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.

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

High Level UI	*Low-Level UI*
Tidak memiliki akses untuk peralatan dengan feature yang spesifik	Mengakses masukkan low-level hanya dengan menekan

Gambar: Perbedaan High-Level UI dengan Low-Level UI

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

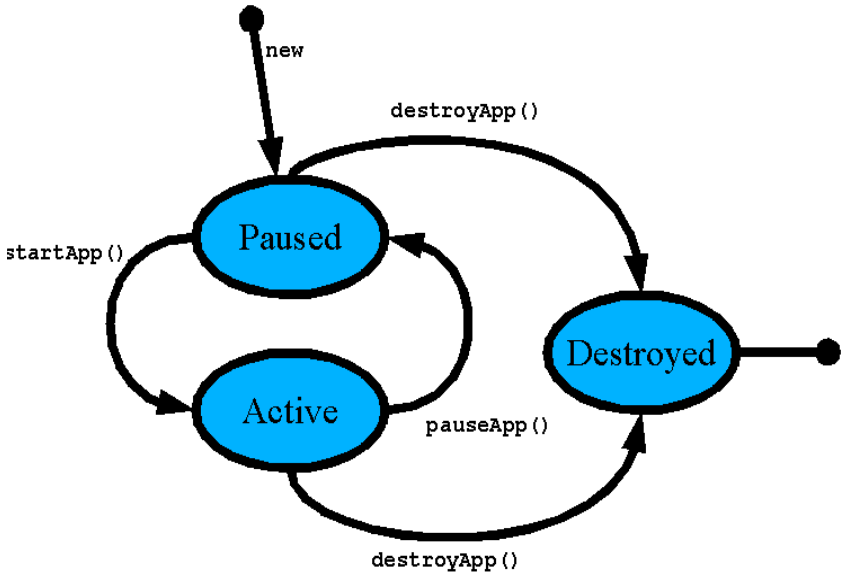
3.2.1 Display

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.



Gambar: MIDlet Life Cycle

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.

Gambar: Properti dari sebuah Displayable

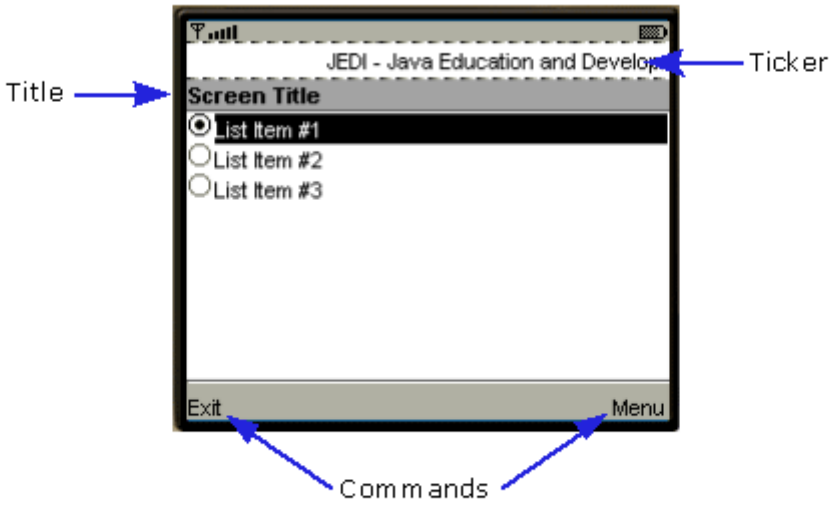
3.2.3 Title

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().

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

Gambar: Contoh tampilan dari multiple Commands

Command memiliki sebuah short label, long label, tipe dan prioritas.

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.

