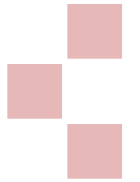
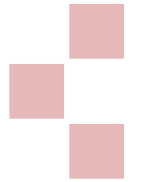


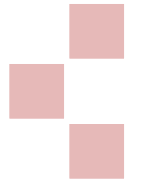
Wireless Java Programming With J2ME

Nurochman



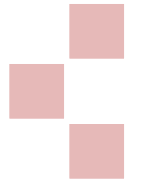
What is J2ME?

- a lean Java platform targeted specifically at applications running on small devices
- Such as mobile phones, PDAs, Internet screenphones, digital television set-top boxes, automotive entertainment and navigation systems, network switches, home automation components, and so on.

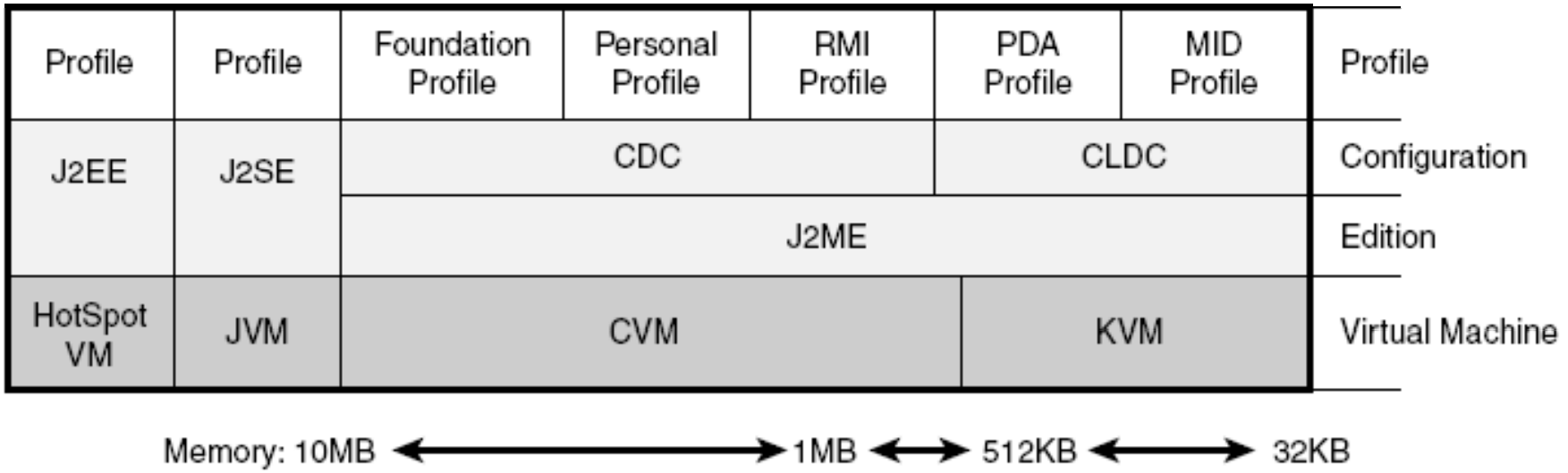


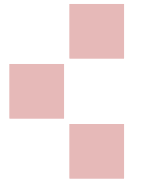
3 editions of Java 2 Platform

- **Java 2 Enterprise Edition (J2EE)**—Designed for heavyweight and scalable business server applications.
- **Java 2 Standard Edition (J2SE)**—Designed for traditional and well-established desktop applications.
- **Java 2 Micro Edition (J2ME)**—Designed for the new generation of applications that target consumer electronics and embedded devices.



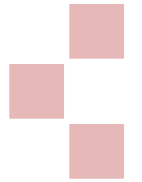
Java 2 Platform





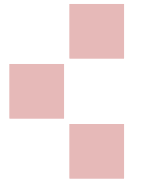
Tree Layers of J2ME (1)

- **Java Virtual Machine Layer**—This layer is an implementation of a Java virtual machine that is customized for a particular device's host operating system and supports a particular J2ME configuration. As shown in Figure 2.1, the virtual machines for J2ME are CVM and KVM.



