```
D:\java>dir
 Volume in drive D is DATA-20 GB
 Volume Serial Number is 0930-1ED9
```

```
 Directory of D:\java

12/06/2003  05:24a      <DIR>          .
12/06/2003  05:24a      <DIR>          ..
12/06/2003  05:35a                125 notHelloWorld.java
12/06/2003  05:39a                435 notHelloWorld.class
               2 File(s)            560 bytes
               2 Dir(s)     813,809,664 bytes free
```

B.3. Menjalankan program

Adapun sintak untuk menjalankan program Java adalah:

```
java [-options] class [args...]
```

Sebagai contoh untuk menjalankan file "notHelloWorld.class" perintahnya adalah sebagai berikut

```
D:\java>java notHelloWorld
Not Hello World
```

B.4. Membuat dokumentasi program

Dalam pengembangan aplikasi dalam skala besar, seringkali sebuah aplikasi dikerjakan dalam tim yang terdiri dari beberapa orang. Class-class yang dibuat oleh kita mungkin saja akan digunakan orang lain sehingga kebutuhan dokumentasi yang tersusun rapi tentang class yang kita buat akan sangat penting. Dalam Java terdapat fasilitas untuk membuat dokumentasi tentang class-class yang kita buat. Keterangan-keterangan tentang class, properti, metodh yang kita buat akan sangat penting dalam pembuatan dokumentasi yang baik. Sintak untuk membuat dokumentasi class yang telah dibuat adalah sebagai berikut

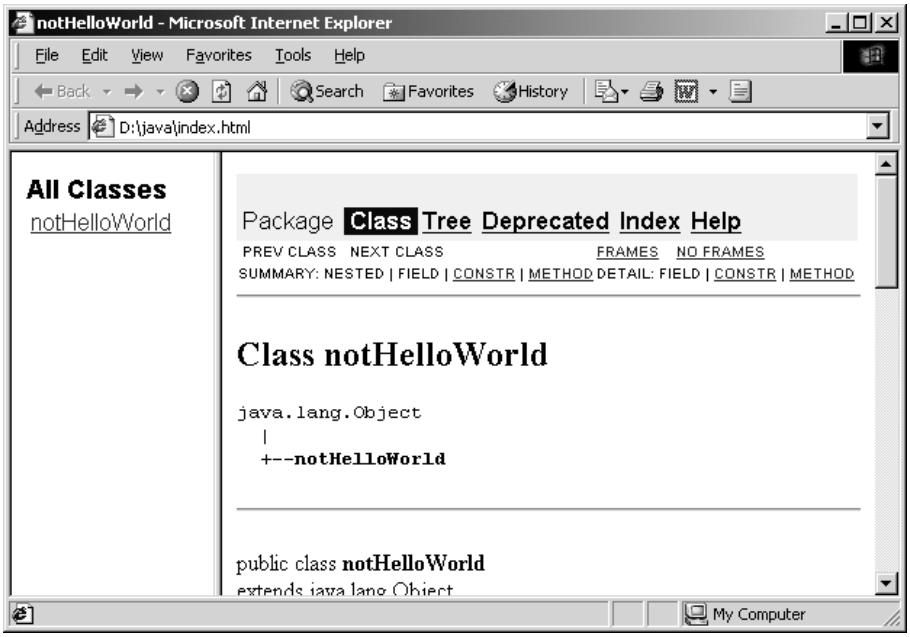
```
javadoc [options] [packagenames] [sourcefiles] [classnames] [@files]
```

Sebagai contoh pembuatan dokumentasi program notHelloWorld di atas yang telah dibuat adalah sebagai berikut

```
D:\java>javadoc notHelloWorld.java
Loading source file notHelloWorld.java...
Constructing Javadoc information...
Standard Doclet version 1.4.0

Generating constant-values.html...
Building tree for all the packages and classes...
Building index for all the packages and classes...
Generating overview-tree.html...
Generating index-all.html...
Generating deprecated-list.html...
Building index for all classes...
Generating allclasses-frame.html...
Generating allclasses-noframe.html...
Generating index.html...
Generating packages.html...
Generating notHelloWorld.html...
Generating package-list...
Generating help-doc.html...
Generating stylesheet.css...
```

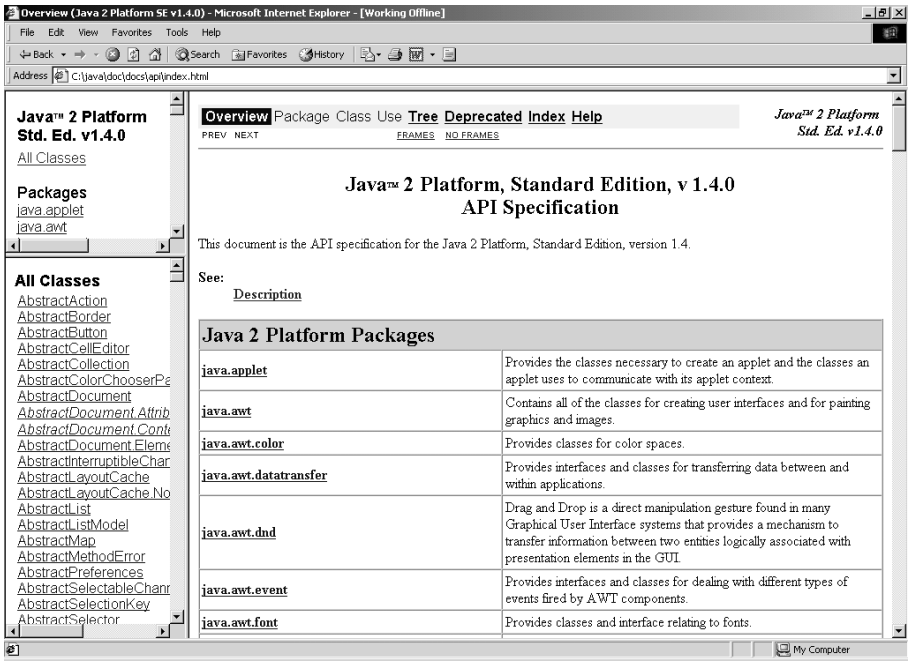
Sehingga akan dihasilkan dokumentasi program dalam format HTML seperti gambar berikut



Gambar 2.1. Dokumentasi Class

B.5. Membiasakan membaca manual dokumentasi API

Membaca manual merupakan kewajiban bagi programmer, sering kita lupa atau belum tahu penggunaan sebuah fungsi/class yang kita butuhkan dalam pengembangan software sehingga membaca manual menjadi sebuah keharusan, bahkan kemudian muncul istilah RTFM (Read The F*** in Manual).



Gambar 2.2. Dokumentasi API

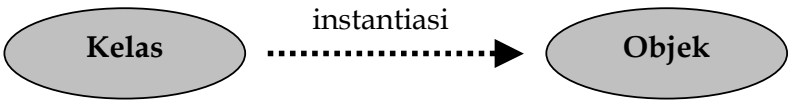
3

Pemrograman Berorientasi Objek dalam Java

Java merupakan bahasa pemrograman yang 100% berorientasi objek sehingga untuk membuat aplikasi berbasis Java, kita harus memahami konsep dan implementasi pemrograman berorientasi objek.

A. Kelas dan Objek

Dalam paradigma pemrograman berorientasi objek dikenal kelas dan objek. Kelas merupakan *blue print* dari objek-objek yang akan dibuat. Analogi kelas dan objek seperti *rancangan* model rumah dan *pembangunan* rumah-rumah, adapun proses pembuatan objek dari kelas dikenal dengan *instantiasi.*



Gambar 3.1. Pembuatan objek dari kelas dengan instantiasi

Sebagai contoh kita ambil kelas manusia. Kelas manusia mempunyai atribut : nama. Selain itu kelas manusia juga mempunyai metodh: tampilkanNama, kerja, makan. Kasus diatas diimplementasikan dalam bahasa Java sebagai berikut

```
 1:  public class manusia
 2:  {
 3:         public String nama;
 4:
 5:         public manusia(String n)
 6:         {
 7:         this.nama = n;
 8:         }
 9:
10:         public String tampilkanNama()
11:         {
12:         return nama;
13:         }
14:
15:
16:         public void makan()
17:         {
18:         System.out.println("Nyam... nyam... nyam...");
19:         }
20:
21:         public void kerja()
22:         {
```

```
23:         System.out.println("Kerja...kerjaaa...");
24:         }
25:   }
```

Adapun kode untuk menginstantiasi kelas manusia menjadi objek Andi yang mengimplementasikan metodh: tampilkanNama dan makan adalah sebagai berikut.

```
 1:  public class andi
 2:  {
 3:         public static void main(String arg[])
 4:         {
 5:         manusia andi= new manusia("Andi");
 6:         System.out.println("Nama= "+ andi.tampilkanNama());
 7:         andi.makan();
 8:         }
 9:  }
```

Hasil eksekusi terhadap *class* andi adalah sebagai berikut:

```
D:\java>java andi
Nama= Andi
Nyam... nyam... nyam...
```

B. Pewarisan

Salah satu kelebihan pemrograman berorientasi objek adalah penggunaan ulang kode-kode yang telah dibuat. Pewarisan adalah salah satu cara untuk menggunakan kode-kode yang telah dibuat sebelumnya.

