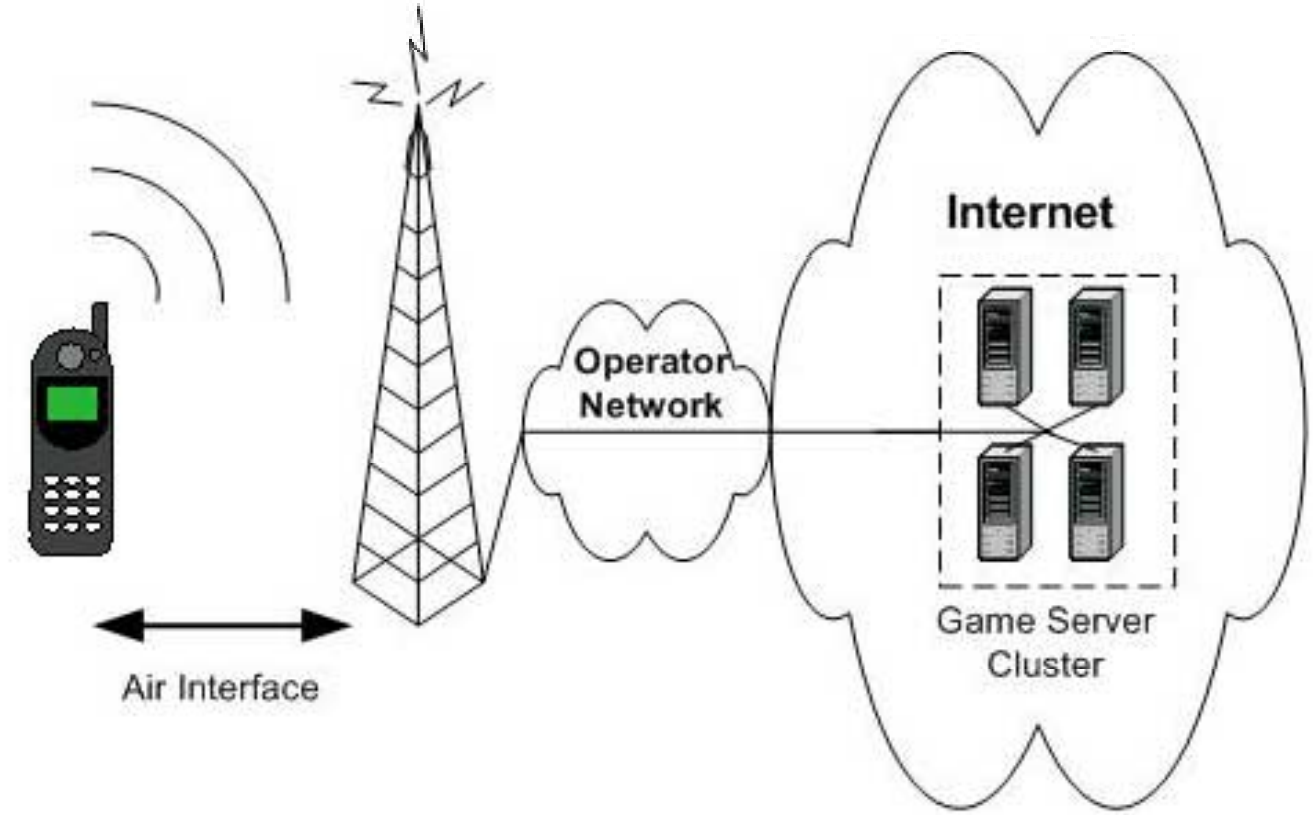
J2ME GUI dan Manajemen Event

Muhammad Zen S. Hadi, ST. MSc.

Contents

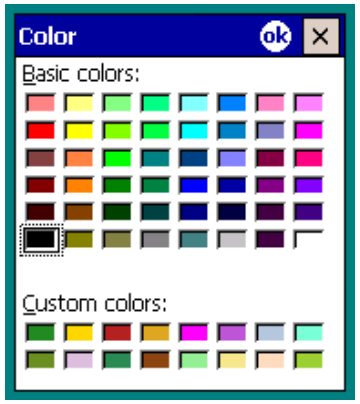
- J2ME (Review)
- Manajemen Event
- GUI dalam J2ME
- Passing parameter di dalam J2ME