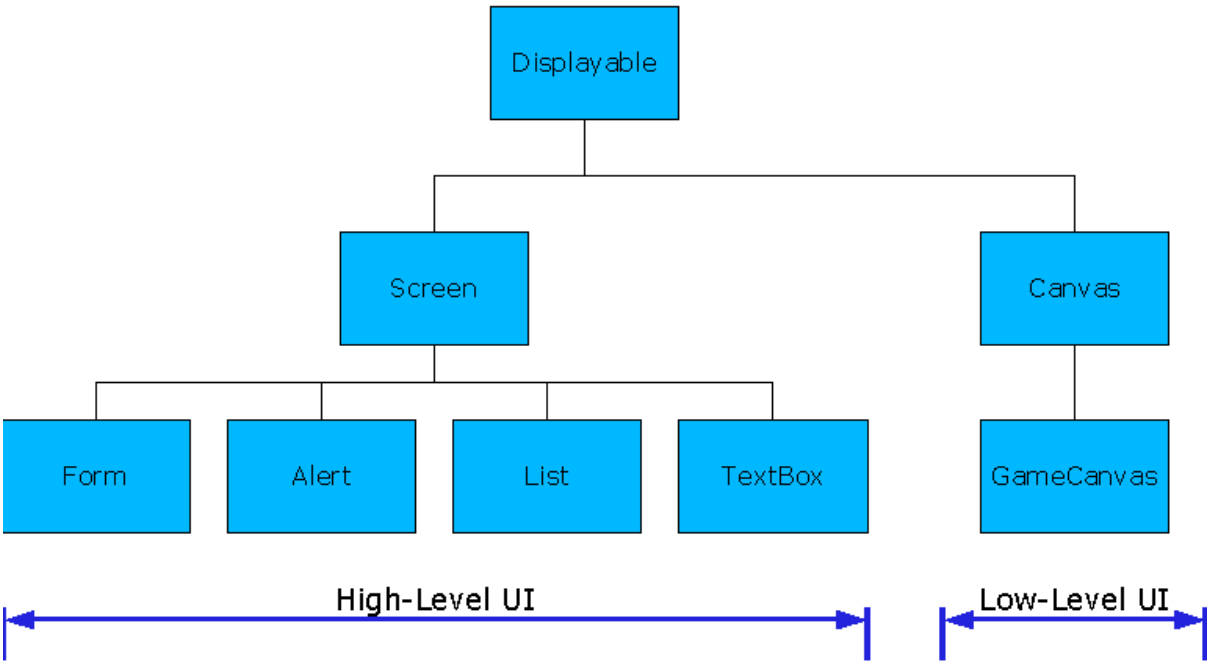
Memberi parameter null pada setTicker akan mengganti semua ticker yang telah dimasukkan pada Displayable. Menghapus ticker dari Displayable dapat menyebabkan perubahan ukuran area dari isi Displayable tersebut. Jika perubahan ukuran area terjadi, maka MIDlet akan memanggil sebuah ukuran dengan method 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.

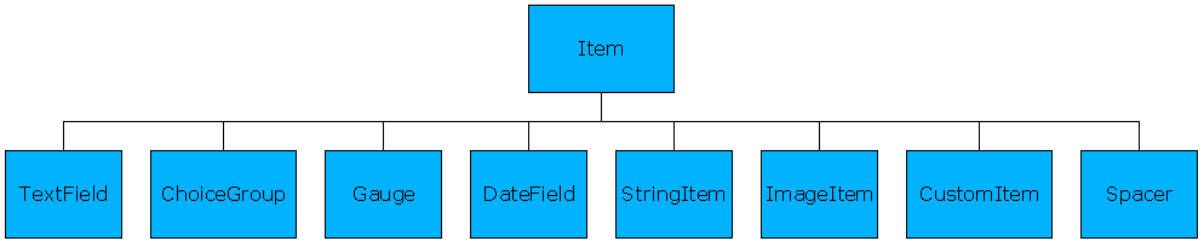


Gambar: Displayable Class Heirarchy

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 menggunkan 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.