Sebagai contoh kelas manusia diturunkan menjadi kelas programmer.

```
 1:  public class programmer extends manusia
 2:  {
 3:         public programmer(String n)
 4:         {
 5:                super(n);
 6:         }
 7:
 8:         public void kerja()
 9:         {
10:         System.out.println("Tak…Tak…Klik..");
11:         }
12:
13:         public void bersantai()
14:         {
15:         System.out.println("Game over, You lose...");
16:         }
17:   }
```

Badu adalah seorang programmer keturunan manusia, setelah dia makan lalu kerja dan terakhir dia bersantai dengan komputernya sehingga implementasi dengan kode javanya adalah

```
 1:  public class badu
 2:  {
 3:         public static void main(String arg[])
 4:         {
 5:         programmer badu= new programmer("badu");
 6:         System.out.println("Nama= "+ badu.tampilkanNama());
 7:         badu.makan();
 8:         badu.kerja();
 9:         badu.bersantai();
```

```
10:         }
11:  }
```

Sehingga *class* Badu jika dieksekusi akan menghasilkan

```
D:\java>java badu
Nama= badu
Nyam... nyam... nyam...
TakàTakàKlik..
Game over, You lose...
```

Setelah Objek Badu memberitahu namanya, dia makan dengan *metodh* warisan dari kelas manusia, kemudian dia kerja dengan metodh khusus kelas programmer dan terakhir dia bersantai juga dengan metodh khusus kelas programmer.

Kelas turunan akan mewariskan atribut-atribut dan metodh-metodh *parentclass*nya/*baseclass*, akan tetapi dia tidak mewarisi konstruktor-konstruktornya sehingga ketika andi makan maka dia makan dengan metodh dari *parentclass*nya (manusia).

Akan tetapi ketika dia kerja, dia kerja dengan metodh baru yang didefinisikan khusus pada kelas programmer ("Tak...Tak...Klik.. " bukan "Kerja....kerjaaa..."), inilah yang disebut dengan *metodh overriding.*

C. Polymorfisne

Salah satu pilar Pemrograman Berorientasi Objek yang lain adalah polymorfisme yaitu kemampuan beberapa objek bertipe sama bereaksi secara berbeda terhadap "pesan" yang sama.

Sebagai contoh kita tambah lagi turunan dari manusia yaitu kelas sopir. Kelas sopir diimplementasikan dalam java

```
 1:  public class sopir extends manusia
 2:  {
 3:         public sopir(String n)
 4:         {
 5:                super(n);
 6:         }
 7:
 8:         public void kerja()
 9:         {
10:         System.out.println("Ngung... Ngung... Ngung...Ciiit..");
11:         }
12:  }
```

Dedi adalah seorang sopir keturunan manusia, untuk menginstantiasi objek dedi ditunjukkan dalam kode berikut.

```
 1:  public class dedi
 2:  {
 3:         public static void main(String arg[])
 4:         {
 5:         sopir dedi= new sopir("Dedi");
 6:         System.out.println("Nama= "+ dedi.tampilkanNama());
 7:         dedi.makan();
 8:         dedi.kerja();
 9:         }
10:  }
```

Kemudian Badu sang programmer dan Dedi sang sopir diperintahkan untuk bekerja, apa reaksinya?

Untuk melihat reaksi masing-masing, perhatikan kode java berikut!

```
 1:  public class pekerja
 2:  {
 3:         public static void main(String args[])
 4:         {
 5:         manusia[] profesi= new manusia[2];
 6:         profesi[0]=new programmer("Andii");
 7:         profesi[1]=new sopir("Dedi");
 8:
 9:         for (int i=0; i<2; i++)
10:         {
11:                profesi[i].kerja();
12:         }
13:         }
14:  }
```

Hasil eksekusi para pekerja adalah sebagai berikut:

```
D:\java>java pekerja
Tak.. Tak.. Klik..
Ngung... Ngung... Ngung...Ciiit..
```

Dari contoh di atas terlihat bahwa dengan “pesan” yang sama akan menghasilkan reaksi yang berbeda tergantung implementasi metodhnya.

D. Pengkapsulan

Pilar terakhir dari 3 pilar Pemrograman Berorientasi Objek adalah pengkapsulan, dimana pengembang software dapat menyembunyikan detail suatu objek.

Hak akses public memungkinkan semua kelas mengaksesnya, hak akses protected hanya diberikan kepada kelasnya sendiri dan turunannya, serta kelas-kelas dalam satu paket. sedangkan private hanya boleh diakses oleh kelasnya sendiri.

E. Interface

Untuk membuat suatu kelas dapat kita turunkan dengan pewarisan field-field dan metodh pada base classnya. Bagaimana kita membuat kelas yang menurunkan sifat dari beberapa base class? misalkan kita akan membuat kelas superman yang dia bisa membuat program layaknya programmer, dia juga ahli menggunakan senjata layaknya tentara, bahkan dia bisa terbang seperti elang (keturunan binatang)? Caranya adalah dengan pewarisan ganda. Dalam Java tidak dikenal pewarisan ganda, sehingga digunakan interface.

Contoh pewarisan ganda yang tidak benar dalam Java

```
1:  public class superman extends programmer,kambing,sapi
2:   {
3:   }
```

Perhatikan contoh berikut:

```
1:  interface programmer {
2:    void memrogram();
3:  }
4:
5:  interface tentara {
6:    void menembak();
7:  }
```

```
 8:
 9:  interface burung {
10:    void terbang();
11:    void buangKotoran();
12:
13:  }
14:
15:  class superman implements programmer,tentara,burung {
16:
17:  public void memrogram(){};
18:  public void menembak(){};
19:  public void terbang(){};
20:  public void buangKotoran(){};
21:  }
```

Kemudian kelas superman diinstantiasi menjadi objek bernama bejo, contoh kode program javanya sebagai berikut:

```
 1:  public class bejo
 2:  {
 3:         public static void main(String arg[])
 4:         {
 5:         superman bj= new superman();
 6:
 7:         bj.memrogram();
 8:         {
 9:         System.out.println("Implementasi memrogram ...tak..tik");
10:         }
11:
12:         bj.menembak();
13:         {
14:         System.out.println("Implementasi menembak ...dor..dor");
15:        }
16:
17:         bj.terbang();
18:         {
19:         System.out.println("Implementasi terbang......Zap....");
20:        }
21:         }
22:  }
```

Dalam kode diatas bejo menentukan sendiri cara mengimplementasi beberapa metodh dari interface yang telah didefiniskan dalam kelas superman (misalkan cara memrogramnya bagaimana, cara menembaknya bagaimana dan cara terbangnya seperti apa), selain itu bejo tidak berminat untuk mengimplementasikan metodh buangKotoran dari interface burung (misalnya karena burung biasa membuang kotoran di sembarang tempat).

Jadi interface dapat dianalogikan seperti menandatangani kontrak kerja, misalnya sebagai dosen dia wajib mengajar, membuat soal ujian dsb, akan tetapi cara mengajarnya dan membuat soalnya dilakukan terserah masing-masing dosen (tidak ditentukan dalam kontrak kerja)

4

Java Lanjut

A. Pemaketan kelas-kelas

Ketika kelas-kelas yang dibuat semakin banyak dan semakin banyak, hal ini akan membuat struktur program menjadi rumit kalau tidak dikelola dengan baik. Untuk itu kelas-kelas disimpan dalam paket-paket tertentu, misalkan kelas programmerC, programmerJava, programmerPHP berada/disimpan dalam paket programmer. Sementara marinir, kopasus, paskhas berada dalam paket tentara. Keuntungan pengaturan program dalam nama paket-paket adalah:

1. Terhindar dari konflik penamaan. Misalkan saja dalam membuat program kita menggunakan/mengimport kelas dari luar yang dibuat oleh programmer lain, sehingga mungkin saja dalam penamaan kelas terjadi persamaan. Dengan menunjukkan nama lengkap paket/kelasnya maka tidak akan terjadi konflik penamaan
2. Teratur. Misalkan paket dosen terdapat kelas dosenPBO, dosenKalkulus, dosenEtika. Paket tentara terdapat marinir, kopasus, paskhas. Dengan struktur demikian, maka akan mempermudah ketika ketika kita akan menggunakan/mengimport kelas, misalkan saja kita akan mengimport kelas kopasus, tentunya kita mencari dalam paket tentara-bukan paket dosen.