Arsitektur Jaringan dgn J2ME

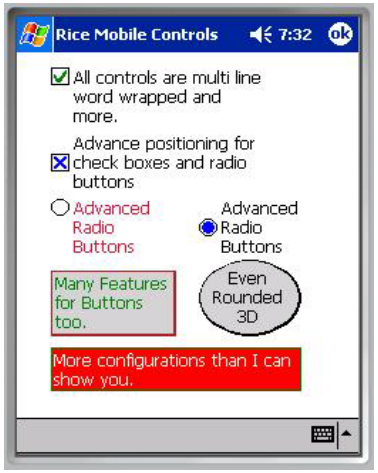


Design Aplikasi

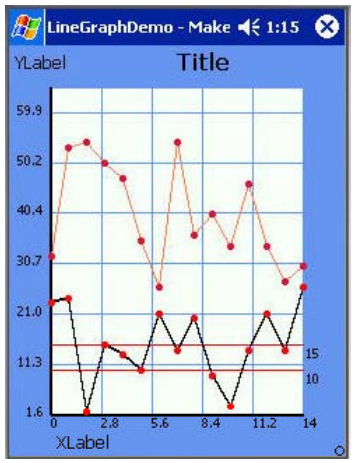
vMiles



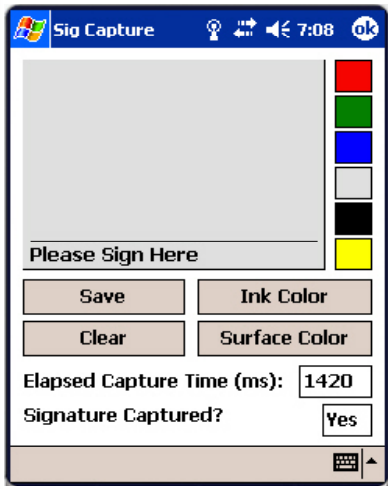
Color Dialog



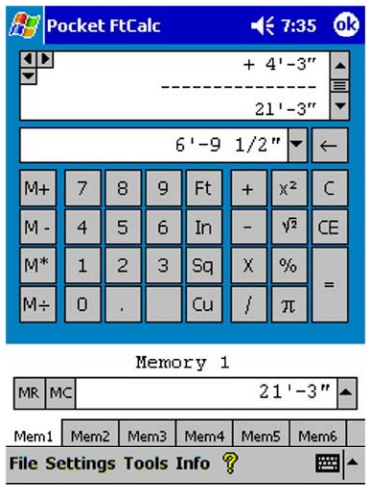
Mobile Controls



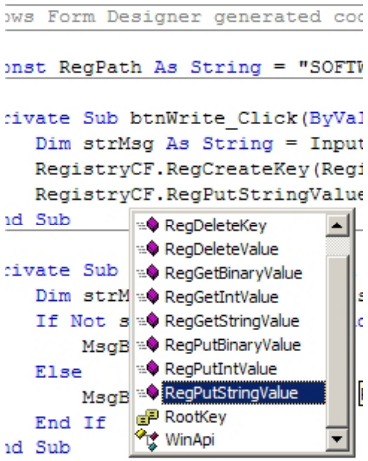
Micrographs



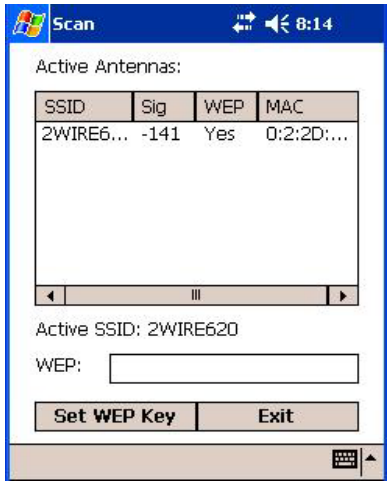
Signature Capture Control



Pocket ftCalc



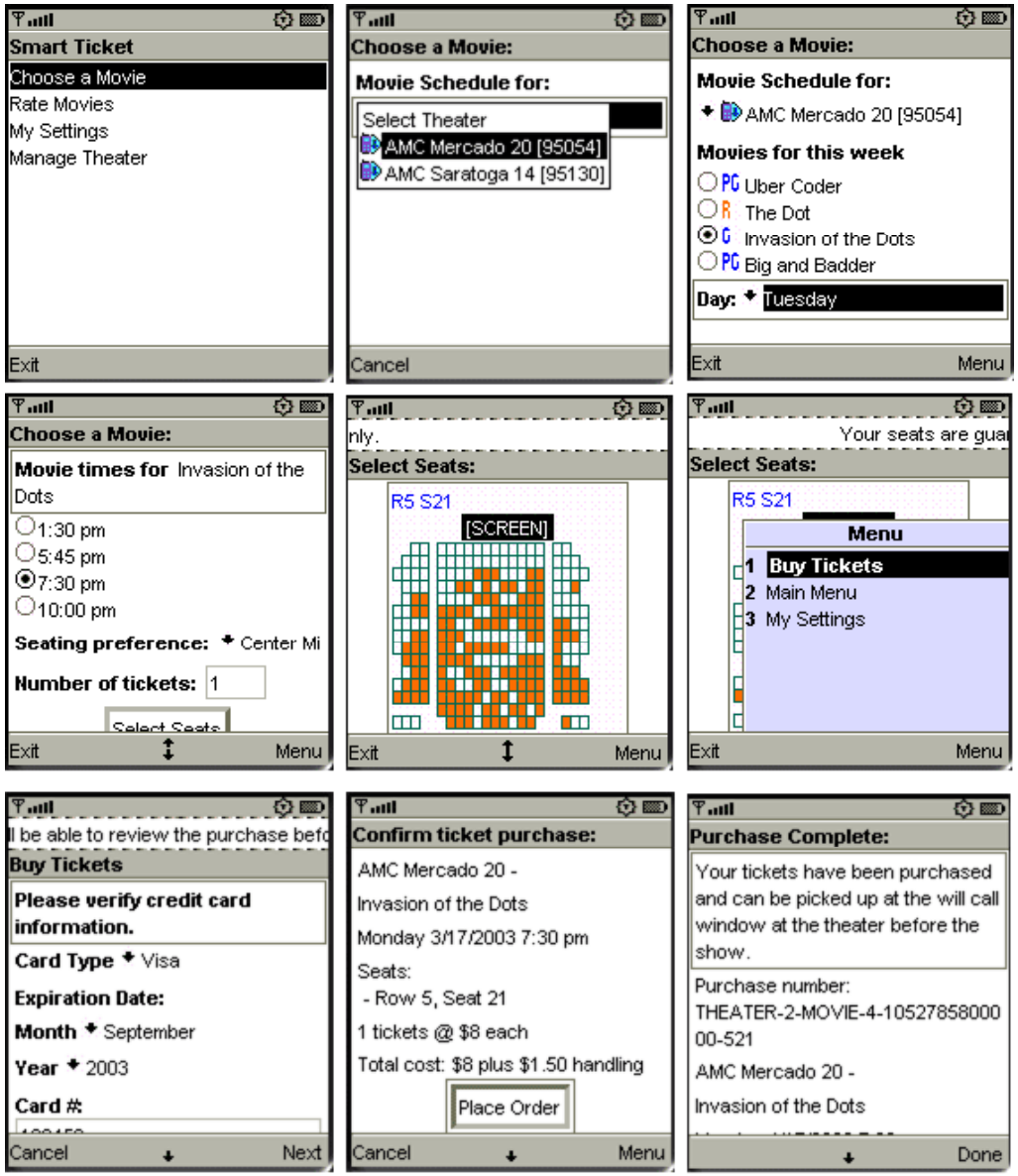
Registry Control



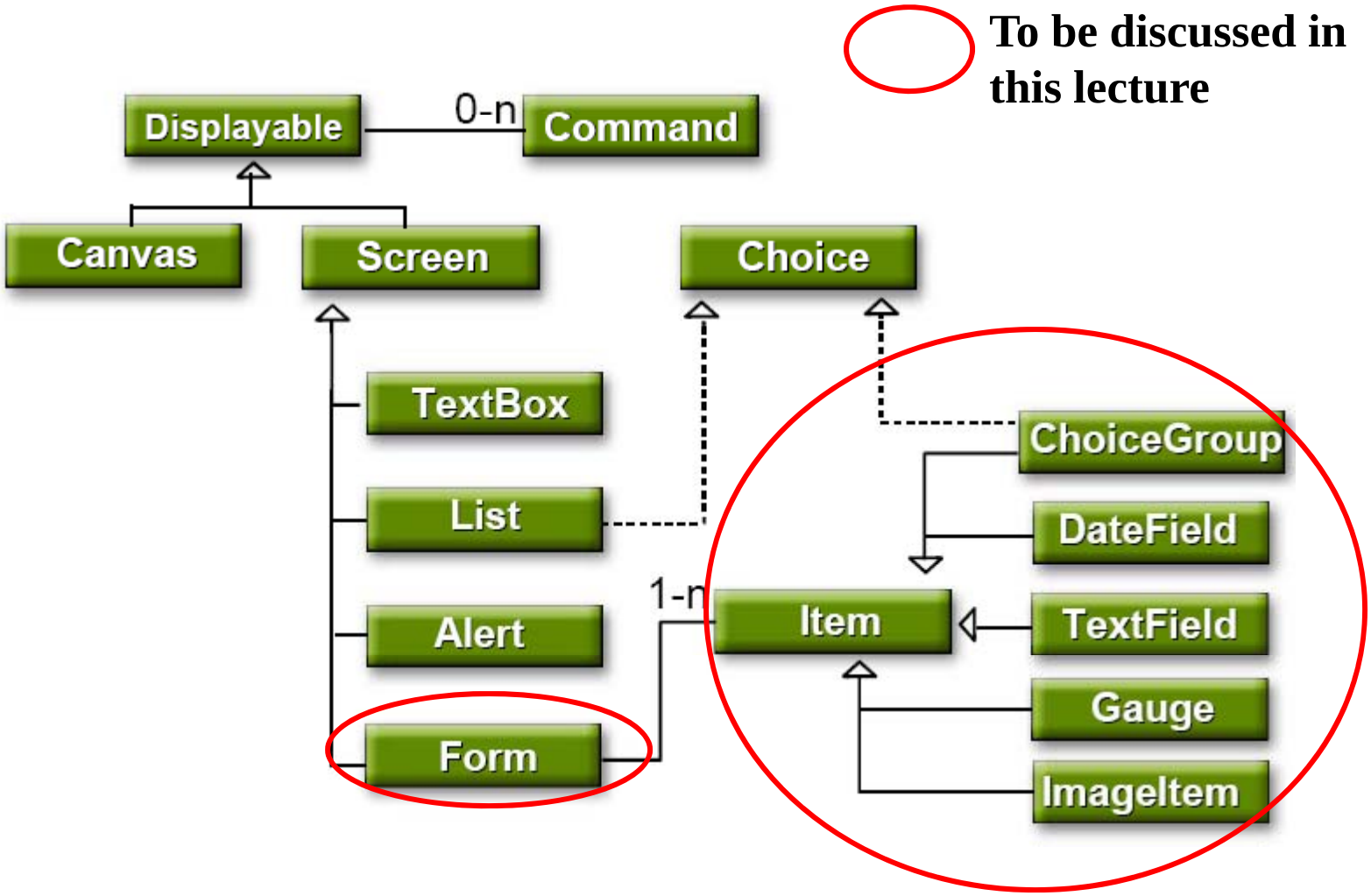
Simple Wireless API

Akses Informasi

- Pembelian tiket bioskop



Klas Utama dalam paket lcdui



Bagaimana program bisa keluar?

