



# **Mobile Information Device Programming (7)**

Lecturer: Alireza Mousavi  
School of Engineering & Design  
[www.brunel.ac.uk/~emstaam](http://www.brunel.ac.uk/~emstaam)



# High-Level User Interface

- Screen
- Form
- Item
- DateField
- Gauge
- StringItem
- TextField
- Choice and ChoiceGroup
- Image and ImageItem



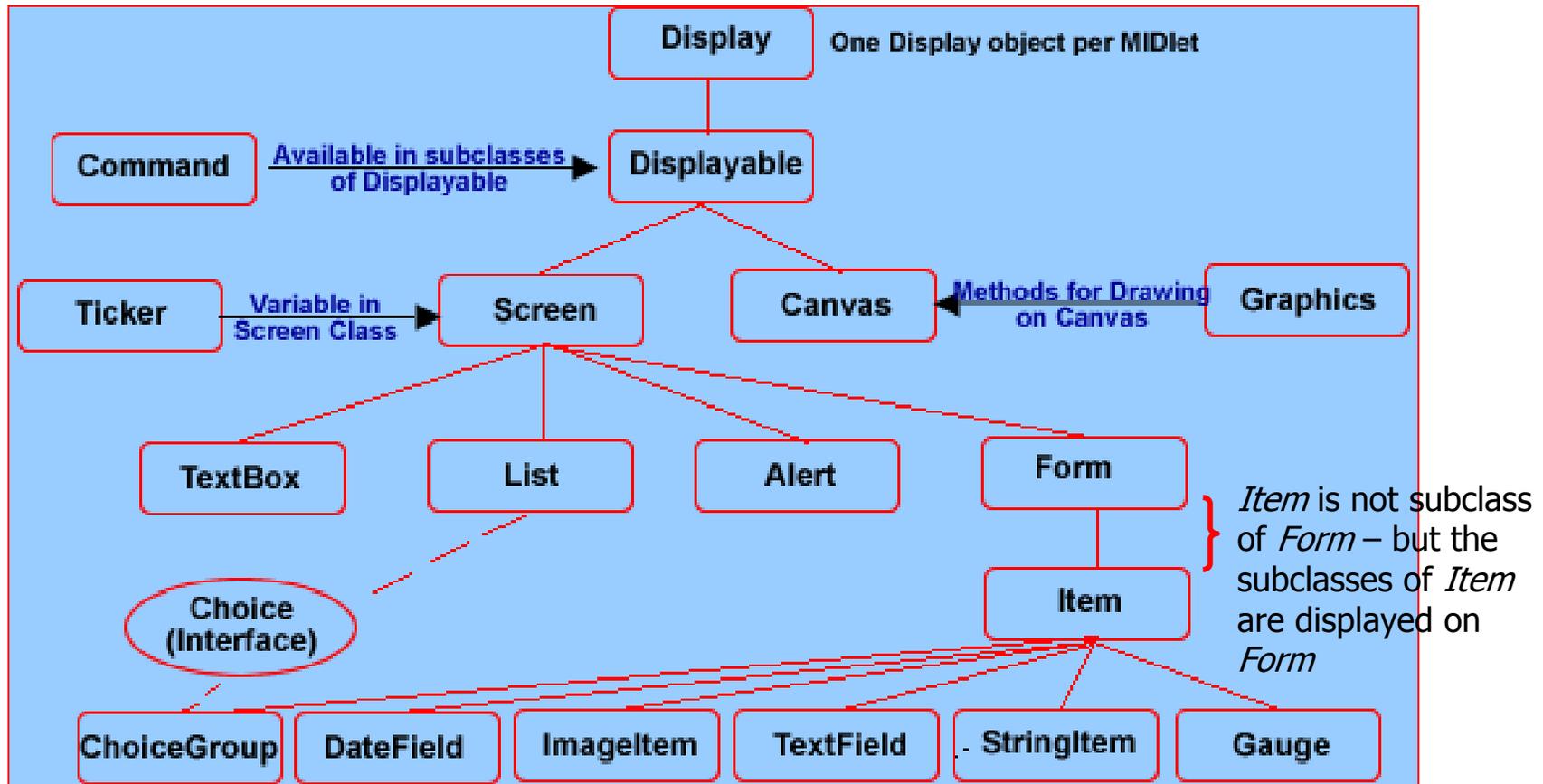
# Screen

- **Screen** is an abstract class and descendent of **Displayable** class
- **Screen** and its subclasses are for high-level UI
- **Screen** is the parent class containing components that can be seen on the display.
- **Screen** itself is not visible the components within it are.