Sebagai contoh kita akan membuat 2 paket: paket programmer dan paket tentara. Paket programmer terdiri dari kelas programmerC dan programmerJava. Sedangkan paket tentara terdiri dari kelas kopasus dan marinir.

```
 1:  /* Disimpan dalam "programmer/programmerC.java" */
 2:
 3:  package programmer;
 4:
 5:  public class programmerC
 6:  {
 7:         public programmerC()
 8:         {
 9:         }
10:
11:         public void kerja()
12:         {
13:         System.out.println("Implementasi metodh kerja Programmer C ..");
14:         }
15:
16:         public void bersantai()
17:         {
18:         System.out.println("Implementasi metodh bersantai Programmer .. ");
19:         }
20:  }
```

```
 1:  /* Disimpan dalam file "programmer/programmerJava.java" */
 2:
 3:  package programmer;
 4:
 5:  public class programmerJava
 6:  {
 7:         public programmerJava()
 8:         {
 9:         }
10:
11:         public void kerja()
12:         {
13:         System.out.println("Implementasi metodh kerja Programmer Java ..");
14:         }
15:
16:         public void bersantai()
17:         {
18:         System.out.println("Implementasi metodh bersantai Programmer Java.. ");
19:         }
20:  }
```

Sedangkan paket tentara terdapat kelas kopasus dan marinir.

```
 1:  /* Disimpan dalam file "tentara/kopasus.java" */
 2:
 3:  package tentara;
 4:
 5:  public class kopasus
 6:  {
 7:         public kopasus()
 8:         {
 9:         }
10:
11:         public void kerja()
12:         {
13:         System.out.println("Implementasi metodh kerja kopasus ....");
14:         }
15:
16:         public void bersantai()
17:         {
18:         System.out.println("Implementasi metodh bersantai kopasus");
19:         }
20:  }
```

```
 1:  /* Disimpan dalam file "tentara/marinir.java" */
 2:
 3:  package tentara;
 4:
 5:  public class marinir
 6:  {
 7:         public marinir()
 8:         {
 9:         }
10:
11:         public void kerja()
12:         {
13:         System.out.println("Implementasi metodh kerja marinir ..");
14:         }
15:
16:         public void bersantai()
17:         {
18:         System.out.println("Implementasi metodh bersantai marinir ..");
19:         }
20:  }
```

B. Mengimport kelas

Untuk dapat mengimport kelas digunakan keyword import [nama paketnya]. Sebagai contoh instantiasi kelas programmerJava dalam paket programmer menjadi objek ahmed.

```
 1:  /* Disimpan dalam file "ahmed.java" */
 2:
 3:  import programmer.programmerC;
 4:
 5:  class ahmed
 6:  {
 7:         public static void main(String arg[])
 8:         {
 9:         programmerC ahmed= new programmerC();
10:         ahmed.kerja();
11:         }
12:  }
```

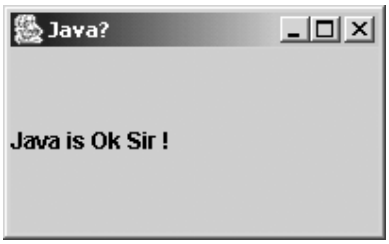
sehingga hasil eksekusinya adalah:

```
Implementasi metodh kerja Programmer C ....
```

Banyak sekali kelas-kelas yang telah dibuat oleh **Sun Microsystem** yang dapat kita gunakan, misalkan untuk membuat windows bisa digunakan/import paket awt dan swing.

```
 1:  /* Disimpan dalam file "JavaOk.java" */
 2:
 3:  import javax.swing.*;
 4:
 5:  public class JavaOk {
 6:
 7:  public static void main(String[] args) {
 8:          JFrame frame = new JFrame("Java?");
 9:          final JLabel label = new JLabel("Java is Ok Sir");
10:          frame.getContentPane().add(label);
11:
12:          frame.pack();
13:          frame.setVisible(true);
14:      }
15:  }
```

yang akan menampilkan hasi sebagai berikut:



Gambar 4.1. Aplikasi GUI dengan swing

5
I/O Stream

A. Membaca isi file

Java menyediakan paket java.io yang berisi kelas-kelas untuk input-output data. Contoh berikut digunakan untuk membaca isi file dan menampilkannya dalam layar. Pertama buatlah file "data.txt" kemudian isikan dengan nama-nama teman Anda. Program bacaFile akan membaca isi file data.txt perbaris dan menampilkannya ke layar.

Gambar 5.1. Isi file data.txt

```
 1:  import java.io.*;
 2:  import java.lang.*;
 3:
 4:  class bacaFile{
 5:  public static void main(String args[])
 6:         {
 7:         String namaFile="data.txt";
 8:         try{
 9:         String semuaTeks, isi;
10:         FileInputStream filein;
11:         BufferedReader datain;
12:
13:         filein=new FileInputStream(namaFile);
14:         datain=new BufferedReader(new InputStreamReader(filein));
15:         semuaTeks="";
16:
17:         while((isi=datain.readLine())!=null){
18:         System.out.println(isi);
19:         }
20:
21:         filein.close();
22:         }
23:         catch(Exception e)
24:         {
25:         System.out.println("Error:"+e);
26:         }
27:         }
28:  }
```

Jika dieksekusi, maka program bacaFile akan menghasilkan

```
D:\java>java bacaFile
Joko
Cahyo
Pandu
Iman
```

Program diatas akan membaca isi file dengan kelas FileInputStream byte demi byte pada tiap baris dan kemudian ditampung dengan BufferedReader dan akhirnya dicetak ke layar.

B. Menulis ke file

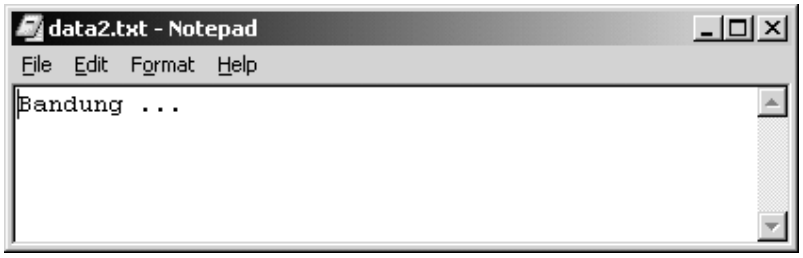
Program kedua tentang I/O stream akan menuliskan ke file "data2.txt". Kelas yang digunakan merupakan lawan pada program membaca file yakni FileOutputStream. Kode program untuk menuliskan ke file "data2.txt" adalah seperti berikut.

```
 1:  import java.io.*;
 2:
 3:  class tulisFile{
 4:  public static void main(String args[])
 5:  {
 6:         String namaFile="data2.txt";
 7:
 8:         try{
 9:         BufferedReader stdin=new BufferedReader(new InputStreamReader(System.in));
10:         String masukan, isi;
11:         System.out.println("Tulis input");
12:         masukan=stdin.readLine();
13:
14:         FileOutputStream fout;
15:         fout=new FileOutputStream(namaFile);
16:         new PrintStream(fout).print(masukan);
17:         new PrintStream(fout).println();
18:         fout.close();
19:         }
20:         catch(Exception e)
21:         {
22:         System.out.println("Error:"+e);
23:         }
24:         }
25:  }
```

Program diatas jika dieksekusi akan seperti :

```
D:\java>java tulisFile
Tulis input
Bandung ...
```

Hasil penulisan ke file "data2.txt" akan berisi string yang dimasukkan ketika menjalankan program tulisFile



Gambar 5.3. Isi file data2.txt

Bagian 2

J2ME

- Pengantar J2ME
- Pengembangan aplikasi dengan J2ME
- MIDlet dasar
- Event handling
- User interface
- Pemrograman database dengan RMS
- Pemrograman Client Server dengan Socket
- Installasi Aplikasi pada Perangkat Wireless
- Penutup

6

Pengantar J2ME

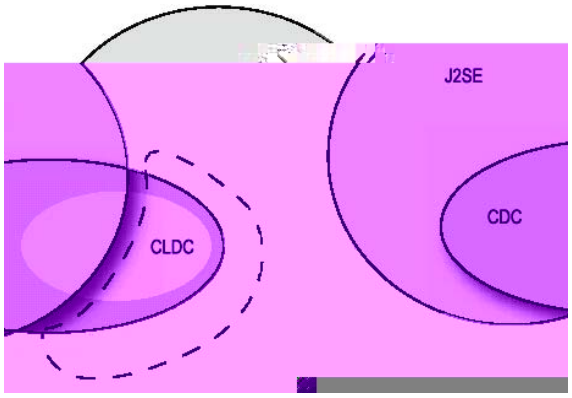
Perkembangan pemrograman aplikasi selama ini terfokus pada pengembangan aplikasi stand alone, kemudian berkembang lagi menjadi aplikasi client server serta aplikasi berbasis web. Dengan berkembangnya teknologi wireless seperti PDA dan handphone pada beberapa tahun terakhir ini, maka mulailah babak baru berupa aplikasi wireless. Dengan semboyan "*write once run everywhere*", teknologi Java dengan portabilitas yang tinggi, memungkinkan untuk jalan di segala sistem operasi selama disitu ada JVM (Java Virtual Machine) termasuk perangkat wireless dengan ukuran memori yang relatif kecil.

A. J2ME vs. WAP

Perbedaan utama antara J2ME dengan WAP adalah pada J2ME aplikasi terdapat di client dan bisa juga di server (client server) tidak seperti WAP yang seluruh aplikasi terdapat di server (di client dalam WAP hanya terdapat WAP browser), sehingga dalam J2ME bisa dibuat aplikasi *standalone* tanpa tergantung koneksi ke operator seluler seperti pada WAP yang memerlukan koneksi ke operator seluler. Dalam J2ME juga terdapat database lokal yakni RMS (Record Management System) yang berguna untuk menyimpan data.