- Program tidak bisa keluar kecuali kita menutup emulator
- Supaya tidak perlu menutup emulator diperlukan sebuah command.
- Command bisa berupa button “OK" atau "Cancel," dan aplikasi kita bisa meresponds jika kita memilih command tersebut

Event Handling dengan Command

- Mengimplementasikan interface CommandListener

```
public class demo extends MIDlet implements CommandListener
{
    ……................................
}
```

-

```
public void commandAction(Command c,Displayable d)
{
    ……................................
}
```

Event Handling dengan Commands

- Displayable, parent dari semua screen display, support adanya Command.
- Perangkat mobile menentukan bagaimana perintah ditunjukkan di layar atau dijalankan oleh user.
- Setiap Displayable menyimpan daftar dari Command. Kita dapat menambah dan menghapus Command menggunakan metode berikut :
  - public void addCommand(Command cmd)
  - public void removeCommand(Command cmd)

Objet Command

- Dalam J2ME, command ditunjukkan dengan soft-button pada perangkat mobile. Gambar berikut terdapat 2 object Command, yaitu “Exit” dan “View

soft-buttons

Object Command

- Jika terdapat banyak command yang akan ditampilkan di layar, perangkat mobile akan membuat menu untuk menyimpan banyak command.

Penggunaan object Command

- Tahapan untuk memproses event dengan object command adalah sebagai berikut :
  1. Membuat Object Command
  2. Menambahkan Command ke Form (atau object GUI lainnya seperti TextBox, List, or Canvas).
  3. Membuat dan men-set listener untuk Form tersebut.
- Untuk mendeteksi dari event diatas, listener akan memanggil metode commandAction().

Menciptakan Command

- Untuk menciptakan Command, dibutuhkan label, type, dan prioritas
- Beberapa type command yang ada bisa dilihat pada tabel dibawah ini:

Command	**Meaning**
BACK	Kembali ke layar sebelumnya
CANCEL	Membatalkan akses ke layar
EXIT	Keluar dari suatu aplikasi
HELP	Menampilkan fasilitas help
ITEM	Menampilkan spesifik item.
OK	Mengacu jawaban positif atas suatu hal
SCREEN	Layar yang akan ditampilkan berdasar aplikasi
STOP	Menghentikan sebuah proses

Pembuatan Command

- Untuk membuat OK command, berikut perintahnya:

```
Command c = new Command("OK", Command.OK, 0);
```

label

type

priority

- Untuk membuat command tertentu ke aplikasi, berikut perintahnya:

```
Command c = new Command(
        "Launch", Command.SCREEN, 0);
```

Respon ke Commands

- Command ditunjukkan pada layar, tetapi tidak ada yang dijalankan secara otomatis ketika user menekan tombol command.
- Perlu object yang disebut listener yang akan dijalankan ketika user menekan tombol command
- Listener adalah object yang mengimplementasikan interface CommandListener.
- Untuk meregister listener dengan Displayable, gunakan metode berikut :
  - public void setCommandListener(CommandListener l)
- Catatan : Satu listener per Displayable, BUKAN satu listener per command.

Contoh

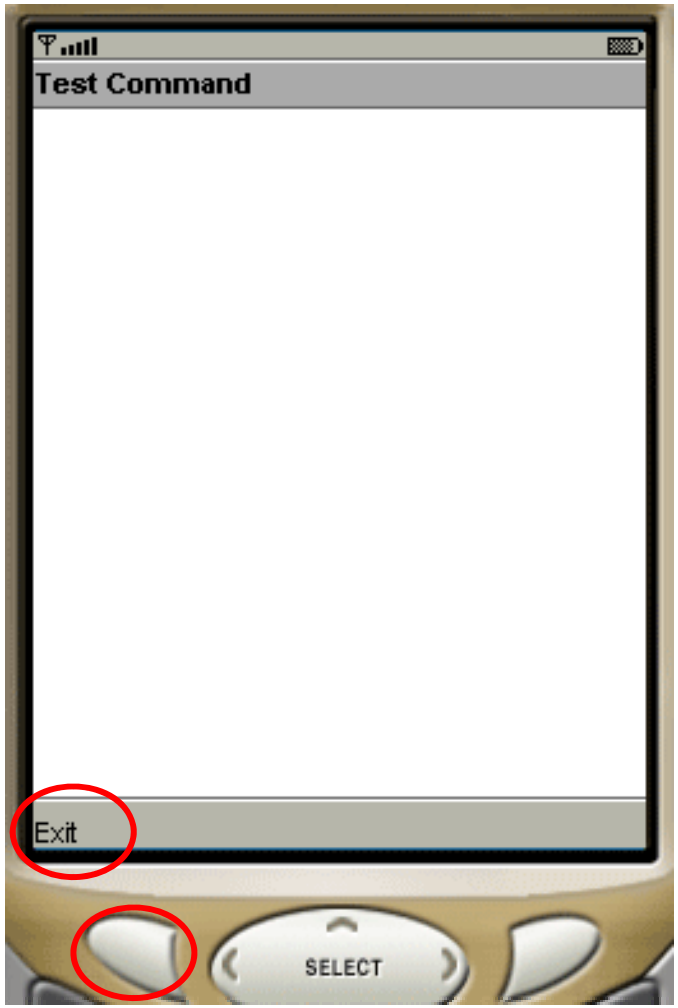
```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class cobaCmd extends MIDlet implements CommandListener {
    public void startApp() {
        Form d = new Form( "Test Command" );
        Command cb = new Command("Exit", Command.EXIT, 0);
        d.addCommand(cb);
        d.setCommandListener(this);
        Display.getDisplay(this).setCurrent(d);
    }

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

    public void commandAction(Command c, Displayable s) {
        notifyDestroyed();
    }
}
```

Abstract method of CommandListener. Akan dipanggil ketika beberapa command di Form dipilih.