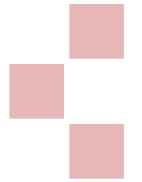
Tree Layers of J2ME (2)

- **Configuration Layer**—A J2ME configuration defines class libraries for a “horizontal” category or grouping of devices based on similar requirements for a total memory budget and processing power. As shown in Figure 2.1, the Connected Device Configuration (CDC) and CLDC are the two configurations identified in J2ME. You’ll find more information on CDC and CLDC in the next section.



Tree Layers of J2ME (3)

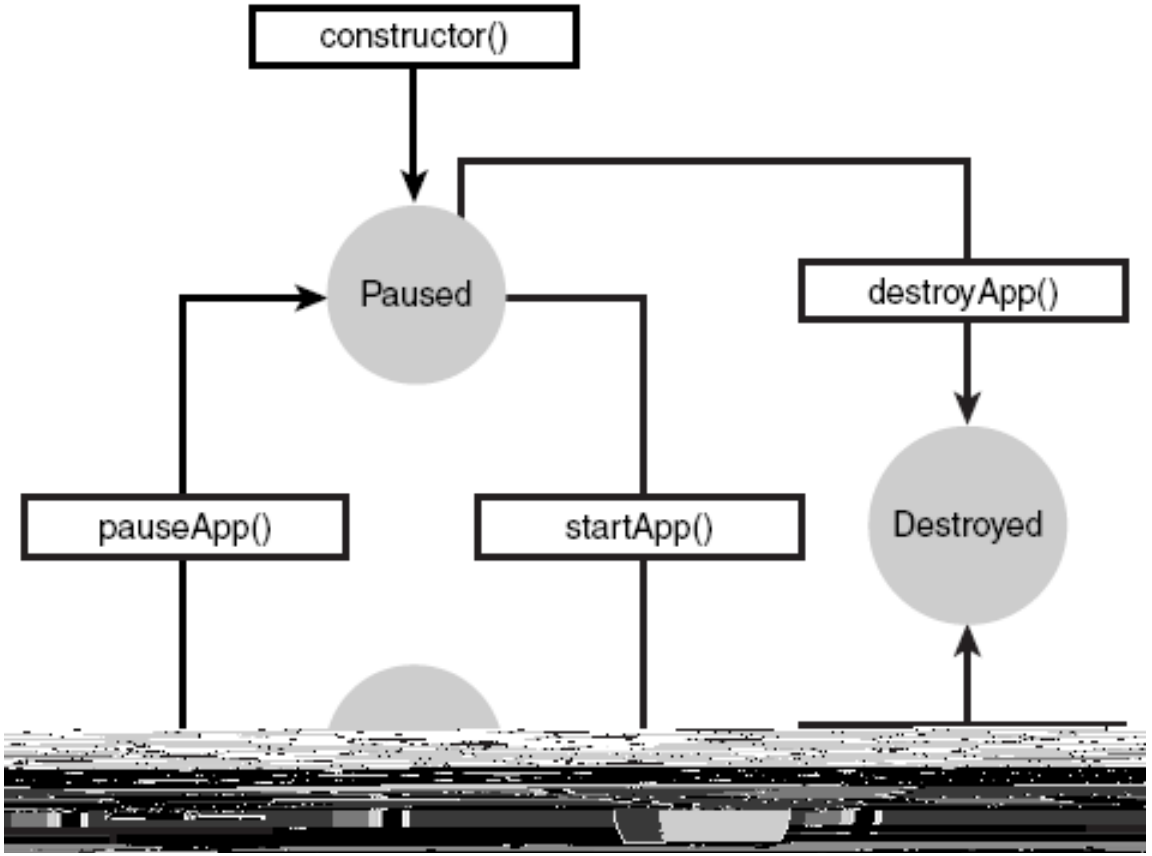
- **Profile Layer**—Built on top of a specific configuration, a J2ME profile defines class libraries to address the specific demands of a certain vertical market segment. As shown in Figure 2.1, the PDAP, MIDP, Foundation Profile, and Personal Profile are a few example profiles identified in J2ME.