B. API pada J2ME

Seperti dijelaskan pada gambar 1.1 dibagian I, J2ME terbagi menjadi 2: CDC dan CLDC. Pada CLDC(Connected Limited Device Configuration) umumnya untuk aplikasi java pada handphone seperti produk-produk Siemens, Nokia, Motorola dll. Sedangkan CDC(Connected Device Configuration) umumnya digunakan pada perangkat dengan memori setidaknya 2 Mb. Subset API pada CLDC dijelaskan pada gambar di bawah ini.



Gambar 6.1. API pada J2ME CLDC

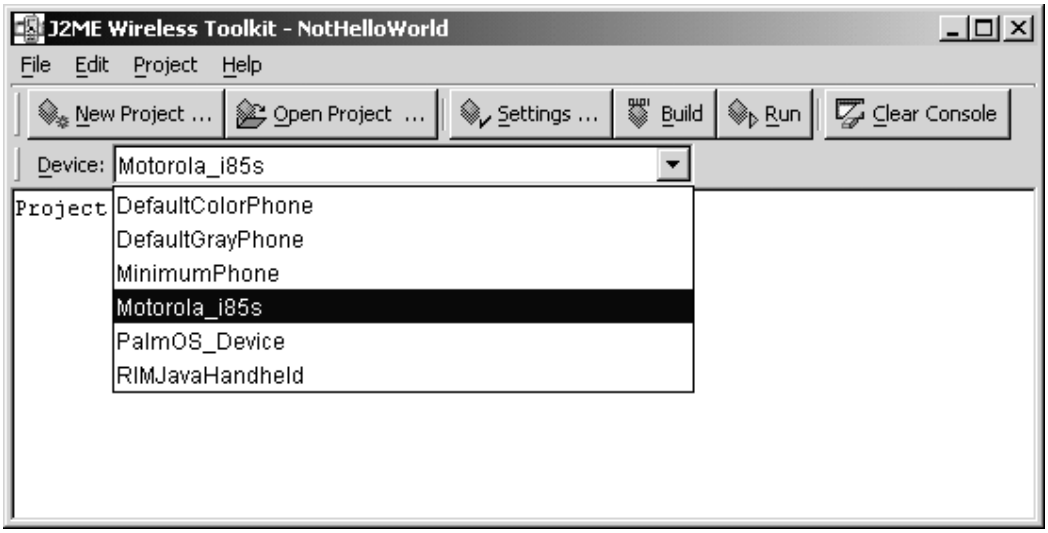
Salah satu kelemahan CLDC yang merisaukan programmer adalah tidak adanya dukungan *floating point* sehingga kreatifitas programmer sangat diperlukan. Pembahasan buku ini mengacu pada J2ME configuration CLDC.

7

Pengembangan Aplikasi Dengan J2ME

A. Perangkat pengembangan

Selain JDK dalam pengembangan aplikasi dengan J2ME diperlukan juga software tambahan yakni J2ME Wireless Toolkit. Dalam Toolkit ini terdapat API J2ME dan emulator beberapa perangkat wireless. Toolkit ini berfungsi untuk mengkompile dan mensimulasikan aplikasi pada emulator.



Gambar 7.1 J2ME Wireless Toolkit

Beberapa menu fungsi utama J2ME Wireless Toolkit antara lain:

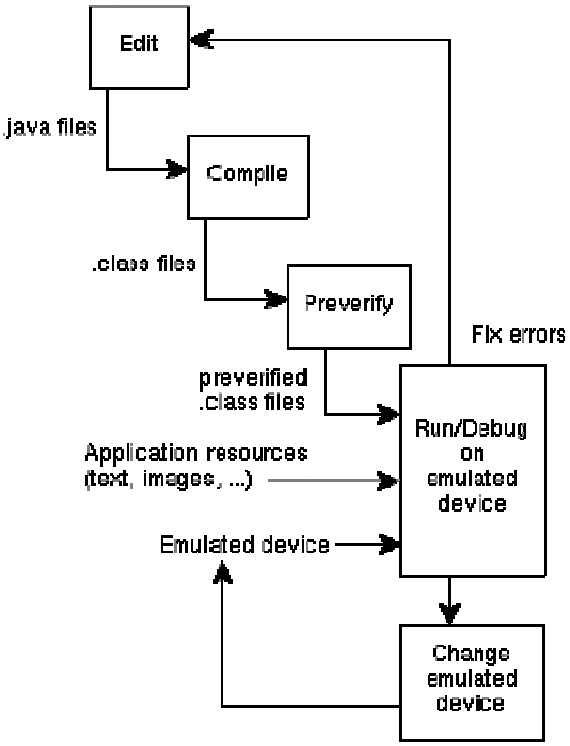
- **File → New Project**, tombol ini digunakan untuk memulai membuat project
- **File → Open Project**, tombol ini digunakan untuk membuka project yang pernah dibuat
- **Project → Build**, tombol ini digunakan untuk mengkompile source kode yang terdapat di dalam direktori /apps/nama_project/src/
- **Project → Run**, tombol ini digunakan untuk menjalankan aplikasi pada emulator sesuai pilihan
- **Project → Package**, menu ini untuk membundel aplikasi yang telah dibuat. Menu ini digunakan terakhir kali setelah semua tahapan pengembangan selesai
- **Project → Setting**, tombol ini digunakan untuk menyeting beberapa parameter aplikasi yang dibuat

Sedangkan keterangan direktori-direktori yang ada didalam J2ME Wireless Toolkit adalah sebagai berikut:

File/Direktori	Keterangan
appdb\	Direktori berisi file-file database, seperti file-file RMS file
apps\	Direktori yang berisi demo aplikasi dan aplikasi-aplikasi yang dibuat dengan KToolbar
bin\	Batch file dan file-file executable tool
docs\	Direktori yang berisi dokumtasi API
lib\midpapi.zip	Arsip kelas-kelas API CLDC dan MIDP. File-file ini digunakan pada proses kompilasi dan preverification
sessions\	Direktori default yang berisi profiling, monitoring memori dan session file pada monitoring network
wktlib\devices\	Direktori yang berisi file-file properti device-device yang digunakan pada emulator

B. Tahapan pengembangan

Pengembangan aplikasi dengan J2ME memerlukan tahapan-tahapan seperti yang dijelaskan dalam gambar dibawah ini



Gambar 7.2 Tahapan pengembangan aplikasi dengan J2ME

Setelah semua fungsional aplikasi sesuai dengan yang direncanakan, aplikasi perlu disimulasikan ke beberapa emulator sesuai dengan target usernya. Secara umum semua perangkat wireless yang telah mensupport Java(J2ME) maka aplikasi yang kita buat akan bisa dijalankan di perangkat tersebut, namun terkadang untuk tiap devais target

mempunyai kekhususan seperti lebar dan tinggi layar/display, sehingga mungkin kita perlu membuat beberapa versi distribusi aplikasi kita yang disesuaikan dengan devais targetnya.

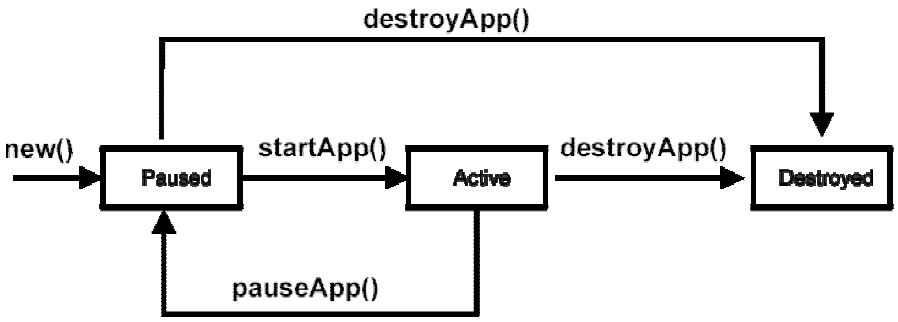
Gambar 7.3 Aplikasi J2ME yang dijalankan pada emulator

8

MIDlet Dasar

A. Siklus hidup sebuah MIDlet

MIDlet (Mobile Information Device Applet) mempunyai 3 kemungkinan kondisi: pause, active dan destroyed. Ketika MIDlet mulai dijalankan, maka MIDlet berada pada kondisi aktif. Jika terdapat interupsi seperti adanya panggilan pada ponsel maka MIDlet berada pada kondisi pause. Dan ketika selesai menjalankan MIDlet maka diperlukan metodh untuk membunuh MIDlet dengan *destroyApp().* Ketiga kondisi ini digambarkan pada diagram di bawah ini:



Gambar 8.1 Siklus hidup sebuah MIDlet

B. MIDlet sederhana

Untuk memulai belajar J2ME ini kita ambil contoh program NotHelloWorld yang akan menampilkan tulisan "Not Hello World" melalui TextBox. Pertama yang perlu dilakukan adalah membuat project baru dan memberi nama NotHelloWorld, kemudian salinlah kode berikut dan simpan ke dalam file "NotHelloWorld.java" di dalam direktori /apps/NotHelloWorld/src/.

```
 1:  import javax.microedition.MIDlet.*;
 2:  import javax.microedition.lcdui.*;
 3:
 4:  public class NotHelloWorld extends MIDlet {
 5:
 6:          private Display display;
 7:
 8:          public NotHelloWorld()
 9:                  {
10:                  display = Display.getDisplay(this);
11:                  }
12:
13:          public void startApp()
14:                  {
```

```
15:                TextBox t = new TextBox("Not","Not Hello World",256,0);
16:                display.setCurrent(t);
17:                }
18:
19:         public void pauseApp()
20:                {
21:                }
22:
23:         public void destroyApp(boolean unconditional)
24:                {
25:                }
26:   }
```

Contoh kode diatas akan menampilkan sebuah TextBoox yang diberi label Not dan didalamnya terdapat kata “Not Hello World”. Ketika program dipanggil maka state MIDlet berada pada active dengan memanggil metodh startApp() dan pada state inilah sebuah TextBoox dibuat (instantiasi) untuk kemudian menampilkan string Not Hello World. Setiap MIDlet harus mempunyai minimal 3 metodh: startApp(), pauseApp() dan destroyApp(boolean unconditional). Output program diatas seperti pada gambar di bawah ini

Gambar 8.2. MIDlet Dasar, NotHelloWorld

9

Event Handling

Event berguna untuk menangani interaksi user dengan program, misalnya user memilih sebuah menu dalam aplikasi MIDlet.Untuk menangani event perlu mengimplementasikan interface CommandListener dan atau ItemListener. CommandListener berfungsi untuk menangani jika user memilih Command tertentu sedangkan ItemListener berfungsi untuk menangani jika user mengubah nilai seperti misalnya mengubah pilihan pada ChoiceGroup.