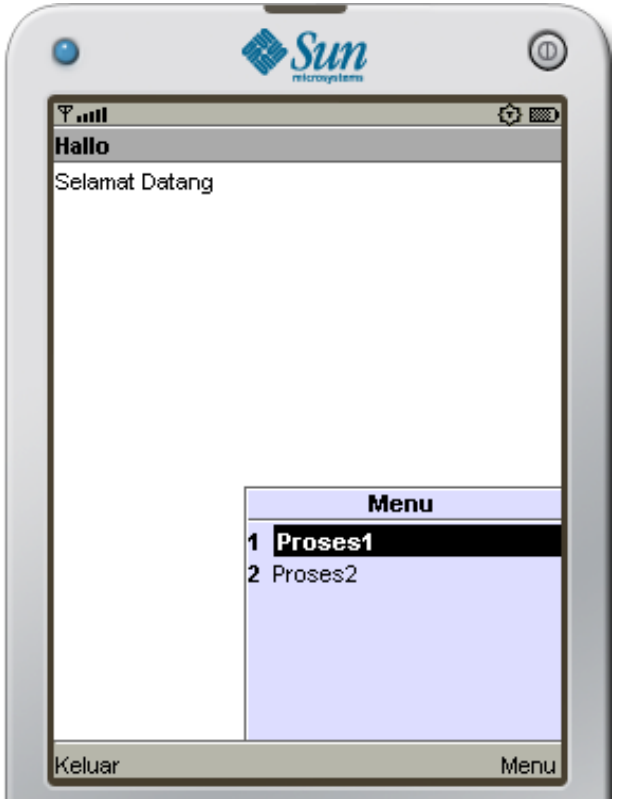

Test Command
Exit
SELECT


Select one to launch:
Exit Commnad
Launch
SELECT

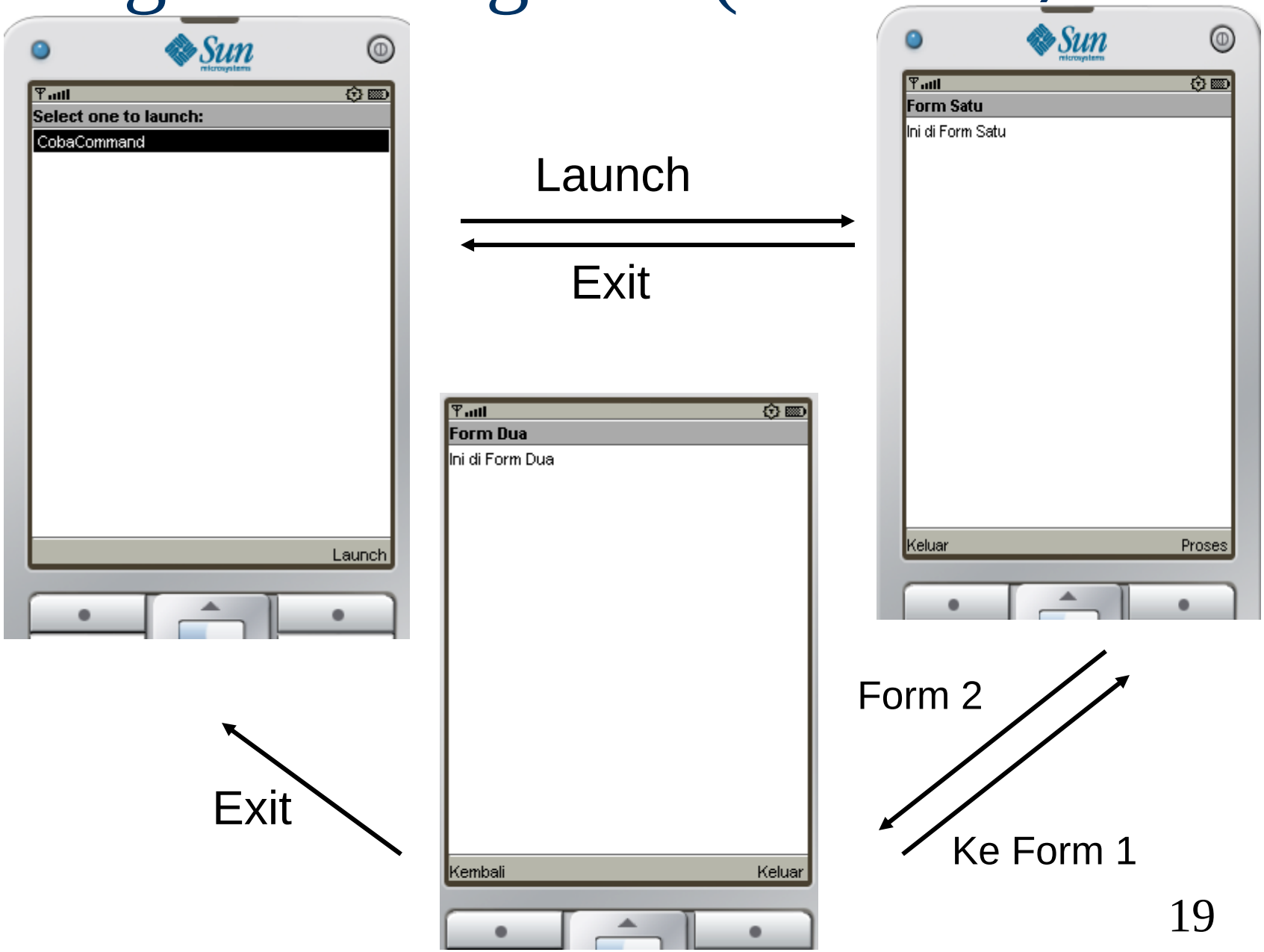
Example Multi Command

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

public class coba extends MIDlet implements CommandListener {
    private Display display;
    private Command exitCmd = new Command ("Keluar",Command.EXIT,0);
    private Command aboutCmd = new Command ("Proses1",Command.OK,1);
    private Command backCmd = new Command ("Proses2",Command.OK,1);
    private Form form;
    public void startApp() {
        form = new Form ("Hallo");
        form.append("Selamat Datang");
        form.addCommand(exitCmd);
        form.addCommand(aboutCmd);
        form.addCommand(backCmd);
        form.setCommandListener(this);
        display.getDisplay(this).setCurrent(form);
    }
    public void pauseApp() {
    }
    public void destroyApp(boolean unconditional) {
    }
    public void commandAction (Command c, Displayable d) {
        notifyDestroyed();
    }
}
```



Bagaimana dgn ini (2 Forms)



Contoh aplikasi

```
public class cobaCmd extends MIDlet implements CommandListener {
    Form satu, dua;
    Command keluarCmd = new Command ("Keluar",Command.EXIT,1);
    Command prosesCmd = new Command ("Proses",Command.OK,2);
    Command kembaliCmd = new Command ("Kembali",Command.BACK,2);
    public void startApp() {
        satu = new Form( "Form Satu" );
        satu.append ("Ini di Form Satu");
        satu.addCommand(keluarCmd);
        satu.addCommand(prosesCmd);
        satu.setCommandListener(this);
        Display.getDisplay(this).setCurrent(satu);
    }
    public void pauseApp() { }
    public void destroyApp(boolean unconditional) { }
    public void awal() {
      Display.getDisplay(this).setCurrent( satu );
    }
    public void keluar() {
      notifyDestroyed();
    }
```