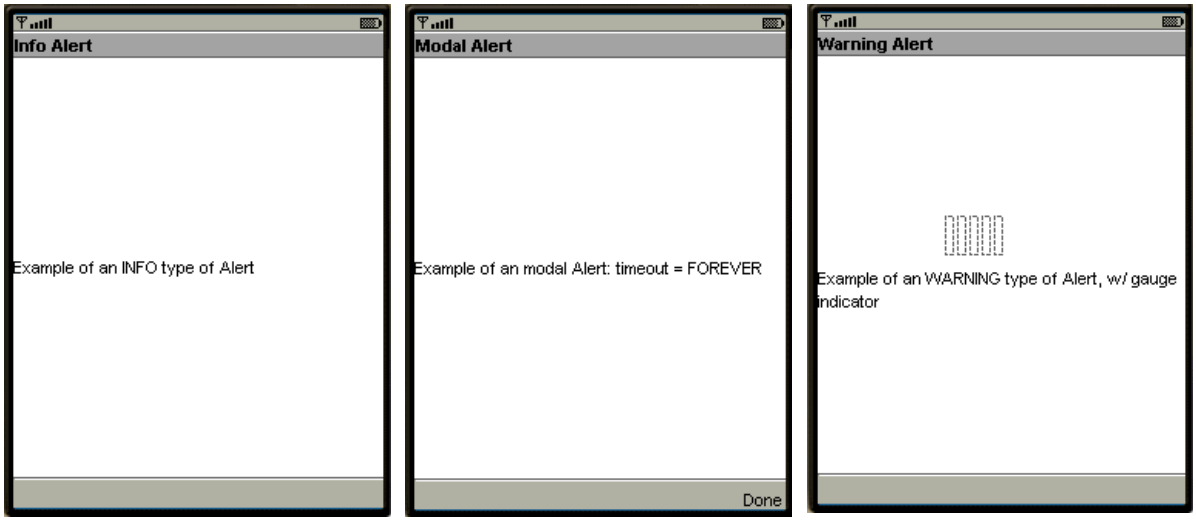
Alert juga dapat menampilkan komponen Gauge (Lihat pada Gauge item) sebagai indikator.

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]);
            }
        }
    }
}
```



INFO Alert ***Modal Alert*** ***Alert w/ gauge indicator***

Gambar: Perbedaan tipe-tipe Alert.

3.4 List

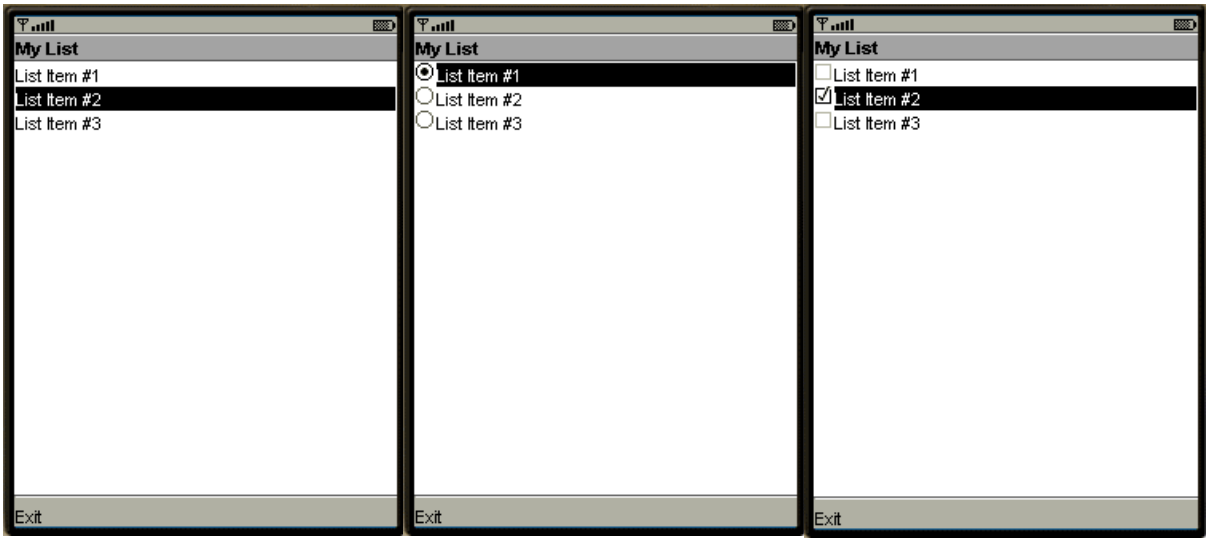
List merupakan subclass dari screen yang berisi sebuah daftar dari suatu pilihan. Sebuah list dapat dibagi menjadi tiga tipe: IMPLICIT, EXCLUSIVE atau MULTIPLE.

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);
            // menjalanjan suatu hal
        }
    }
}
```