Untuk memberikan gambaran sebuah event bekerja, perhatikan contoh berikut.

```
 1:  import javax.microedition.MIDlet.*;
 2:  import javax.microedition.lcdui.*;
 3:
 4:  public class NotHelloWorld2 extends MIDlet implements CommandListener{
 5:
 6:         private Command cmdExit;
 7:         private Display display;
 8:
 9:         public NotHelloWorld2()
10:                {
11:                display = Display.getDisplay(this);
12:                cmdExit = new Command("Exit", Command.SCREEN,2);
13:                }
14:
15:         public void startApp()
16:                {
17:                TextBox t = new TextBox("Not ","Not Hello World",256,0);
18:                t.addCommand(cmdExit);
19:                t.setCommandListener(this);
20:
21:                display.setCurrent(t);
22:                }
23:
24:         public void pauseApp()
25:                {
26:                }
27:
28:         public void destroyApp(boolean unconditional)
29:                {
30:                }
31:
32:         public void commandAction(Command cmd,Displayable disp)
33:                {
34:                if(cmd==cmdExit)
35:                       {
36:                       destroyApp(false);
37:                       notifyDestroyed();
38:                       }
39:                }
40:  }
```

Pada contoh kedua ini, kita tambahkan sebuah *Command* untuk keluar dari aplikasi yang telah kita buat pada contoh 1. Ketika program pertama kali dipanggil, maka state berada pada Aktif, disini sebuah *TextBox* dan *Command* dibuat. Kemudian program menunggu respon dari user dengan mengimplementasikan *CommandListener*. Ketika user menekan Command "cmdExit" (baris 32-39), maka program memanggil metodh(destroyApp) untuk membunuh MIDlet.

Gambar 9.1. Penggunaan event

10

User Interface

Dalam sebuah aplikasi seringkail membutuhkan user interface seperti misalnya dibutuhkan input pada aplikasi yang akan dibangun. Contoh di bawah ini mendemonstrasikan input dengan sebuah TextField dan output dengan StringItem.

```
 1:  import javax.microedition.MIDlet.*;
 2:  import javax.microedition.lcdui.*;
 3:
 4:  public class converter extends MIDlet implements CommandListener{
 5:
 6:         private Display display;
 7:         private Command cmdExit;
 8:         private Command cmdConvert;
 9:         private Command cmdOK;
10:         private Form form1;
11:         private Form form2;
12:         private TextField input;
13:         private StringItem hasil;
14:
15:         public void startApp()
16:                {
17:                display      = Display.getDisplay(this);
18:                cmdConvert   = new Command("Convert",Command.SCREEN,1);
19:                cmdExit      = new Command("Exit",Command.SCREEN,1);
20:                cmdOK        = new Command("OK",Command.SCREEN,1);
21:
22:                form1        = new Form("Converter");
23:                input        = new TextField("Input:",null,256,TextField.ANY);
24:
25:                form1.addCommand(cmdConvert);
26:                form1.addCommand(cmdExit);
27:                form1.setCommandListener(this);
28:                form1.append(input);
29:
30:                form2        = new Form("Hasil");
31:                hasil        = new StringItem(null,null);
32:                form2.append(hasil);
33:                form2.addCommand(cmdOK);
34:                form2.setCommandListener(this);
35:
36:                display.setCurrent(form1);
37:                }
38:
39:         public void pauseApp()
40:                {
41:                }
42:
43:         public void destroyApp(boolean unconditional)
44:                {
45:                }
46:
47:         public void commandAction(Command cmd,Displayable disp)
48:                {
```

```
49:                String displayString       = null;
50:                {
51:                if(cmd==cmdExit)
52:                       {
53:                       destroyApp(false);
54:                       notifyDestroyed();
55:                       }
56:                else if(cmd==cmdConvert)
57:                       {
58:                       String s            = input.getString();
59:                       displayString= convert(s);
60:                       hasil.setText(displayString);
61:                       display.setCurrent(form2);
62:                       }
63:                else if(cmd==cmdOK)
64:                       {
65:                       input.setString(null);
66:                       display.setCurrent(form1);
67:                       }
68:                }
69:                }
70:
71:         private String convert(String s)
72:
73:                {
74:                       return s.toUpperCase();
75:                }
76:  }
```

Contoh diatas mendemonstrasikan penggunaan user interface dalam hal ini TextField yang digunakan sebagai media input dan input string tersebut diproses dalam fungsi untuk mengkonversi menjadi huruf kapital yakni dengan metodh String.toUpperCase() untuk kemudian dikeluarkan melalui StringItem. Dalam pengoperasian fungsi-fungsi diatas tentunya dilengkapi dengan Command-Command yang mengimplementasikan *CommandListener*.Output program diatas tampil pada gambar di bawah ini.

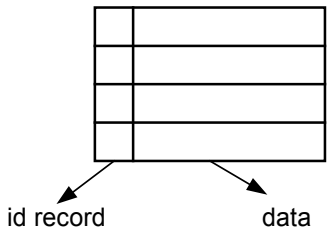
Gambar 10.1. Input konverter karakter

Gambar 10.2. Output konverter karakter

11

Pemrograman Database dengan RMS

J2ME CLDC menyediakan paket RMS (Record Management System) untuk penanganan database. Sedangkan dalam penulisan dan pembacaan data digunakan stream. Struktur RMS diilustrasikan dalam gambar berikut.



Gambar 11.1. Struktur RMS

Id record dapat dianalogikan sebagai *primary key* dan selebihnya adalah data. Untuk mempejelas penggunaan RMS, perhatikan program database kota, yakni menambahkan nama kota ke dalam database RMS.

```
 1:  import javax.microedition.rms.*;
 2:  import java.io.*;
 3:  import javax.microedition.MIDlet.*;
 4:  import javax.microedition.lcdui.*;
 5:
 6:  public class tambahData extends MIDlet implements CommandListener
 7:  {
 8:
 9:       private Display display;
10:
11:          private Command cmdExit =  new Command("Exit",Command.EXIT,2);
12:          private Command cmdAdd  =  new Command("Tambahkan",Command.OK,1);
13:          private Form formTambahKota;
14:          private TextField tKota;
15:          private RecordStore myDb;
16:
17:       public  tambahData()
18:                {
19:                }
20:
21:       public void startApp()
22:                {
```

```
23:                display    = Display.getDisplay(this);
24:               bukaDb();
25:               }
26:
27:       public void tambahData()
28:               {
29:               formTambahKota  = new Form("Tambah Kota ");
30:               tKota           = new TextField("",null,30,TextField.ANY);
31:
32:               formTambahKota.append(tKota);
33:               formTambahKota.addCommand(cmdAdd);
34:               formTambahKota.addCommand(cmdExit);
35:               formTambahKota.setCommandListener(this);
36:               display.setCurrent(formTambahKota);
37:               }
38:
39:          public void tambahkan()
40:                {
41:                try{
42:
43:                   String kota                    = tKota.getString();
44:
45:                   ByteArrayOutputStream baos     = new ByteArrayOutputStream();
46:                   DataOutputStream dos           = new DataOutputStream(baos);
47:                   dos.writeUTF(tKota.getString());
48:
49:                   byte[] b = baos.toByteArray();
50:                   myDb.addRecord(b, 0, b.length);
51:                   display.setCurrent(formTambahKota);
52:                   }
53:                catch(IOException e)
54:                      {
55:                      }
56:
57:                catch(RecordStoreException e)
58:                      {
59:                      }
60:                }
61:
62:          public void bukaDb()
63:                {
64:                try{
65:                   myDb = RecordStore.openRecordStore("kota",true);
66:                   tambahData();
67:                   }
68:
69:                   catch(RecordStoreNotFoundException e)
70:                   {
71:                   }
72:                   catch(RecordStoreException e)
73:                   {
74:                   }
75:                 }
76:
77:          public void Exit(){
78:
79:                try
80:                   {
81:                   myDb.closeRecordStore();
82:                   this.destroyApp(true);
83:                   }
84:
85:                catch(RecordStoreNotOpenException e)
86:                   {
87:                   }
88:
89:                catch(RecordStoreException e)
90:                   {
91:                   }
92:
```

```
 93:                catch(NullPointerException e)
 94:                    {
 95:                     this.destroyApp(true);
 96:                    }
 97:          }
 98:
 99:          public void commandAction(Command c, Displayable d)
100:                 {
101:                       String lbl = c.getLabel();
102:                       if(lbl.equals("Exit"))
103:                       {
104:                       Exit();
105:                       }
106:
107:                 else if(lbl.equals("Tambahkan"))
108:                       {
109:                       tambahkan();
110:                       }
111:                  }
112:
113:          public void pauseApp()
114:                 {
115:                  }
116:
117:          public void destroyApp(boolean unconditional)
118:                 {
119:                 notifyDestroyed();
120:                 }
121:   }
```

Dalam program penambahan data diatas, awalnya program memanggil *bukaDb()* yang disitu terdapat inisialisasi database kota

```
myDb = RecordStore.openRecordStore("kota",true);
```

Jika database "kota" belum ada, maka RMS akan membuatkan database "kota". Kemudian program memanggil *tambahData()* sebagai media input berupa textField, sehingga program akan menuju pengisian nama kota seperti pada gambar berikut.