```
    public void proses() {
        dua = new Form ( "Form Dua" );
        dua.append ("Ini di Form Dua");
        dua.addCommand(kembaliCmd);
        dua.addCommand(keluarCmd);
        dua.setCommandListener(this);
        Display.getDisplay(this).setCurrent(dua);
    }
    public void commandAction (Command c, Displayable d) {
        String data = c.getLabel();
        if (data == "Keluar") {
            keluar();
        } else if (data == "Proses") {
            proses();
        } else if (data == "Kembali") {
            awal();
        }
    }
}
```

Form dan komponen Item

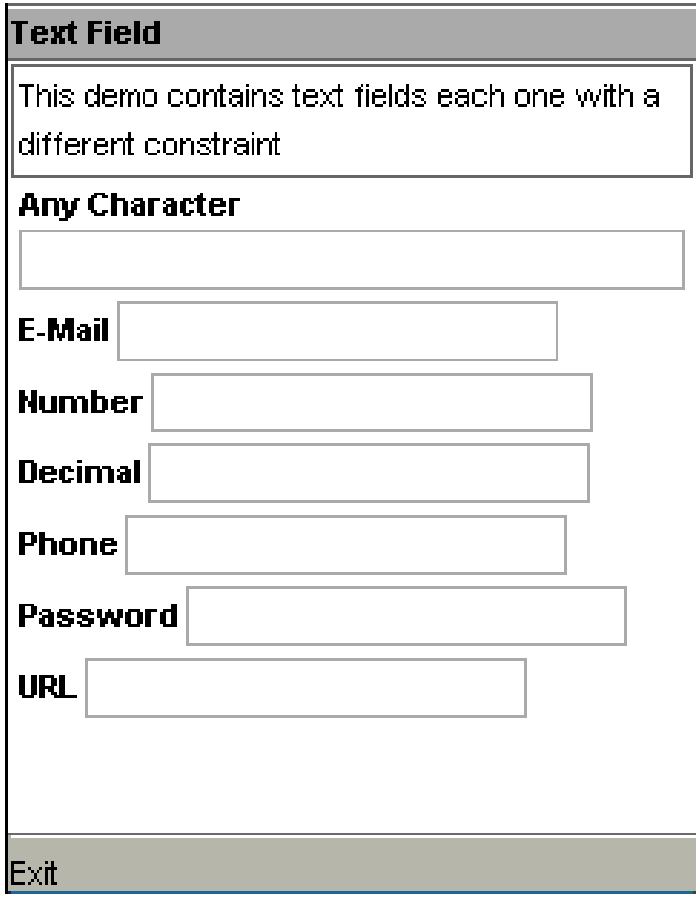
- **Form** adalah container untuk menampung komponen, dimana setiap komponen adalah subclass dari class **Item**.
- Untuk membuat Form:
  - public Form(String title)
  - public Form(String title, Item[] items)
- Metode yang pertama membuat form kosong.
- Metode yang kedua akan berisi komponen Item yang didefinisikan di array item.

Form dan komponen Item

- Berikut adalah komponen Item:
  - DateField
  - Gauge
  - StringItem
  - TextField
  - ChoiceGroup
  - Spacer
  - Image and ImageItem

Komponen TextField

- TextField serupa dengan text masukan field.



Komponen TextField

- Kita dapat menentukan label, jumlah maks karakter, tipe data yang akan diterima.
- Komponen TextField juga mengimplementasikan password yang berupa mask character ketika diinputkan.
- Konstruktor untuk TextField:
  - TextField(String label, String text, int maxSize, int constraints)
- Parameter **constraints** untuk menentukan tipe input yang diijinkan dalam TextField.
- Gunakan metode **getString()** untuk mendapatkan data dari text yang diinputkan.

Format penulisan

TextField (String Title, String Text, int maxSize,int const)

Contoh :

```
TextField nama = new TextField ("Nama Anda ","", 15,
    TextField.ANY);
```

Constraints:

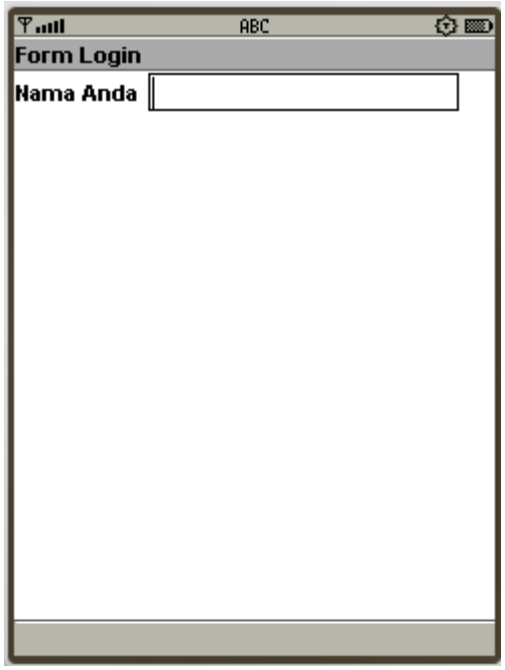
TextField.ANY : sembarang karakter teks

TextField.NUMERIC : teks berupa angka saja

TextField.PASSWORD : dengan karakter tertentu ‘*’

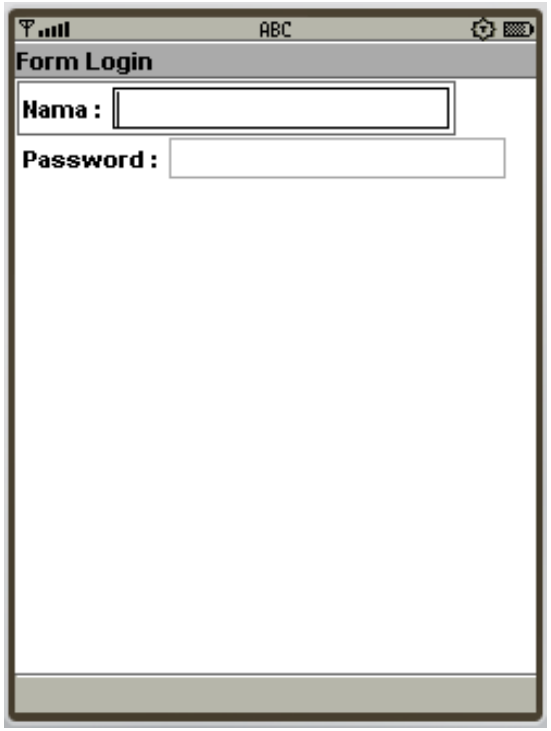
Contoh aplikasi TextField

```
public class TextDemo extends MIDlet {
    public TextDemo() {
    }
    public void startApp() {
        TextField nama = new TextField ("Nama Anda ","", 15, TextField.ANY);
        Form f = new Form( "Form Login" );
        f.append(nama);
        Display.getDisplay(this).setCurrent( f );
    }
    public void pauseApp() {
    }
    public void destroyApp( boolean unconditional ) {
    }
}
```

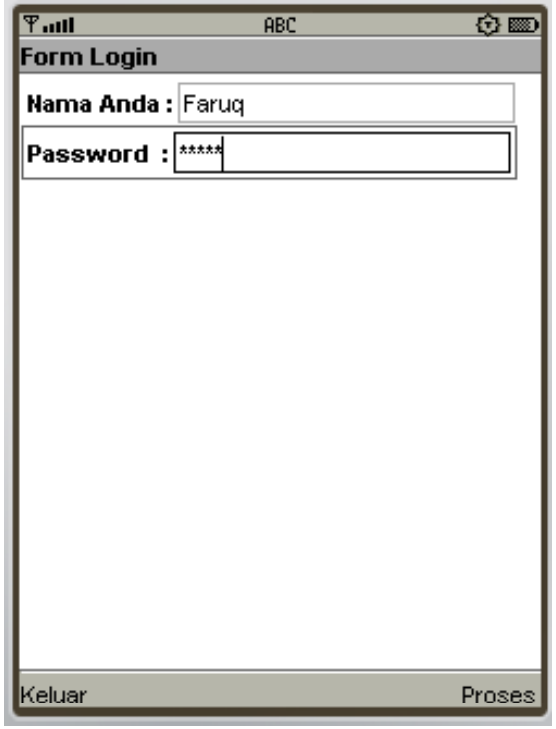


Contoh aplikasi TextField