*Also see MIDP5 lecture notes*



# Displayable class Hierarchy [Machow 2002]





# Methods available to Screen

## Screen API

Screen Class: `javax.microedition.lcdui.screen`

<i>Method</i>	<i>Description</i>
String <i>getTitle( )</i>	Get the title of the Screen
void <i>setTitle ( )</i>	Set title for the Screen
Ticker <i>getTicker( )</i>	Get Ticker associated with the Screen
void <i>setTicker( )</i>	Set Ticker for the Screen



# Form

- Handheld devices have limitations on display
- Form is used to manage these limitations and utilise the capabilities
- Forms provide with:
  - Multiple components (Item subclasses)
  - Scrolling facilities
  - Methods to append, insert, replace and delete components



# Item

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

Example: ChoiceGroup, DateField, ImageItem,  
StringItem and TextField

The class associated with an **Item** is *ItemStateListener*.

The method associated with **Item** is *ItemStateChanged( )*,  
using this method you can specify the changed **Item**  
and determine the action taken.



## Example for Item

The code below appends an Item to a form and creates a listener for processing:

```
private Form ItemForm; // a Form declaration
private DateField dfDate; // a DateField declaration
...
ItemForm = new ItemForm("My First Item"); // create the form Object
dfDate = new DateField("Today is:", DateField.DATE); // Create the Datefield
...
ItemForm.append(dfDate); // add item to the form
ItemForm.setItemListener(this); // listen for events in this midlet
...
public void ItemStateChanged(Item item) {
    If (item ==dfDate)
...
}
```



# Exercise

This exercise aims at creating various items on a form.

This specific example when uploaded gives a date provides you with an ordering system which will take the day and the quantity of product to be ordered.

- Create a Java File called "FormItems":
  1. Create a Java class called *FormItems*
  2. Create a form called *Main*
  3. An Insert "*cmInsert*" and an Exit "*cmExit*" Command
  4. Create and add DateField *dfDate* and two TextField *tfSize* and *tfQuant*
  5. Create an integer value called DateIndex



# Result





# Gauge

- A **Gauge** is a method of showing progress of something for example ringer volume, downloading percentage etc.
- Should you need to develop a Gauge for your device UI you can use the **Gauge** component
- There are two types of gauges *Interactive* and *non-Interactive*

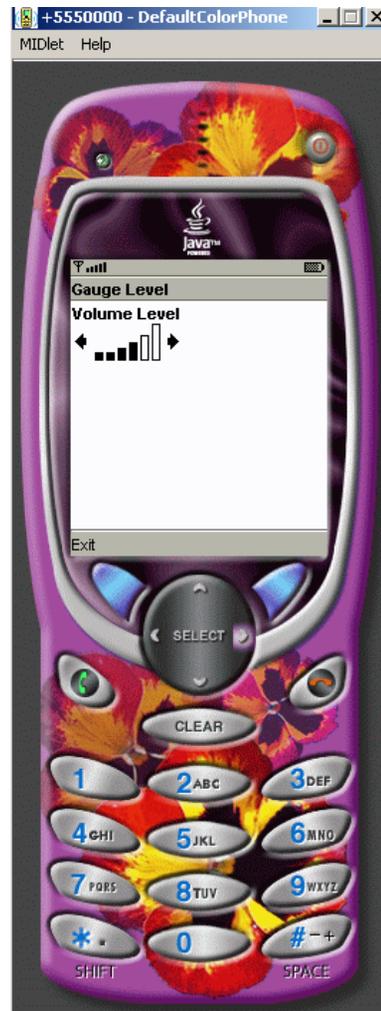


# Example

- Create an application where the user can increase and decrease sound level on their device:
  1. Create a Java file called **InterGauge**
  2. Create the InterGauge constructor
  3. Create a form
  4. Add an exit command
  5. Add a Gauge *gaVol*
  6. Run the application



# Result





# Non Interactive Gauge

## (Assignment)

- Create a non-Interactive Gauge – use a **Timer** to provide your MIDlet with periodic interruptions where the gauge is automatically incrementing.
  1. Create a **NonIntGauge** Java file
  2. Create a Form
  3. Add Exit and Stop commands to the form
  4. Add a Gauge *gaProg*
  5. Add a Timer *tm*
  6. Add DownloadTimer *tt* for the task to run.
  7. Use the *scheduleAtFixedRate(tt, 0, 1000)* method to set the timer off every 1 second



# Non Interactive Gauge (Assignment) Cont.

- Add the following class at the end of your application to handle the timer task:

```
/* *****  
    handle the timer task  
    *****/  
private class DownloadTimer extends TimerTask {  
    public final void run( )  
    {  
        // is current value of the gauge equal to the max  
        if (gaProg.getValue( ) - gaProg.getMaxValue( ) < 0)  
            gaProg.setValue(gaProg.getValue( ) + 1) ;  
        else {  
            // Remove stop command and replace with exit  
            fmMain.removeCommand(cmStop);  
            fmMain.addCommand(cmExit);  
            // Change gauge label  
            gaProg.setLabel("DownLoad Complete!");  
            // stop the timer  
            cancel(); }  
    }  
}
```



# Result



Press stop



Left to complete