Gambar 11.2. Form penambahan kota

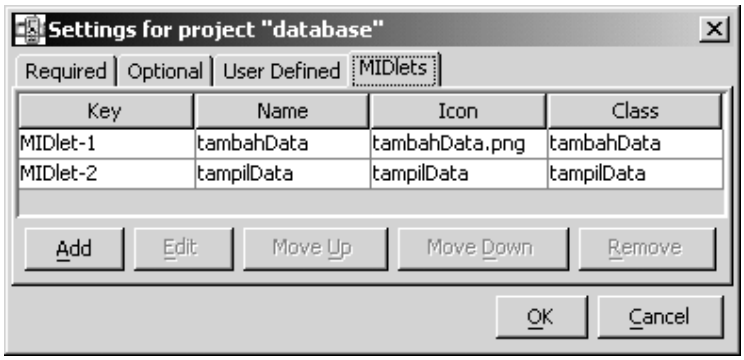
Ketika user menekan Command Tambahkan(cmdAdd), maka proses penambahan data dilakukan dengan stream.

```
String kota                = tKota.getString();

ByteArrayOutputStream baos  = new ByteArrayOutputStream();
DataOutputStream dos        = new DataOutputStream(baos);
dos.writeUTF(tKota.getString());

byte[] b = baos.toByteArray();
myDb.addRecord(b, 0, b.length);
```

Untuk menampilkan data, kita tambahkan program untuk menambahkan data. Untuk mempermudah, kedua MIDlet ini digabung dalam satu project dengan menambahkan MIDlet pada setting projectnya.



Gambar 11.3. Menambahkan MIDlet dalam project

```
 1:  import javax.microedition.rms.*;
 2:  import java.io.*;
 3:  import javax.microedition.MIDlet.*;
 4:  import javax.microedition.lcdui.*;
 5:
 6:  public class tampilData extends MIDlet
 7:  implements CommandListener
 8:  {
 9:       private Display display;
10:          private Command cmdSelect = new Command("Select",Command.EXIT,1);
11:          private Command cmdExit   = new Command("Exit",Command.EXIT,1);
12:
13:          private Form formListKota;
14:          private TextField t2;
15:          private RecordStore rmsDbKota;
16:
17:          private static class Record
18:                {
19:                String kota;
20:                }
21:
22:           public  tampilData()
23:                {
24:                }
25:
26:         public void startApp()
27:                {
28:                display    = Display.getDisplay(this);
29:                bukaDbTampil();
30:                }
31:
32:           public void tampilkanKota()
33:                {
34:                try{
35:                    formListKota   = new Form("Kota");
36:                   ChoiceGroup ChoiceGrp = new ChoiceGroup("",Choice.EXCLUSIVE);
37:                   formListKota.addCommand(cmdExit);
38:                   formListKota.addCommand(cmdSelect);
39:                   formListKota.setCommandListener(this);
40:                   formListKota.append(ChoiceGrp);
41:
42:                   byte[] recData = new byte[150];
43:                   ByteArrayInputStream bais = new ByteArrayInputStream(recData);
44:                   DataInputStream dis = new DataInputStream(bais);
```

```
 45:
 46:               RecordEnumeration enum = rmsDbKota.enumerateRecords(null,null, false);
 47:               Record r=new Record();
 48:
 49:              while(enum.hasNextElement())
 50:               {
 51:                 int recId = enum.nextRecordId();
 52:
 53:                 bais = new ByteArrayInputStream(rmsDbKota.getRecord(recId));
 54:                 dis = new DataInputStream(bais);
 55:                 r.kota = dis.readUTF();
 56:
 57:                 ChoiceGrp.append(r.kota,null);
 58:                  }
 59:
 60:                  display.setCurrent(formListKota);
 61:               }
 62:               catch(IOException e)
 63:                      {
 64:                      }
 65:               catch(RecordStoreNotOpenException e)
 66:                      {
 67:                      }
 68:               catch(RecordStoreException e)
 69:                      {
 70:                      }
 71:        }
 72:
 73:  public void Kembali()
 74:        {
 75:        display.setCurrent(formListKota);
 76:        }
 77:
 78:  public void bukaDbTampil()
 79:     {
 80:     try{
 81:          rmsDbKota = RecordStore.openRecordStore("kota",true);
 82:          tampilkanKota();
 83:         }
 84:
 85:     catch(RecordStoreNotFoundException e)
 86:         {
 87:         }
 88:     catch(RecordStoreException e)
 89:         {
 90:         }
 91:      }
 92:
 93:  public void keluar()
 94:      {
 95:        try{
 96:          rmsDbKota.closeRecordStore();
 97:          this.destroyApp(true);
 98:           }
 99:        catch(RecordStoreNotOpenException e)
100:           {
101:           }
102:        catch(RecordStoreException e)
103:           {
104:           }
105:        catch(NullPointerException e)
106:           {
107:           this.destroyApp(true);
108:           }
109:       }
110:
111:  public void commandAction(Command c, Displayable d)
112:        {
113:         String lbl = c.getLabel();
114:         if(lbl.equals("Exit"))
```

```
115:             {
116:              keluar();
117:             }
118:          else if (lbl.equals("Select"))
119:             {
120:             }
121:          }
122:       public void pauseApp()
123:          {
124:          }
125:       public void destroyApp(boolean unconditional)
126:          {
127:           notifyDestroyed();
128:          }
129:  }
```

Program diatas akan menampilkan semua data dengan stream dan menampilkannya dalam choice group.

```
bais = new ByteArrayInputStream(rmsDbKota.getRecord(recId));
dis = new DataInputStream(bais);
r.kota = dis.readUTF();

ChoiceGrp.append(r.kota,null);
```

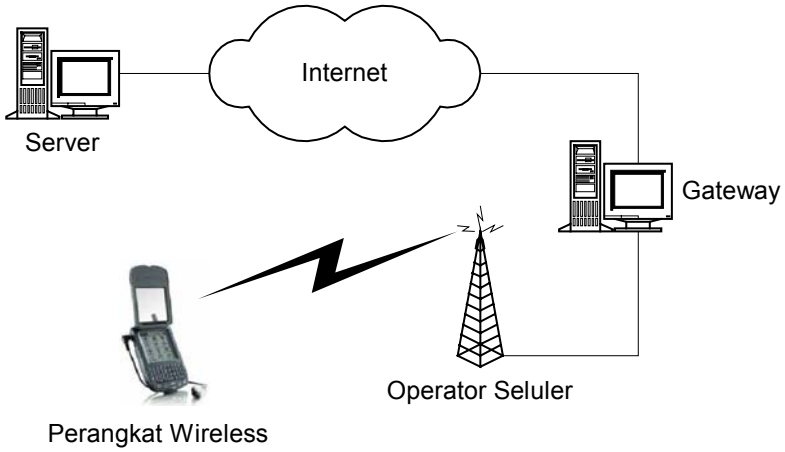
Output program diatas seperti pada gambar di bawah ini. Jika user menekan Command Exit maka program memannggil metodh untuk membunuh MIDlet

Gambar 11.4. Menampilkan data

12

Pemrograman Client Server dengan Socket

Tiba saatnya kita masuk ke pembahasan inti, pemrograman client server. Perangkat wireless dapat berkomunikasi dengan perangkat lainya apakah PC atau antar perangkat wireless dapat dengan berbagai macam, seperti dengan infra merah, bluetooth atau koneksi GPRS (General Packet Radio Service). Koneksitas antara server dengan perangkat wireless pada aplikasi yang akan kita buat dengan menggunakan GPRS diilustrasikan pada gambar berikut di bawah ini.



Gambar 12.1. Koneksi server dengan perangkat wireless dengan GPRS

Aplikasi yang akan kita buat berupa program untuk melihat status pengiriman pada sebuah perusahaan yang menerima delivery order. Program ditanam di server dan di client. Pada server digunakan MySQL sebagai databasenya dan JDBC sebagai koneksitasnya dengan aplikasi yang kita buat.

