Mobile Information Device Programming (6)

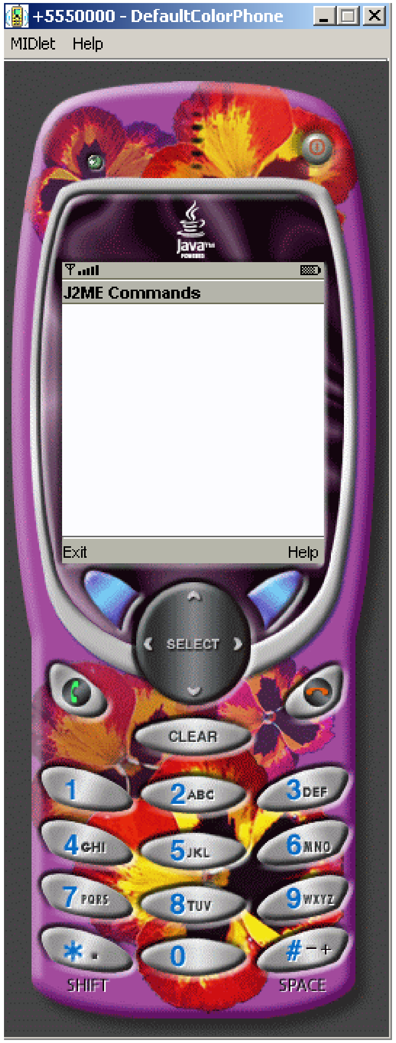
Lecturer: Alireza Mousavi
School of Engineering & Design
www.brunel.ac.uk/~emstaam

Commands and Buttons

- How to map commands to buttons

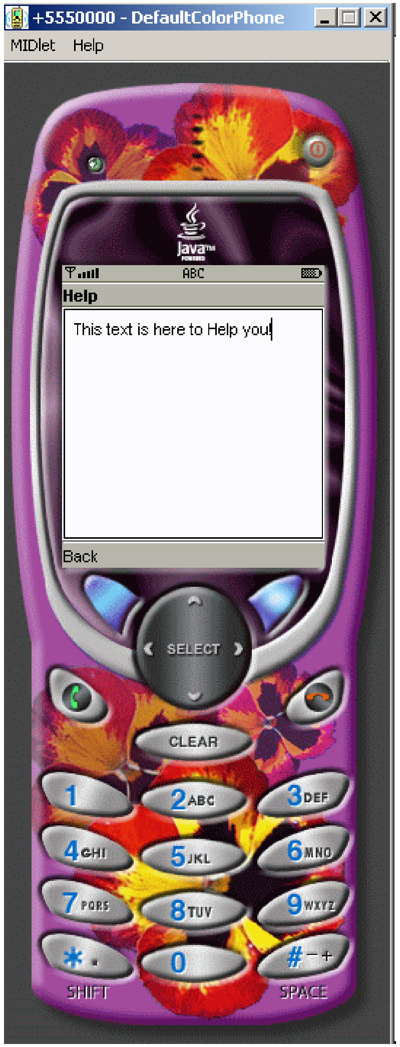
```
private Command cmExit;
private Command cmBack
Private Command cmHelp;
…
cmExit = new Command(“Exit”, Command.EXIT, 1);
cmHelp = new Command(“Help”, Command.HELP, 1);
cmBack = new Command(“Back”, Command.Back, 1);
```

What you see ...



+5550000 - DefaultColorPhone

J2ME Commands



+5550000 - DefaultColorPhone

Help

This text is here to Help you!

An Example

- Create a Midlet with three commands “Exit”, “Help” and “Back”

- Add a TextBox to your application with a text displaying: *“this is the help text!”*

- Check if the “Help” button is selected if so display the text, if “Exit” button is selected exit the application

E6.1 - CommandMap.java

Results

- Look at slide 2 the result on the device

Check this:

```
public void commandAction(Command c, Displayable s){
    if (c == Exit){
      destroyApp(false);
      notifyDestroyed();
    }
    else if (c ==Help)
      myDisplay.setCurrent(tbHelp);
    else if (c == Back)
      myDisplay.setCurrent(Main);
     }
}
```