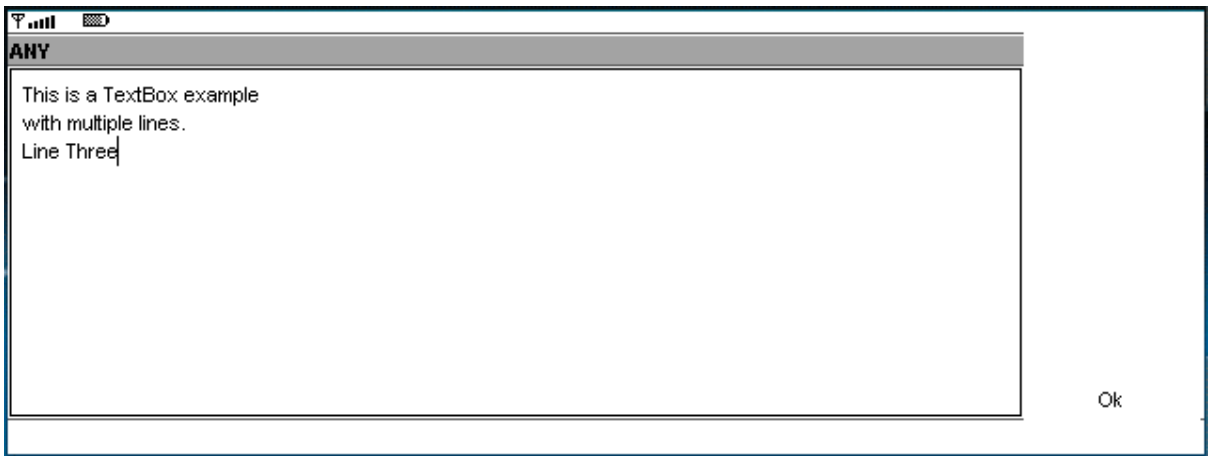
List.IMPLICIT ***List.EXCLUSIVE*** ***List.MULTIPLE***

Gambar: Tipe-tipe List

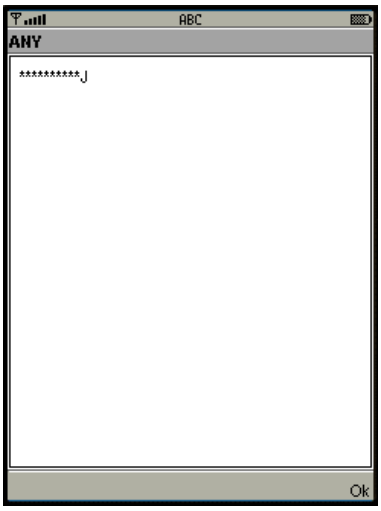
3.5 Text Box

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.