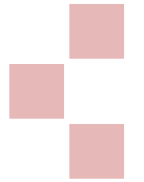


MIDlet

- An application on a MIDP device is called a *MIDlet.*
- *A MIDlet is very much like a Java applet.*
- It does not have a main() method; instead, a MIDlet must extend the javax.microedition.midlet.MIDlet class and implement its three abstract methods: startApp(), pauseApp(), and destroyApp(). A MIDlet also has to define a public no-argument constructor.

MIDlet's life cycle

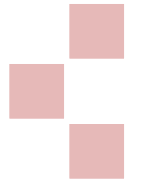




The definition of the javax.microedition.midlet.MIDlet class

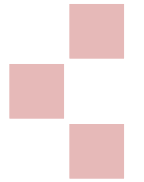
```
public abstract class MIDlet extends Object {
  protected MIDlet()
  protected abstract void startApp() throws MIDletStateChangeException
  protected abstract void pauseApp()
```

```
  public final void notifyPaused()
  public final String getAppProperty(String key)
  public final void resumeRequest()
}
```



J2SE VS J2ME

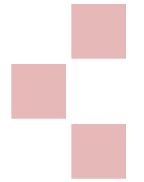
- In J2SE, Main class adalah class yang memiliki method main()
- Method main akan dipanggil secara implisit ketika aplikasi J2SE dijalankan
- In J2ME, Main Class adalah class yang di-extends dari class MIDlet
- Method startApp() akan dipanggil secara implisit ketika aplikasi J2ME dijalankan



Membuat midlet pertama

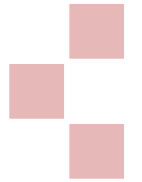
```
import javax.microedition.midlet.MIDlet;
public class Hello extends MIDlet {
  public void startApp() {}
  public void pauseApp() {}
  public void destroyApp(boolean b) {}
}
```

Method startApp() merupakan permulaan kode program kita



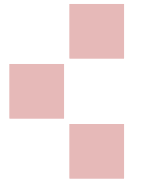
contoh

```
import javax.microedition.midlet.MIDlet;
public class Hello extends MIDlet {
  public void startApp() {
      System.out.println(“Hello”);
  }
  public void pauseApp() {}
  public void destroyApp(boolean b) {}
}
```



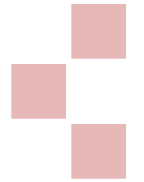
GUI Classes

- The classes associated with the GUI and event handling are defined in the `javax.microedition.lcdui` package.
- The Screen class is the superclass for all user interface components of the high-level API.
- These UI components include Alert, Form, List, TextBox, and so on.
- The Canvas class and Graphics class are the two main classes in the low-level API. Game applications will most likely make heavy use of the GUI classes in the low-level API.



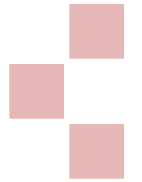
Class Display

- Merupakan class yang mengatur komponen apa yang akan di-display pada LCD
- Untuk membuat reference ke objek Display kita gunakan method static getDisplay()
- Method setCurrent(Displayable d) digunakan untuk menentukan komponen apa yang akan tampil pada LCD
- Class Displayable adalah class abstract, dimana subclassnya antara lain : Form, Textbox, Alert, dll



Membuat Display reference

```
public void startApp() {
  Display d = Display.getDisplay(this);
}
```



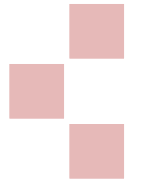
Class Form

- Kontruktor Form(String title)

```
Form f = new Form(“Hello”);
```

- Tampilkan form dalam display

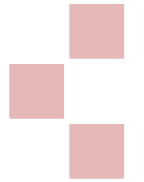
```
d.setCurrent(f);
```



Program menjadi

```
import javax.microedition.midlet.MIDlet;
Import javax.microedition.midlet.lcdui.*;
public class Hello extends MIDlet {
  public void startApp() {
        Display d = Display.getDisplay(this);
        Form f = new Form(“Hello”);
        d.setCurrent(f);
  }
  public void pauseApp() {}
  public void destroyApp(boolean b) {}
}
```

Gimana outputnya???