Add some more commands **(Assignment)**

- Add a TextBox called tbAction
- The text in the TextBox to be *"Processing Data Please Wait"*
- Add *"Upload"* and *DownLoad"* commands e.g. *cmUload* and *cmDload*
- Check if "*DownLoad"* or *"Upload* is selected then show *tbAction*
  – *If (c == cmUload ¦¦ c == cmDload)*

E6.2 - MoreCommand

Item and ItemStateListener

- An **Item** is any component that can be added to a **Form**

- The MIDP library contains the following Items:

  ChoiceGroup, DateField, Gauge, ImageItem, StringItem, and TextField

- When you add an **Item** to a Form you also add a **Listener** to capture events

- Once a change occurs the ItemStateChange( ) method is called

ItemStateChanged( ) method

Within this method you can define which **Item** was changed and what needs to be done.

Create an Item

Item is an abstract class – meaning we do not need to create instances of the **Item** class.

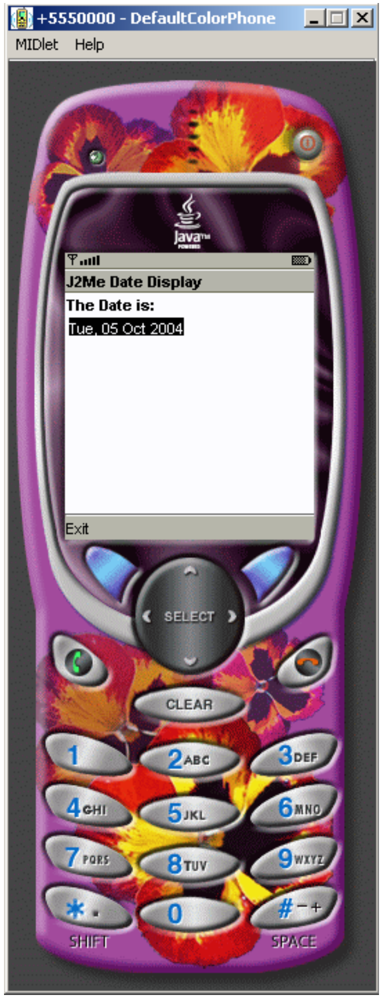
Example: create a DateField, add it a Form and set a Listener so the Form (Displayable object) can detect events.

Example

```
private Form fmMainForm; // declare a Form
private DateField dfDay;  // declare a DateField
 …
// create a date and print the current date
dfDay= new DateField(“Today is:”, DateField.DATE);
dfDay.setDate(new java.util.Date( ));
fmMainForm = new Form(“Date”);
fmMainForm.append(dfDay);
…
// listen for events
fmMainForm.setCommandListener(this);
```

E6.3 – EventCapture.java

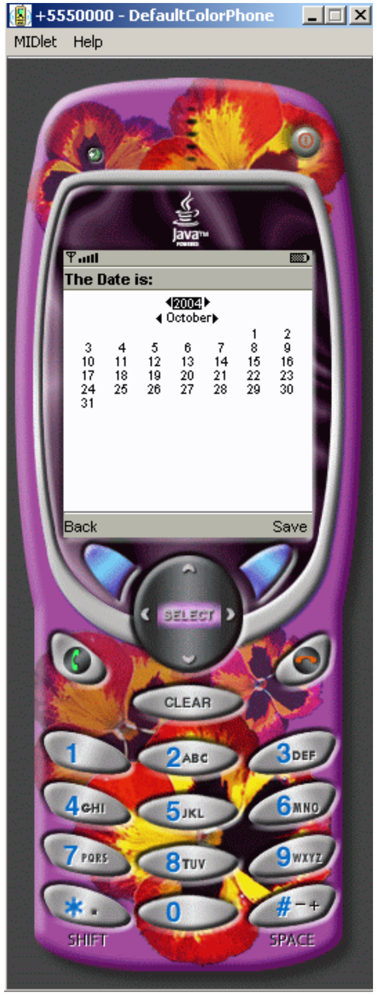
Results



J2Me Date Display

The Date is:

Tue, 05 Oct 2004



The Date is:

```
          ◂2004▸
        ◂October▸
                          1   2
 3   4   5   6   7   8   9
10  11  12  13  14  15  16
17  18  19  20  21  22  23
24  25  26  27  28  29  30
31
```