A. Sisi server

Pada sisi server kita membutuhkan database dalam hal ini kita gunakan MySQL. Database dan tabel yang kita pakai diberi nama “status”. Contoh query untuk menampilkan seluruh data pada tabel status adalah sebagai berikut.

```
mysql> select * from status;
+-------+---------------+----------+--------------------+
| id    | tanggal_datang | status   | tanggal_pengiriman |
+-------+---------------+----------+--------------------+
| 00001 | 2003-12-07     | terkirim | 2003-12-10         |
| 00002 | 2003-12-07     | pending  | 2003-12-11         |
| 00003 | 2004-01-07     | terkirim | 2004-01-09         |
| 00004 | 2004-01-08     | pending  | 2004-01-09         |
| 00005 | 2004-01-09     | pending  | 2004-01-11         |
+-------+---------------+----------+--------------------+
5 rows in set (0.00 sec)
```

Program di sisi server menggunakan socket untuk membuat program yang siap menerima koneksi dari client.

```
 1:  import java.io.*;
 2:  import java.net.*;
 3:  import java.sql.*;
 4:
 5:
 6:  public class OrderStatusServer {
 7:
 8:  private int port =2004;
 9:  private ServerSocket serverSocket;
10:
11:  static final String dbURL=
12:        "jdbc:mysql://localhost/status?"+
13:        "user=root&password=";
14:
15:  public OrderStatusServer()
16:        throws ClassNotFoundException
17:        {
18:        Class.forName("org.gjt.mm.mysql.Driver");
19:        }
20:
21:  public void acceptConnection()
22:        {
23:        try
24:              {
25:              serverSocket = new ServerSocket(2004);
26:              }
27:        catch(IOException e)
28:              {
29:              System.err.println("Gagal menginisialisasi Socket Server");
30:              e.printStackTrace();
31:              System.exit(0);
32:              }
33:        while(true)
34:              {
35:              try
36:                    {
37:                    Socket newConnection = serverSocket.accept();
38:                    System.out.println("koneksi diterima");
39:                    ServerThread st = new ServerThread(newConnection);
40:                    new Thread(st).start();
41:                    }
42:              catch(IOException ioe)
43:                    {
44:                    System.err.println("gagal menerima koneksi");
45:                    }
46:              }
47:        }
```

```
 48:
 49:        public static void main(String args[])
 50:              {
 51:              OrderStatusServer server =null;
 52:              try
 53:                    {
 54:                    server =new OrderStatusServer();
 55:                    }
 56:              catch
 57:                    (
 58:                    ClassNotFoundException e)
 59:                          {
 60:                          System.out.println("Gagal meload driver JDBC");
 61:                          e.printStackTrace();
 62:                          System.exit(1);
 63:                          }
 64:                    server.acceptConnection();
 65:                    }
 66:
 67:  class ServerThread implements Runnable
 68:        {
 69:        private Socket socket;
 70:        private DataInputStream datain;
 71:        private DataOutputStream dataout;
 72:
 73:        public ServerThread(Socket socket)
 74:              {
 75:              this.socket = socket;
 76:              }
 77:
 78:              public void run()
 79:              {
 80:              try
 81:              {
 82:              datain= new DataInputStream
 83:              (new BufferedInputStream(socket.getInputStream()));
 84:              dataout= new DataOutputStream
 85:              (new BufferedOutputStream(socket.getOutputStream()));
 86:              }
 87:              catch(IOException e)
 88:              {
 89:              return;
 90:              }
 91:
 92:              byte[] ba =new byte[6];
 93:              boolean conversationActive=true;
 94:              while(conversationActive)
 95:              {
 96:              String orderNumber=null;
 97:              try
 98:                    {
 99:                    datain.read(ba,0,6);
100:                    orderNumber=new String(ba);
101:                    if(orderNumber.toUpperCase().charAt(0)=='C')
102:                          {
103:                          conversationActive=false;
104:                          }
105:                    else
106:                          {
107:                          System.out.println("order Number="+orderNumber);
108:                          String status=getStatus(orderNumber);
109:                          System.out.println("status:"+status);
110:
```

```
111:        dataout.write(status.getBytes(),0,status.length());
112:                        dataout.write("\n".getBytes(),0,1);
113:                        dataout.flush();
114:                        }
115:                  }
116:                  catch(IOException ioe)
117:                  {
118:                  conversationActive=false;
119:                  }
120:            }
121:            try
122:            {
123:            System.out.println("closing socket");
124:            datain.close();
125:            dataout.close();
126:            socket.close();
127:            }
128:            catch(IOException e)
129:            {
130:            }
131:        }
132:
133:        private String getStatus(String orderNumber)
134:        {
135:        String status="Not on file";
136:        Connection conn=null;
137:        try
138:              {
139:              conn=DriverManager.getConnection(dbURL);
140:              Statement stmt=conn.createStatement();
141:              String query=
142:                    "SELECT status from status where id="+orderNumber;
143:              ResultSet rs =stmt.executeQuery(query);
144:              if(rs.next())
145:                    {
146:                    status=rs.getString(1);
147:                    }
148:              }
149:
150:              catch(SQLException e)
151:              {
152:              status="server error";
153:              }
154:
155:              finally
156:              {
157:              if (conn!= null)
158:                    {
159:                    try
160:                          {
161:                          conn.close();
162:                          }
163:                    catch (SQLException e)
164:                          {
165:                          }
166:                    }
167:              }
168:              return status;
169:        }
170:  }
171:  }
```

Setelah dikompile dan dieksekusi, ceklah apakah program di server sudah siap menerima koneksi, perhatikan contoh output netstat berikut pada sisi server!

```
C:\>netstat /na

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:21             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:42             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2004           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING
```

Di sisi server berfungsi:
- Membuka port untuk menerima koneksi
- Menerima koneksi dari client
- Membaca nomor order
- Dengan JDBC mengeksekusi perintah SQL untuk menampilkan status order
- Mengirim hasilnya ke client

B. Sisi client

Di sisi client dibuat program untuk mengakses server, kode program sebagai berikut

```
 1: import javax.microedition.midlet.*;
 2: import javax.microedition.lcdui.*;
 3:
 4:
 5: import javax.microedition.io.*;
 6: import java.io.*;
 7:
 8: public class OrderStatusClient extends MIDlet implements CommandListener
 9: {
10:       Display display;
11:       private boolean commandAvailable;
12:       private byte[] xmit;
13:       CommandThread commandThread;
14:
15:
16:       List menu;
17:       Form inputForm;
18:       Form outputForm;
19:       TextField orderNumber;
20:       StringItem orderStatus;
21:
22:       Command cmdBack;
23:       Command cmdExit;
24:       Command cmdOK;
25:
26:
27:       private static final String URL="socket://127.0.0.1:2004";
28:
29:       StreamConnection conn= null;
30:       InputStream is = null;
31:       OutputStream os = null;
32:
33:       public void startApp(){
34:             display = Display.getDisplay(this);
35:             inputForm=new Form ("Order Status");
36:             orderNumber =new TextField("Order=", null,6,
37:             TextField.NUMERIC);
38:             inputForm.append(orderNumber);
39:             cmdOK= new Command("OK", Command.SCREEN,1);
```

```
41:               cmdExit=new Command("Exit", Command.EXIT,1);
42:               inputForm.addCommand(cmdOK);
43:               inputForm.addCommand(cmdExit);
44:               inputForm.setCommandListener(this);
45:
46:
47:               outputForm=new Form("Order Status");
48:               orderStatus=new StringItem(null,null);
49:               outputForm.append(orderStatus);
50:               cmdBack=new Command("Back", Command.BACK,1);
51:               outputForm.addCommand(cmdBack);
52:               outputForm.addCommand(cmdExit);
53:               outputForm.setCommandListener(this);
54:
55:         try   {
56:
57:         conn=(StreamConnection)Connector.open(URL,Connector.READ_WRITE,true);
58:               os=conn.openOutputStream();
59:               is=conn.openInputStream();
60:               }
61:
62:         catch (Exception e)
63:               {
64:               destroyApp(false);
65:               notifyDestroyed();
66:               }
67:
68:         commandAvailable =false;
69:         commandThread=new CommandThread(this);
70:         commandThread.start();
71:
72:         display.setCurrent(inputForm);
73:         }
74:
75:         public void pauseApp()
76:         {
77:         }
78:
79:         public void destroyApp(boolean unconditional){
80:         try{
81:               os.write("Q".getBytes());
82:               conn.close();
83:               }
84:         catch (IOException e){
85:               }
86:         }
87:
88:         public void commandAction(Command cmd, Displayable d) {
89:               if(cmd==cmdExit){
90:                     destroyApp(false);
91:                     notifyDestroyed();
92:               }
93:
94:               else if(cmd==cmdBack){
95:                     orderNumber.setString(null);
96:                     display.setCurrent(inputForm);
97:               }
98:
99:               else if (cmd==cmdOK)
100:                    synchronized (this){
101:                    xmit=orderNumber.getString().getBytes();
102:                    commandAvailable =true;
103:                    notify();
104:              }
105:
106:
107:
108:
109:
110:
```

```
111:        }
112:
113:        class CommandThread extends Thread{
114:
115:              MIDlet parent;
116:              boolean exit=false;
117:
118:              public CommandThread(MIDlet parent){
119:                    this.parent =parent;
120:              }
121:
122:              public void run()
123:              {
124:                    while(true) {
125:                    synchronized (parent){
126:                          while(!commandAvailable){
127:                                try{
128:                                      parent.wait();
129:                                }
130:                                catch (InterruptedException e){
131:                                      }
132:                                }
133:                                commandAvailable =false;
134:                    }
135:                    getStatus();
136:                    }
137:              }
138:
139:              public void getStatus() {
140:              try
141:                    {
142:                    os.write(xmit);
143:                    os.flush();
144:
145:
146:                    int ch;
147:                    StringBuffer sb=new StringBuffer();
148:                    while ((ch =is.read())!=-1) {
149:                          sb.append((char)ch);
150:                          if((char)ch =='\n') {
151:                          break;
152:                          }
153:                    }
154:
155:              orderStatus.setText(sb.toString());
156:              display.setCurrent(outputForm);
157:              }
158:              catch(Exception e) {
159:              }
160:        }
161:  }
162:  }
```

Pada client digunakan connector.open() untuk membuat StreamConnection. Untuk komunikasi 2 arah digunakan getInputStream dan getOutputStream.