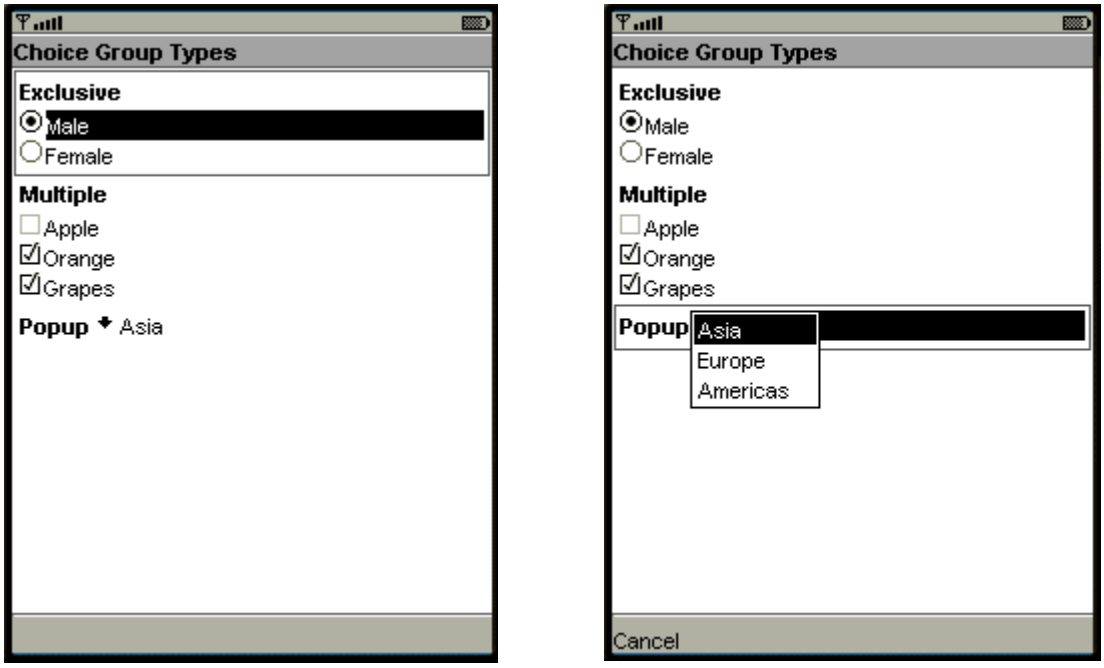
3.7 ChoiceGroup

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

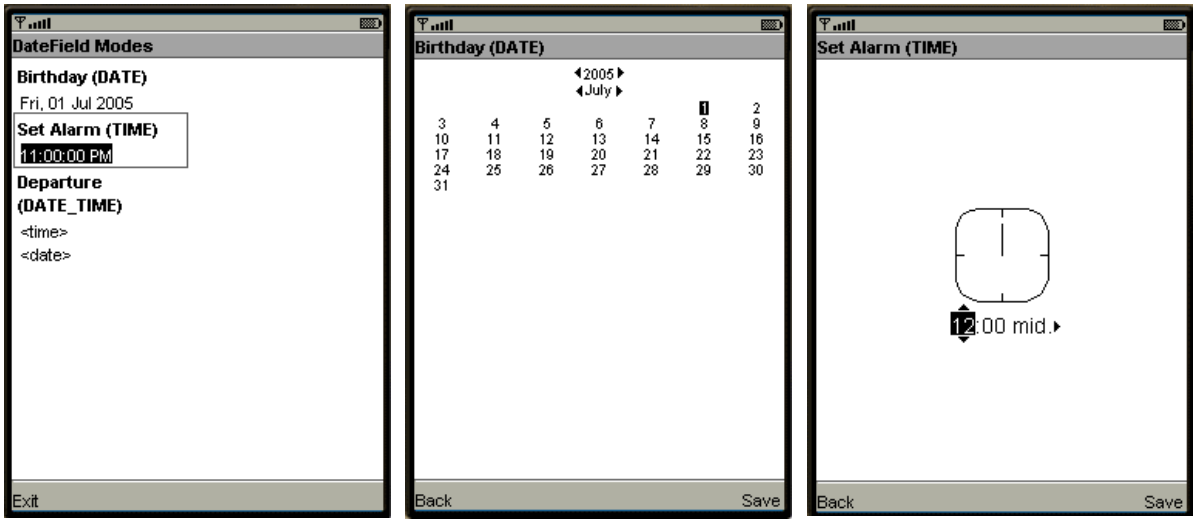
3.8 Date Field

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):

