Mobile Information Device Programming (8)

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

StringItem

- A **StringItem** displays a static label or text message
- A **StringItem** does not recognise events, it therefore cannot be edited.
- You can use the *getText( ) or setText( )* methods inherited for the **Item** class

Example Changing Label and Message Text E8.1

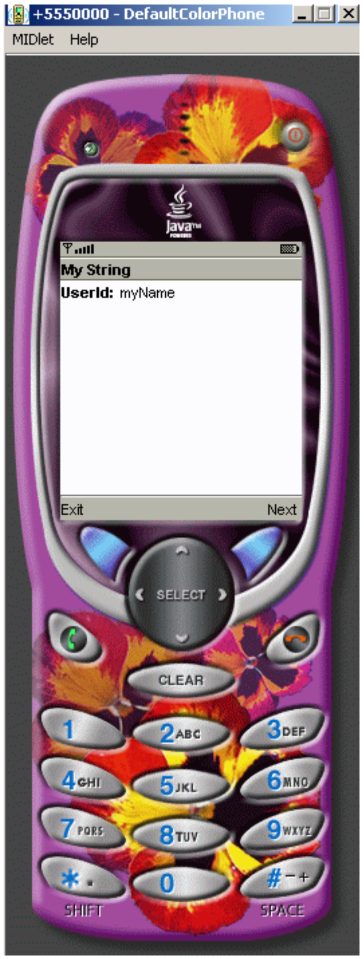
- Create a Java file called *myLabelText.java*

- Create a *Display, Form, StringItem,* and two *Command "Exit"* and "*Next"*

- Use the *setLabel( )* and *setText ( )* methods to set the text:

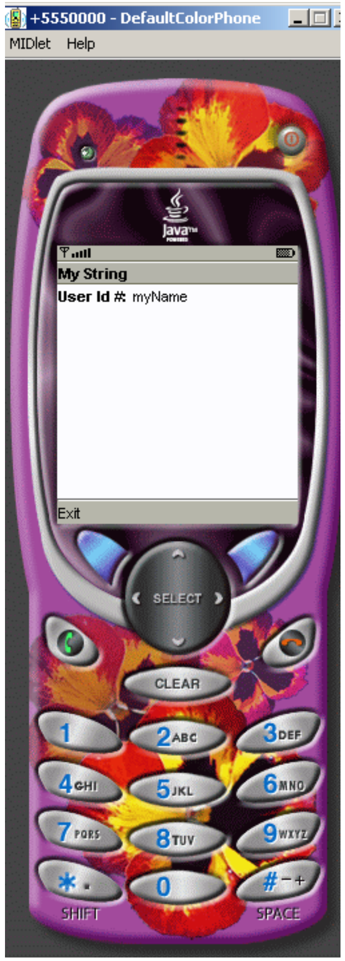
```
siUser.setLabel(“My Number: ”);
siUser.setText(“123”);
```

Result


+5550000 - DefaultColorPhone
MIDlet Help

My String
UserId: myName

Exit Next


+5550000 - DefaultColorPhone
MIDlet Help

My String
User Id #: myName

Exit

Some more explanation

- In the constructor *myLabelText* we define a **StringItem** that contains a label and a text

- We also added a Command *cmNext* which invoked a call to CommandAction( ), where we change the label and the text.

Another way

- You can directly *append* a **String** to a Form.

- There are no Labels like **StringItem** associated with the text

Method Example: *msgID = fmMain.append(“User Id: “John_Smith”)*

Example E8.2

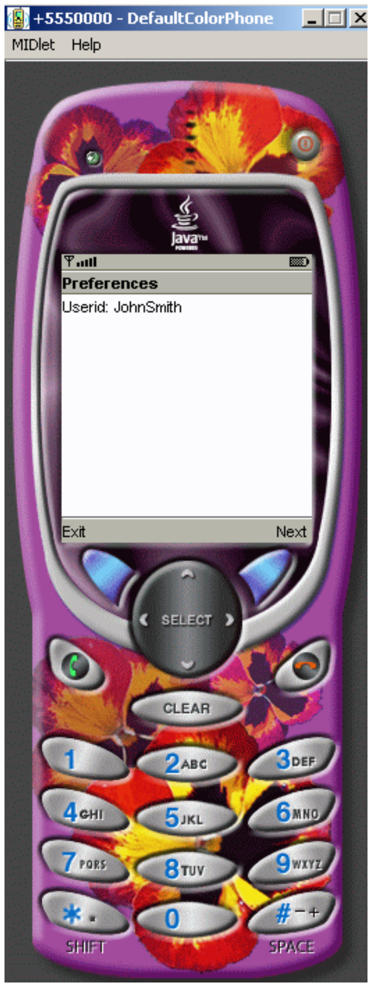
- Create a *StringText.java* file
- Create a *Display, Form, 2 Commands "Next" and "Exit",*
- Create two integers *int msgIndex, count*
- Append a string text *"UserId: JohnSmith"* to the Form: *msgIndex = fmMain.append("UserId: JohnSmith");*
- In your commandAction(Command c, Displayable s){