Setelah dikompile dan dieksekusi maka pada sisi client akan tampil seperti sebagai berikut

Gambar 12.2. Query status order

Gambar 12.3. Hasil query status order

Dan di sisi server akan tampil seperti sebagai berikut

```
D:\java>java OrderStatusServer
koneksi diterima
order Number=00001
status:terkirim
order Number=00002
status:pending
```

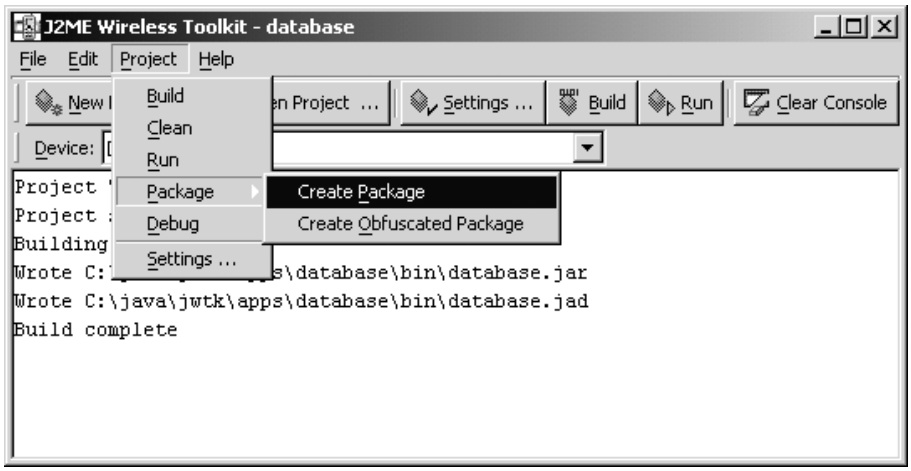
13

Installasi Aplikasi pada Perangkat Wireless

Setelah aplikasi yang kita buat dapat berjalan dengan baik sesuai dengan yang kita inginkan, maka pada tahap akhir adalah menginstall aplikasi yang telah dibuat pada perangkat wireless.

A. Pembundelan aplikasi

Sebelum aplikasi diinstall ke perangkat wireless, file-file yang terkait dengan aplikasi selain file classnnya seperti file gambar, data dan file-file pendukung lainnya perlu dibundel dalam 1 file JAR. Dalam J2ME Toolkit sudah terdapat fasilitas untuk membundel aplikasi yang kita buat yaitu pada menu **Project→ Package → Create Package**



Gambar 13.1. Pembundelan aplikasi

Hasil pembundelan aplikasi ini berupa file JAR dan JAD, file JAR merupakan aplikasi yang kita buat sedangkan file JAD merupakan file yang berisi informasi aplikasi yang kita buat. Kedua file ini yang akan diinstall ke dalam perangkat wireless.

B. Installasi aplikasi

Ada 2 teknik installasi pada perangkat wireless yakni: OTA(Over The Air) dan Installasi langsung.

B.1. OTA (Over The Air)

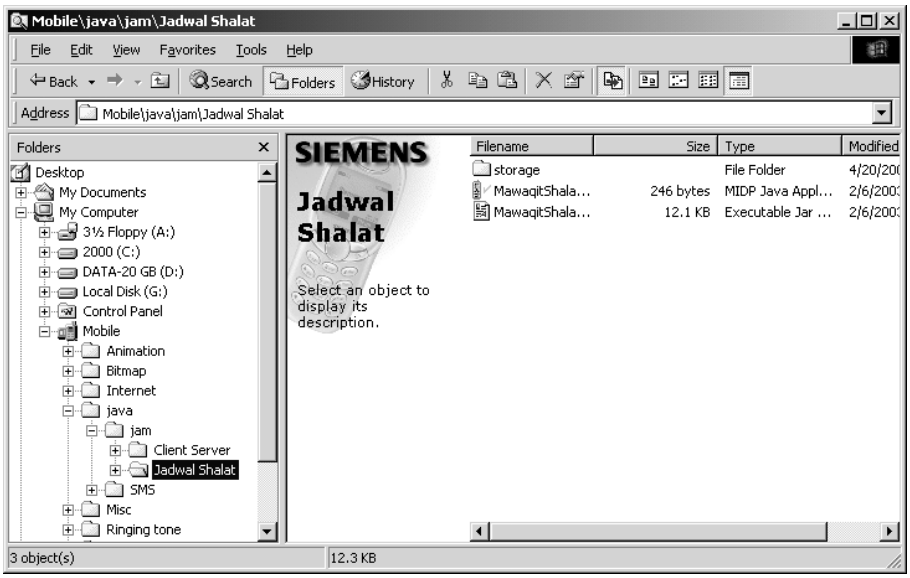
Pada OTA file JAD dan JAR disimpan di web server yang terhubung ke internet (dengan IP public tentunya). Pada web server perlu ditambahkan agar mendukung file JAD dan JAR yaitu pada konfigurasi file **mime.types**.

Setelah file JAD dan JAR disimpan dalam web server, akseslah alamat url file JADnya sehingga perangkat wireless yang dalam keadaan terkoneksi mendownload file JARnya.

B.2. Installasi langsung dari PC

Setiap vendor biasanya membuat program untuk transfer data antara PC dengan perangkat wireless. Media transmisi tergantung fasilitas yang disupport oleh produk tersebut seperti kabel data, infra merah, bluetooth.

Sebagi contoh installasi langsung pada ponsel Siemens dengan media kabel data memerlukan software DES(Data Exchange Software). Dengan software ini maka drive handhone akan terlihat pada Windows Explorer.



Gambar 13.2. Drive perangkat wireless dalam Windows Explorer

Untuk menginstall aplikasi J2ME dengan metode ini hanya dengan mengkopikan file JAR dan JAD ke dalam direktori tempat program Java (dalam hal ini direktori /java/jam/direktori aplikasi kita.

14
Penutup

Untuk bisa menguasai pemrograman pada perangkat wireless tentunya tidak hanya mengandalkan buku ini. Buku ini sengaja disusun secara sederhana yang mencakup garis-garis besarnya saja. Untuk lebih lengkapnya Anda harus membuka dokumentasi API dari J2ME dan API tambahan vendor sesuai dengan perangkat wirelessnya.

Selain itu Anda juga perlu membaca manual tentang teknik installasi sesuai vendor jika Anda ingin menggunakan teknik installasi langsung dari PC (bukan OTA).

Untuk mempermudah belajar Anda bisa mengikuti berbagai milis serta aktif membuka-buka arsip forum diskusi tentang aplikasi wireless.

DAFTAR PUSTAKA

1. Tremblet, Paul. Wireless Java with J2ME, McGraw-Hill/Osborne
2. Raka, Cokorda. Trik Pemrograman Java untuk Jaringan dan Internet, Elex Media Computindo
3. Hartono, Puji. Modul Praktikum Pemrograman Berorientasi Objek, Laboratorium Komputer STMIK-AMIKBANDUNG
4. Wicaksono, Ady. Pemrograman Aplikasi Wireless dengan Java, Elex Media Computindo
5. Sanjaya, Ridwan. Pemrograman di Linux dengan Java, Elex Media Computindo

A. Seting PATH

Agar program-program di direktori \bin\ bisa dieksekusi dari sembarang direktori, maka perlu di set PATHnya

Contoh:
Direktori \bin terletak di C:\java\j2se\bin maka seting pathnya

```
SET PATH=%PATH%;C:\Java\j2se\bin
```

B. Seting CLASSPATH

Agar class-class dapat menemukan class-class yang diimport-kan, maka perlu diset CLASSPATH nya.

Contoh:
Direktori \lib terletak di C:\java\j2se\lib maka seting classpathnya

```
SET CLASSPATH=C:\Java\j2se\lib;%CLASSPATH%
```

C. Seting JDBC

Agar class-class jdbc dapat diakses maka perlu ditambahkan seting CLASSPATH nya.

Contoh:
Direktori \jdbc terletak di C:\java\jdbc maka seting classpathnya

```
SET CLASSPATH=C:\Java\jdbc\;%CLASSPATH%
```