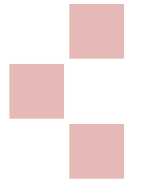


Class TextField

- Merupakan komponen utk input text, seperti username, password, dll
- Konstruktor TextField(label, text, maxSize, constraint)
- Label merupakan keterangan data yang akan diinputkan
- Text merupakan string dalam text field
- maxSize, panjang max dari simbol yg masuk
- Constraint, jenis input (misal:numeric, password, any)

Contoh membuat objek TextField

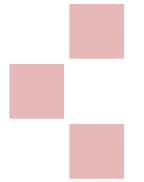
- TextField tfNama = new TextField(“Nama”, “”, 30, TextField.ANY);
- TextField.ANY, merupakan constraint bahwa semua karakter boleh dimasukan
- TextField.NUMERIC, hanya angka yang boleh masuk
- TextField.PASSWORD, karakter yang masuk akan berubah jadi *



Terapkan dalam program!

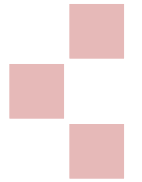
```
public void startApp() {
  Display d = Display.getDisplay(this);
  Form f = new Form("Hello");
  TextField tfNama = new TextField("Nama",
  "", 30, TextField.ANY);
  f.append(tfNama);
  d.setCurrent(f);
}
```

Perintah **f.append(tfNama)**, untuk memasukan text field ke dalam form.



Class Command

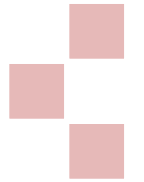
- Merupakan class untuk membuat objek tombol
- Konstruktor Command(label, commandType, priority)
- Label, merupakan label dari tombol
- commandType, tipe command (Command.OK, Command.EXIT, Command.BACK, dll)
- Priority, merupakan prioritas sebuah tombol untuk dieksekusi



Contoh membuat tombol

- Command cmdExit = new Command(“Exit”, Command.EXIT, 0);
- Menambahkan command ke form

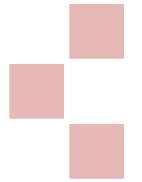
  f.addCommand(cmdExit);



Terapkan dalam program!

```
public void startApp() {
  Display d = Display.getDisplay(this);
  Form f = new Form("Hello");
  TextField tfNama = new TextField("Nama",
  "", 30, TextField.ANY);
  f.append(tfNama);
  Command cmdExit = new Command("Exit",
  Command.EXIT, 0);
  f.addCommand(cmdExit);
  d.setCurrent(f);
}
```

Gimana outputnya???

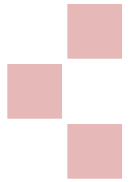


Event Handling pada Command

- Tombol exit tadi belum merespon terhadap aksi user
- Karena blm memiliki Event Handler
- Bagaimana menambahkan event handler?
- Form harus memberitahukan event handler-nya dimana dengan method setCommandListener().

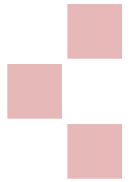
  setCommandListener(CommandListener cl);
- Parameter dr method tsb harus objek dari class yang implements interface CommandListener
- Class yg implements interface CommandListener harus meng-override method commandAction(…)

```
import javax.microedition.midlet.MIDlet;
Import javax.microedition.midlet.lcdui.*;
public class Hello extends MIDlet implements CommandListener {
  public void startApp() {
      Display d = Display.getDisplay(this);
      Form f = new Form("Hello");
      TextField tfNama = new TextField("Nama", "", 30,
      TextField.ANY);
      f.append(tfNama);
      Command cmdExit = new Command("Exit", Command.EXIT, 0);
      f.addCommand(cmdExit);
      d.setCurrent(f);
      f.setCommandListener(this);
  }
  public void pauseApp() {}
  public void destroyApp(boolean b) {}

  public void commandAction(Command c, Displayable d) {
      destroyApp(true); //keluar aplikasi
  }
}
```