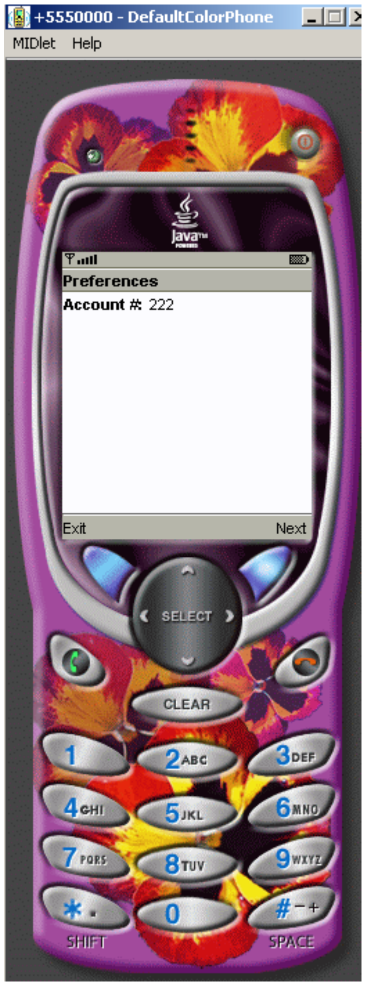
…} you will have two options

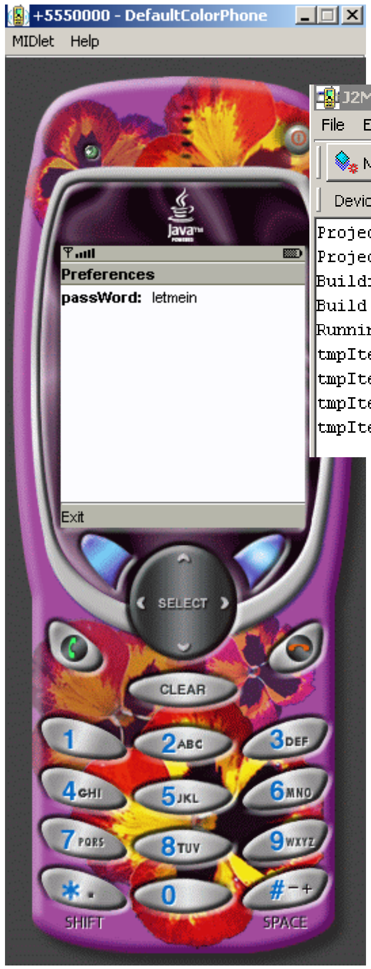
E8.2 cont.

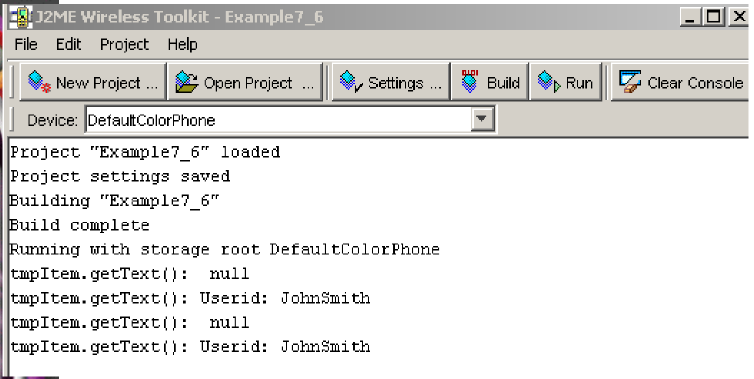
```
public void commandAction(Command c, Displayable s){
  if(c == cmNext){ // check if the chosen command is Next
    if( count++ == 0){ // and check if loop counter is 0
      /* ---------------------------------------------
         option 1
         This is the first Time through this method*/
      StringItem tmpItem = (StringItem) fmMain.get(msgIndex);
      System.out.println("tmpItem.getText():  " + tmpItem.getLabel());
      // inherited from item class
      tmpItem.setLabel("Account #: ");
      tmpItem.setText("222");
    }
    else {
      /* Option 2
         Second time through this method*/
      fmMain.set(msgIndex, new StringItem("passWord:  ", "letmein"));
      // Remove the Next command
      fmMain.removeCommand(cmNext);
    }
  }
  else if (c == cmExit){
    destroyApp(false);
    notifyDestroyed();
  }
}
```

Result









```
Project "Example7_6" loaded
Project settings saved
Building "Example7_6"
Build complete
Running with storage root DefaultColorPhone
tmpItem.getText():  null
tmpItem.getText(): Userid: JohnSmith
tmpItem.getText():  null
tmpItem.getText(): Userid: JohnSmith
```

TextField

- A **TextField** provides constraints on the data that a user can enter
- The code behind a **TextField** can help with limiting the characters that can be used
- In addition to character constraints **TextField** i.e. number of acceptable characters

TextField Constraints

Value	Description
Constraint_Mask	Used to determine the current value of the constrain
ANY	Allows any Character
EMAILADDR	Allows only valid email Characters
NUMERIC	Allows only numeric values
PASSWORD	Masks all characters to allow privacy
PHONENUMBER	Allows characters valid for phone numbers
URL	Allows only characters valid to point to a URL

TextField cont.

```
TfPwd = new TextField(“Password: “, “ “, 10, TextField.ANY ¦
TextField.PASSWORD);
```

```
tfPhone = new TextField(“Phone No:”, “ “, 15, TextField.PHONENUMBER);
```