```
public class passDemo extends MIDlet {
    public passDemo() {
    }
    public void startApp() {
        String s = "Halo faruq";
        TextField nama = new TextField ("Nama : ","", 15, TextField.ANY);
        TextField pass = new TextField ("Password : ","", 15, TextField.PASSWORD);
        Form f = new Form( "Form Login" );
        f.append(nama);
        f.append(pass);
        Display.getDisplay(this).setCurrent( f );
    }
    public void pauseApp() {
    }
    public void destroyApp( boolean unconditional ) {
    }
}
```



Passing Parameter dgn TextField



Passing Parameter dgn TextField

```
public class pass2 extends MIDlet implements CommandListener {
   Command keluarCmd = new Command ("Keluar",Command.EXIT,1);
   Command prosesCmd = new Command ("Proses",Command.OK,2);
   Command kembaliCmd = new Command ("Kembali",Command.BACK,2);
   Form f;                 //agar bisa diakses di semua fungsi
   TextField tnama;        //agar bisa diakses di semua fungsi
   TextField tpass;        //agar bisa diakses di semua fungsi
   public pass2() {
   }
   public void startApp() {
      tnama = new TextField ("Nama Anda :","", 15, TextField.ANY);
      tpass = new TextField ("Password  :","", 15, TextField.PASSWORD);
      f = new Form( "Form Login" );
      f.append(tnama);
      f.append(tpass);
      //Menambahkan obyek command
      f.addCommand(keluarCmd);
      f.addCommand(prosesCmd);
      //Menghubungkan dengan perintah CommandListener
      f.setCommandListener(this);
      Display.getDisplay(this).setCurrent( f );
   }
```

```
public void commandAction (Command c, Displayable d)
{
   String data = c.getLabel();
   System.out.println (data);
   if (data == "Keluar") {
      keluar();
   } else if (data == "Proses") {
      proses();
   } else if (data == "Kembali") {
      awal();
   }
}
```