An other Example: “Hello World”

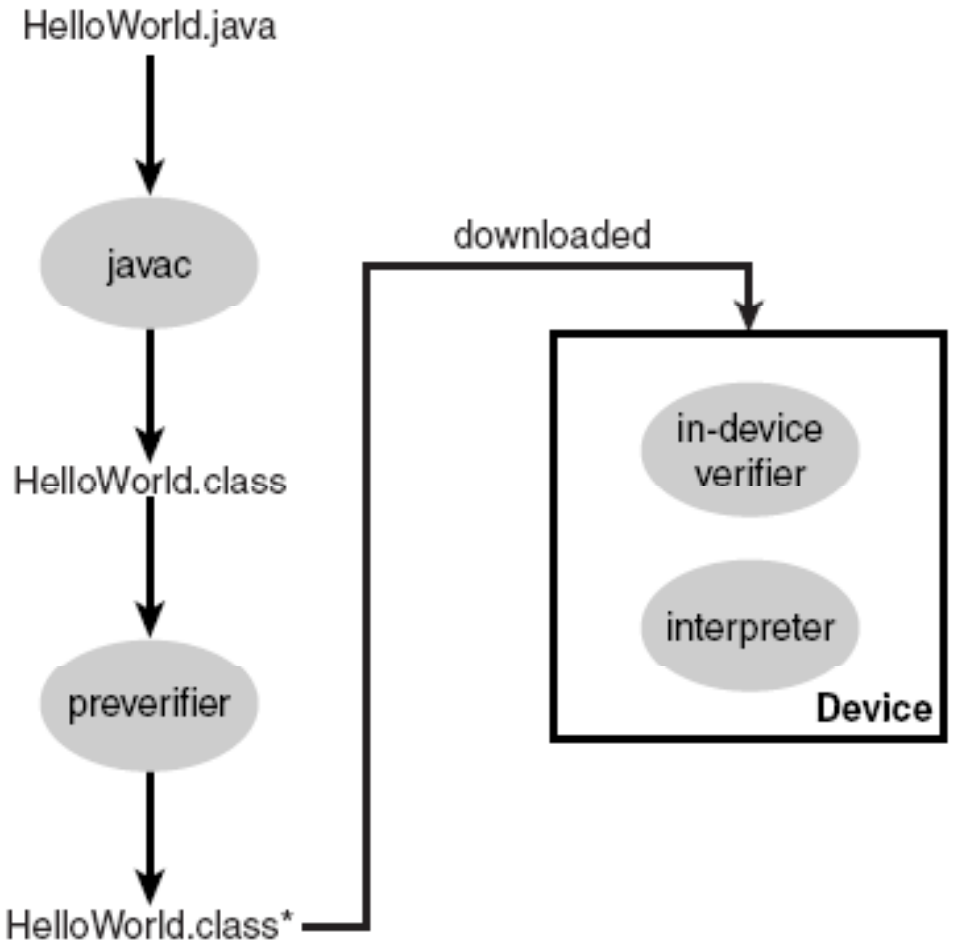
```
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.*;

public class HelloWorld extends MIDlet implements CommandListener{
    private Display display;
    private TextBox mainScreen = null;
    private Command exit;

    public HelloWorld() {
        display = Display.getDisplay(this);
        mainScreen = new TextBox("HelloWorld", "Hello World", 512,0);
        exit = new Command("exit", Command.EXIT, 2);
        mainScreen.addCommand(exit);
        mainScreen.setCommandListener(this);
    }
```



```
    public void startApp() {
        display.setCurrent(mainScreen);
    }

    public void pauseApp() {}

    public void destroyApp(boolean unconditional) {}

    public void commandAction(Command c, Displayable s) {
        if(c == exit) {
            destroyApp(false);
            notifyDestroyed();
        }
    }
}
```

+5550000 - DefaultColorPhone
MIDlet
Help
ABC
HelloWorld
Hello World
exit
SELECT
CLEAR
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
* .
SHIFT
0
- +
SPACE

Class verification process in KVM



Pertanyaan???