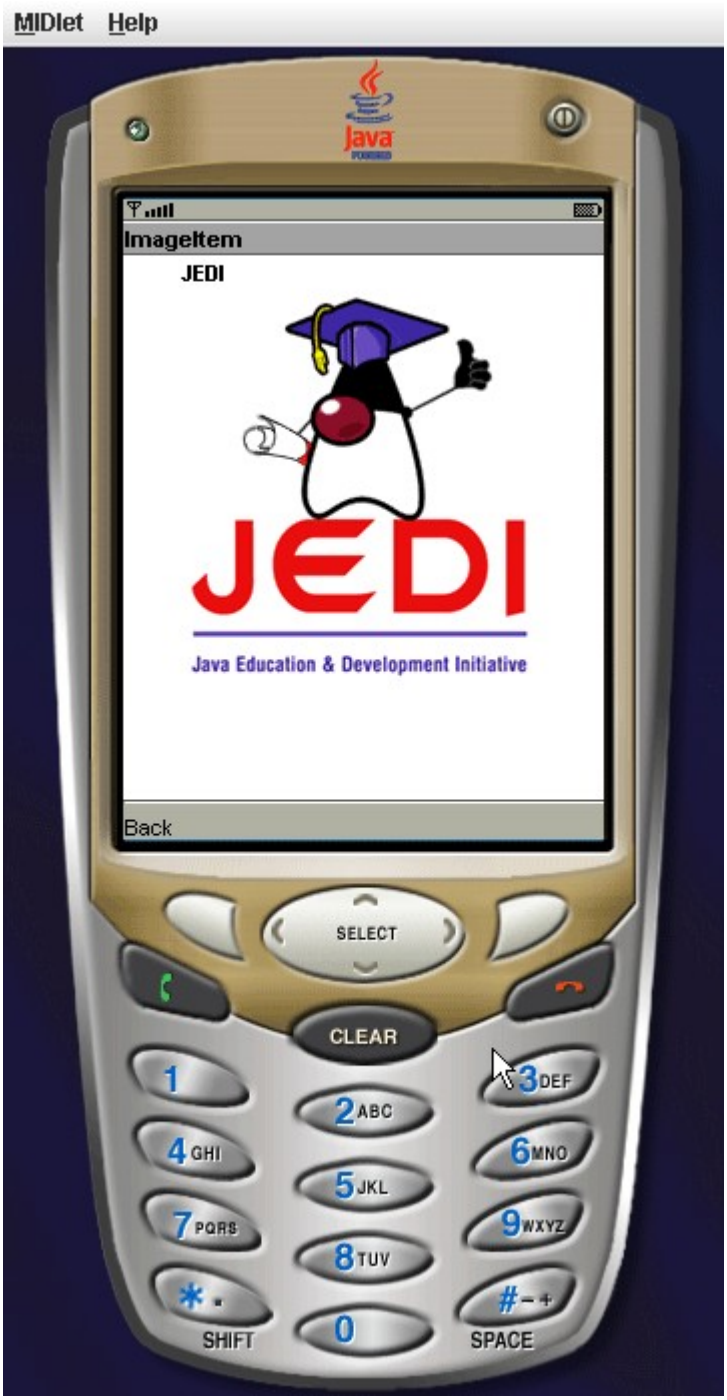
```
public ImageItem(
          String label,
          Image img,
          int layout,
         String altText)
```

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, "jeni 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

3.12 Latihan

3.12.1 List Dinamis

Buatlah sebuah MIDlet yang memiliki List IMPLICIT sebagai Screen main. Masukan tiga Command kedalam List ini - "Add Item", "Remove Item" dan "Exit". Comman "Add Item" akan memberikan layanan pada user untuk memasukan list menggunakan TextBox, kemudian insert item tersebut sebelum current item yang dipilih dari list. "Remove Item" akan menghapus currently selected list item (getSelectedIndex). Command "Exit" akan keluar dari program.