```
public void proses() {
   String dtnama = tnama.getString();
   String dtpass = tpass.getString();
   Form pf = new Form ("Proses Data");
   pf.append ("Nama anda : ");
   pf.append (dtnama);
   pf.append ("\n");
   pf.append ("Password : ");
   pf.append (dtpass);
   pf.addCommand (kembaliCmd);
   pf.setCommandListener(this);
   Display.getDisplay(this).setCurrent( pf );
}
```

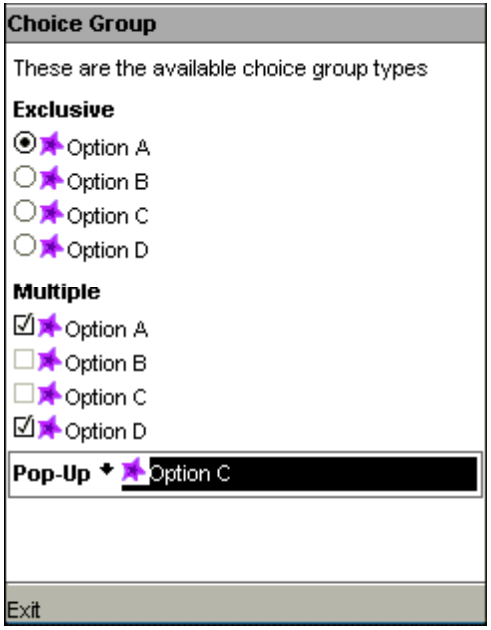
Komponen ChoiceGroup

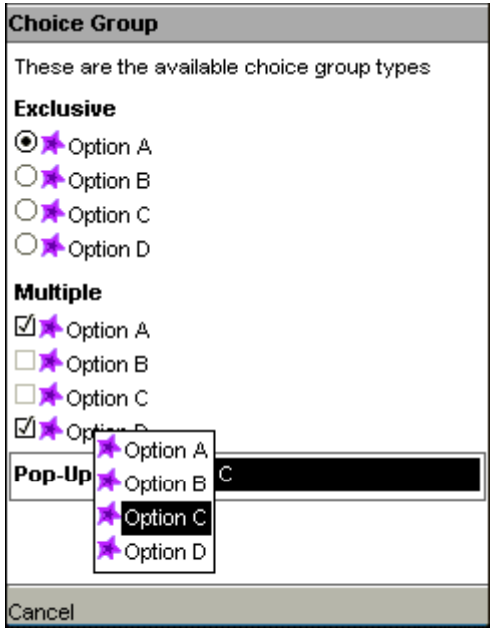
- Memberikan user pilihan dari list yang tersedia.
- Terdapat 3 format ChoiceGroup :

  EXCLUSIVE : satu pilihan

  MULTIPLE : banyak pilihan

  POPUP : satu pilihan dengan tampilan popup





Komponen ChoiceGroup

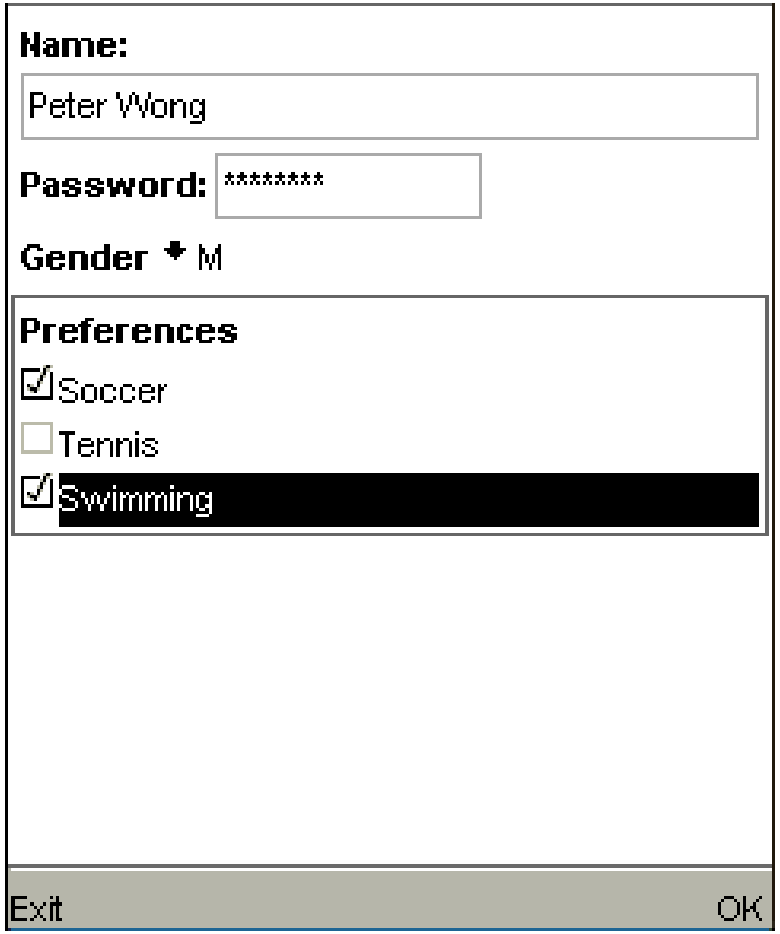
- Format:
  - ChoiceGroup(String label, int choiceType)

    Membuat obyek ChoiceGroup dan menentukan title dan tipenya.
  - ChoiceGroup(String label, int choiceType, String[] stringElements, Image[] imageElements)

    Membuat obyek ChoiceGroup, menentukan title dan tipe serta array dari string dan gambar untuk nilai awal.
- Untuk passing parameter dari pilihan user gunakan getSelectedIndex(), getString(int elementNum) and isSelected(int elementNum).

Contoh TextFieldChoiceGroupTest

- Form Registrasi untuk memasukkan data pribadi dan menampilkannya di dalam console.



Untuk Satu Pilihan (EXCLUSIVE)

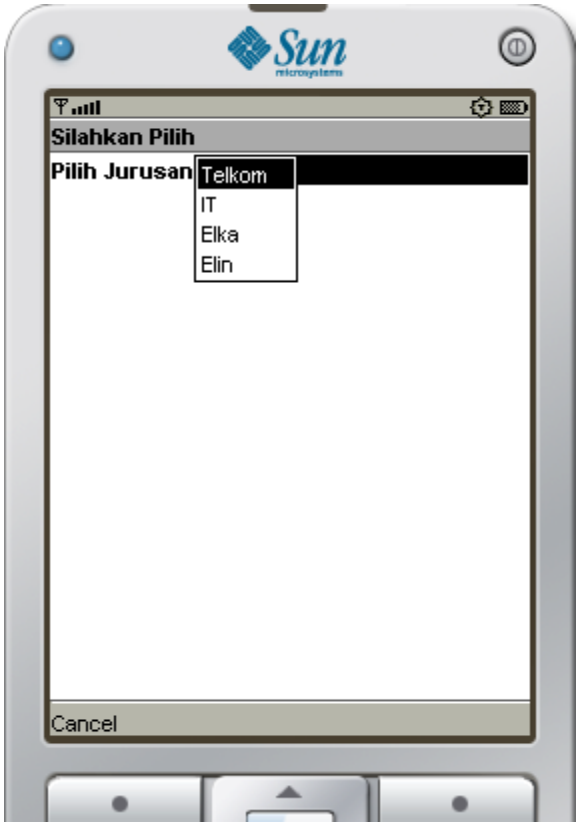
```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class radioDemo extends MIDlet {
    public radioDemo() {
    }
    public void startApp() {
        ChoiceGroup cg = new ChoiceGroup ("Pilih Jurusan",Choice.EXCLUSIVE);
        Form f = new Form( "Silahkan Pilih" );
        cg.append("Telkom",null);
        cg.append("IT",null);
        cg.append("Elka",null);
        cg.append("Elin",null);
        f.append(cg);
        Display.getDisplay(this).setCurrent( f );
    }
    public void pauseApp() {
    }
    public void destroyApp( boolean unconditional ) {
    }
}
```

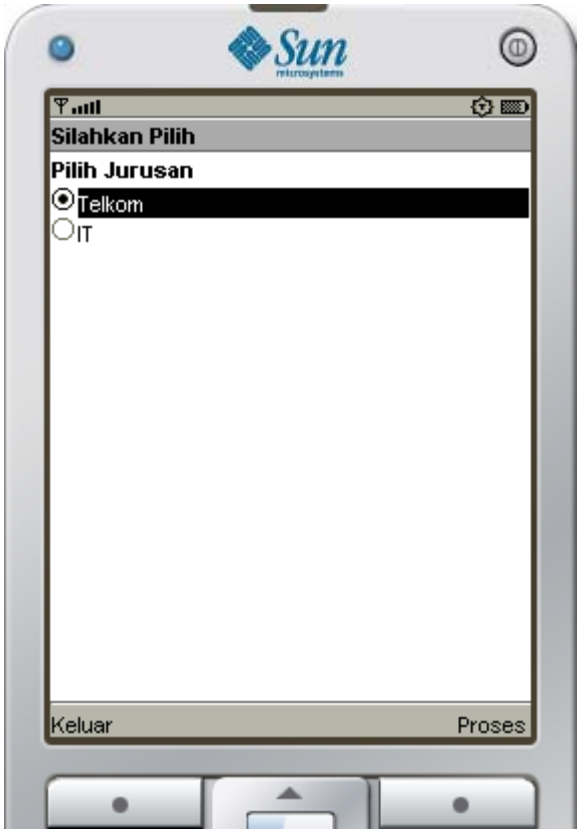
Untuk Satu Pilihan (POPUP)

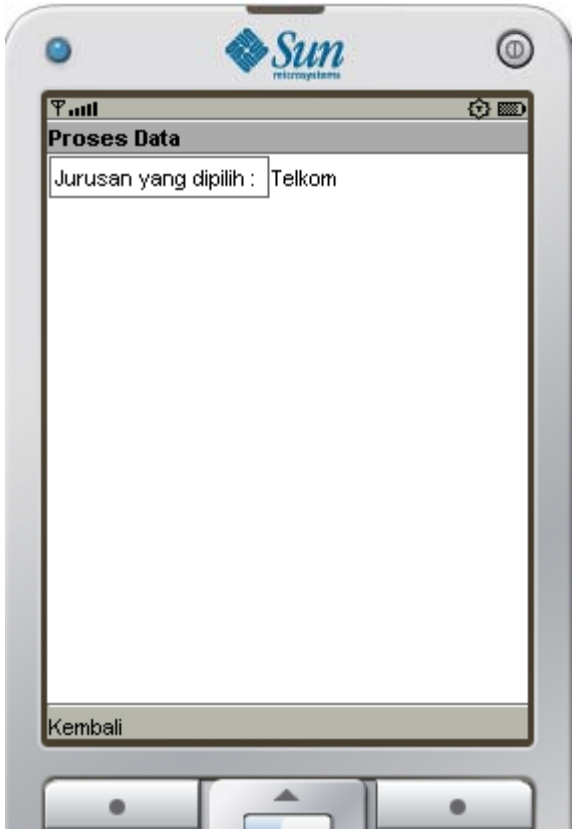
```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class radioDemo extends MIDlet {
    public radioDemo() {
    }
    public void startApp() {
        ChoiceGroup cg = new ChoiceGroup ("Pilih Jurusan",Choice.POPUP);
        Form f = new Form( "Silahkan Pilih" );
        cg.append("Telkom",null);
        cg.append("IT",null);
        cg.append("Elka",null);
        cg.append("Elin",null);
        f.append(cg);
        Display.getDisplay(this).setCurrent( f );
    }
    public void pauseApp() {
    }
    public void destroyApp( boolean unconditional ) {
    }
}
```



Passing Parameter dgn RadioButton





Passing Parameter dgn RadioButton

```
public class radio2 extends MIDlet implements CommandListener {
   private Command keluarCmd = new Command ("Keluar",Command.EXIT,1);
   private Command prosesCmd = new Command ("Proses",Command.OK,2);
   private Command kembaliCmd = new Command ("Kembali",Command.BACK,2);
   private Form f;                //agar bisa diakses di semua fungsi
   private ChoiceGroup cg;        //agar bisa diakses di semua fungsi
   public void startApp() {
      cg = new ChoiceGroup ("Pilih Jurusan",Choice.EXCLUSIVE);
      f = new Form( "Silahkan Pilih" );
      cg.append("Telkom",null);
      cg.append("IT",null);
      f.append(cg);
      //Menambahkan obyek command
      f.addCommand(keluarCmd);
      f.addCommand(prosesCmd);
      //Menghubungkan dengan perintah CommandListener
      f.setCommandListener(this);
      Display.getDisplay(this).setCurrent( f );
   }
```

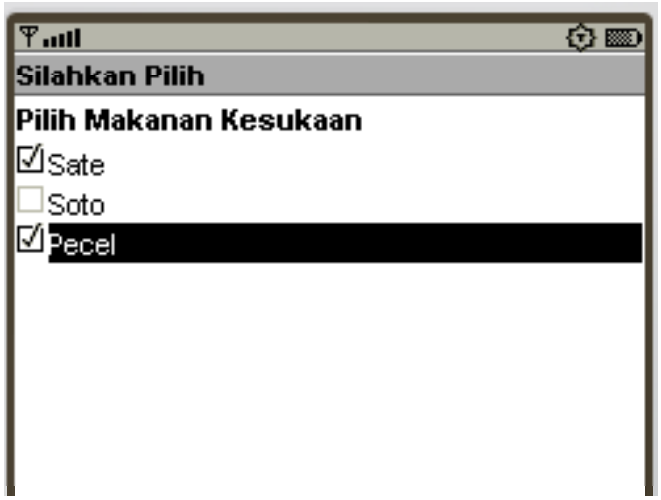
```
public void proses() {
   Form pf = new Form ("Proses Data");
   pf.append ("Jurusan yang dipilih : ");
   if (cg.isSelected(0))
      pf.append (cg.getString(0));
   else if (cg.isSelected(1))
      pf.append (cg.getString(1));
   pf.addCommand (kembaliCmd);
   pf.setCommandListener(this);
   Display.getDisplay(this).setCurrent( pf );
}
```

```
public void commandAction (Command c,Displayable d){
   String data = c.getLabel();
   if (data == "Keluar") {
      keluar();
   } else if (data == "Proses") {
      proses();
   } else if (data == "Kembali") {
      awal();
   }
}
```

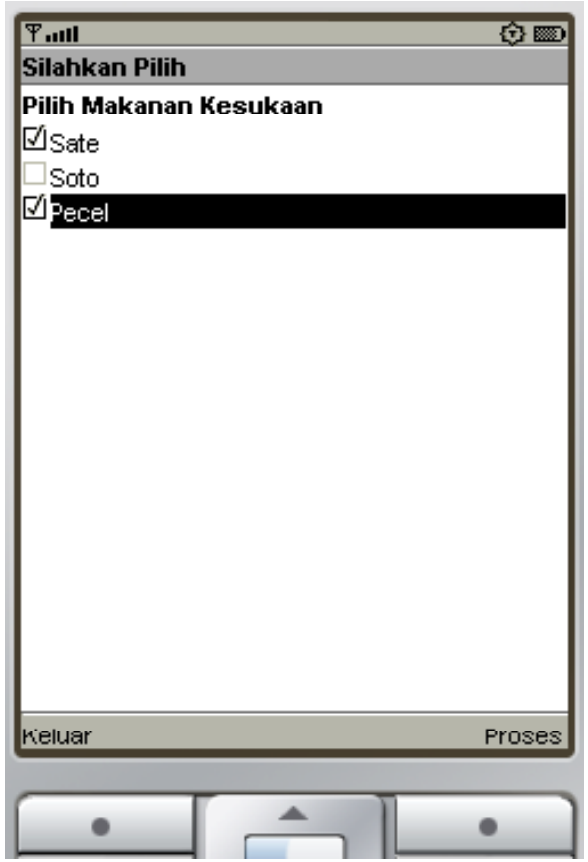
Banyak Pilihan (MULTIPLE)

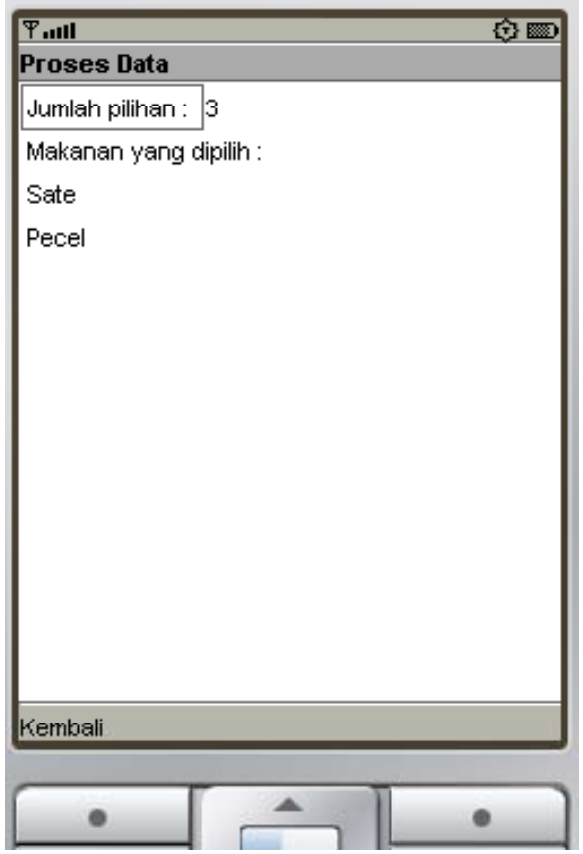
```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class checkboxDemo extends MIDlet {
    public checkboxDemo() {
    }
    public void startApp() {
        ChoiceGroup cg = new ChoiceGroup ("Pilih Makanan Kesukaan",Choice.MULTIPLE);
        Form f = new Form( "Silahkan Pilih" );
        cg.append("Sate",null);
        cg.append("Soto",null);
        cg.append("Pecel",null);
        f.append(cg);
        Display.getDisplay(this).setCurrent( f );
    }
    public void pauseApp() {
    }
    public void destroyApp( boolean unconditional ) {
    }
}
```



Passing Parameter dgn CheckBox





Aplikasi untuk CheckBox

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

public class checkbox2 extends MIDlet implements CommandListener {
    private Command keluarCmd = new Command ("Keluar",Command.EXIT,1);
    private Command prosesCmd = new Command ("Proses",Command.OK,2);
    private Command kembaliCmd = new Command ("Kembali",Command.BACK,2);
    private Form f;                 //agar bisa diakses di semua fungsi
    private ChoiceGroup cg;         //agar bisa diakses di semua fungsi
    public checkbox2() {
    }
    public void startApp() {
      cg = new ChoiceGroup ("Pilih Makanan Kesukaan",Choice.MULTIPLE);
      f = new Form( "Silahkan Pilih" );
      cg.append("Sate",null);
      cg.append("Soto",null);
      cg.append("Pecel",null);
      f.append(cg);
      //Menambahkan obyek command
      f.addCommand(keluarCmd);
      f.addCommand(prosesCmd);
      //Menghubungkan dengan perintah CommandListener
      f.setCommandListener(this);
      Display.getDisplay(this).setCurrent( f );
    }
```

- Untuk fungsi pauseApp(), destroyApp(), awal(), keluar() sama dengan program sebelumnya

```java
public void proses() {
    String s;
    Form pf = new Form ("Proses Data");
    pf.append ("Jumlah pilihan : ");
    s = String.valueOf(cg.size());
    pf.append (s);
    pf.append ("\n");
    pf.append ("Makanan yang dipilih : ");
    pf.append ("\n");
    for (int i=0;i < cg.size();i++)
      if (cg.isSelected(i)) {
          pf.append (cg.getString(i));
          pf.append ("\n");
      }
    pf.addCommand (kembaliCmd);
    pf.setCommandListener(this);
    Display.getDisplay(this).setCurrent( pf );
}
public void commandAction (Command c, Displayable d) {
    String data = c.getLabel();
    System.out.println (data);
    if (data == "Keluar") {
        keluar();
    } else if (data == "Proses") {
        proses();
    } else if (data == "Kembali") {
        awal();
    }
}
```