Mobile Application Development

J2ME - Forms

Dr. Christelle Scharff
cscharff@pace.edu
Pace University, USA

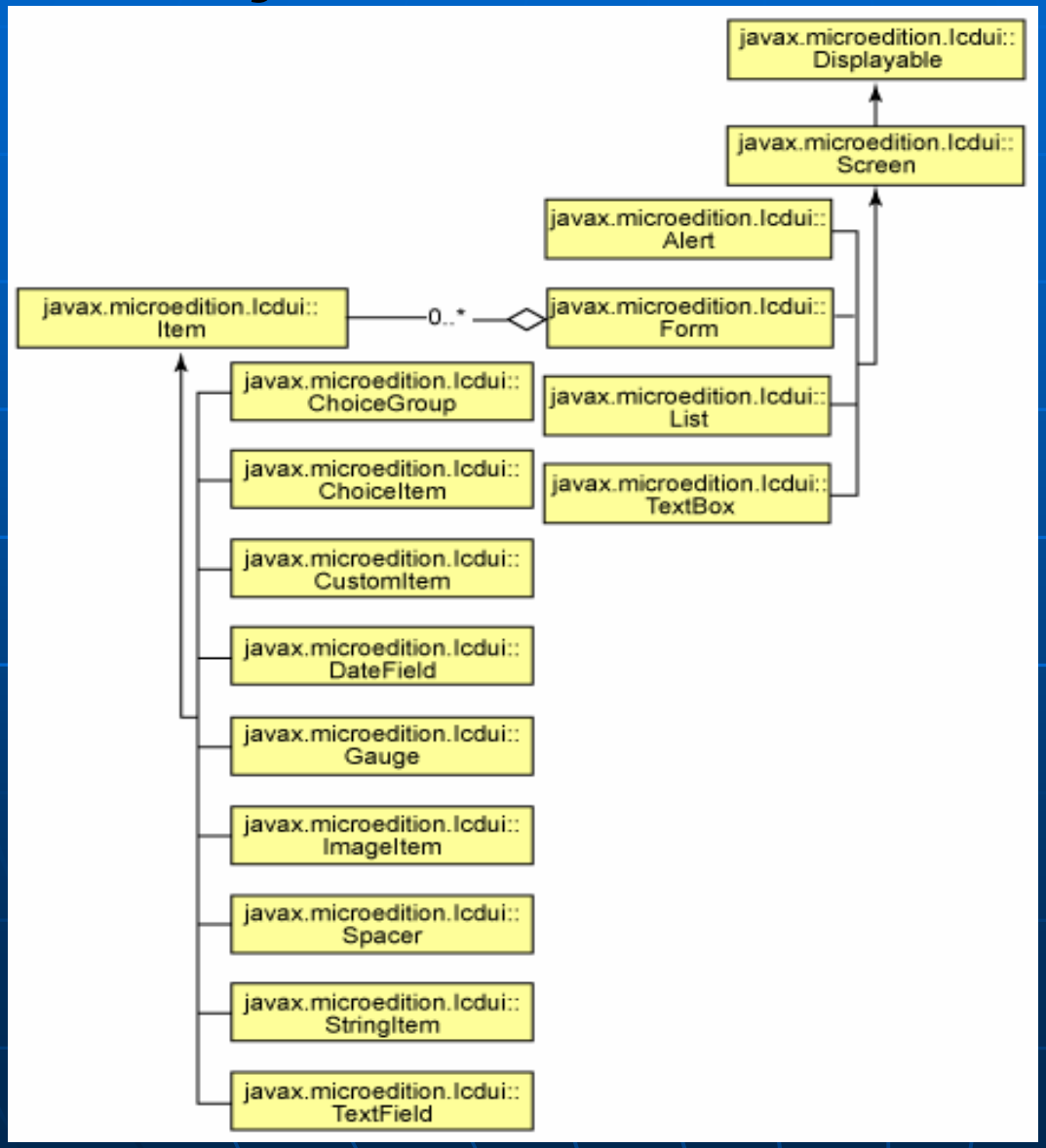
Objectives

- Understand and manipulate:
  - Display
  - Displayable
  - Command
  - Form
  - Items
  - Event handling for Commands
- Write a simple J2ME application based on Forms

GUI

- Do not think about designing mobile phone applications like you are designing web or standard applications!
- A different GUI paradigm
  - Instead of seeing the GUI as multiple windows, the GUI could be seen as a deck of screens when only one screen is active at a given time
- High-level GUI APIs – portable, easy-to-use, little control over the GUI look and feel
- Lower-level GUI APIs – full control of graphics and inputs, based on the Canvas, Graphics, Image and Font classes, low-level drawing with specific additional classes

Major Classes of LCDUI MIDP 2.0



http://www.ibm.com/developerworks/wireless/library/wi-prep/fig8-midp2-hl-lcdui.gif

Display and Displayable

- The `Display` class represents the screen of a device
- The `Displayable` class describes object that can be visible on a Display
- To get the (unique) `Display` of a MIDlet use
  - `public static Display` **`getDisplay`**`(MIDlet m)`
- To request an object to be made visible on the `Display` use
  - `public void` **`setCurrent`**`(Displayable nextDisplayable)`

Display and Displayable

```
public class MyMIDlet extends MIDlet implements
   CommandListener {
    …
   Display myDisplay;
   Form myForm;

    protected void startApp(){
        …
        myDisplay = Display.getDisplay(this);
        …
        myDisplay.setCurrent(myForm);
        …
    }
   …
}
```

Command and CommandListener

- The `Command` class represents an action the user can perform WITHOUT defining the action
- The action is defined in a `CommandListener` associated with the `Displayable`
- Actions are described in the commandAction methods of the `CommandListener` interface that must be implemented by the MIDlet class
  - `public void commandAction(Command arg0, Displayable arg1)`
- The device organizes how the commands are displayed on the screen

Command

- A `Command` is defined by:
  - A short label
  - A long label
  - A type
    - Common commands - `BACK`, `CANCEL`, `EXIT`, `HELP`, `OK`, `STOP`
    - Application-specific commands – `SCREEN`
    - Form-specific commands - `ITEM`
  - A priority – lower priorities are more important

Form

- A `Form` is a screen that contains an arbitrary number of items descendant of the class `Item`:
  - `StringItem`, `TextField`, `ImageItem`, `ChoiceGroup`, `Gauge`, `Spacer`, `DateField` and custom items
  - An item belongs to one form only
- A `Form` can be edited using append, delete, insert and set methods
- Items are referred by their index in the `Form`
- The device handles layout (organized by rows), traversal, and scrolling

Form Operations

- `public int append(Item item)`
- `public void insert(int itemNum, Item item)`
- `public void delete(int itemNum)`
- `public void set(int itemNum, Item item)`
- `public Item get(int itemNum)`

Example

```
public class MyMIDlet extends MIDlet implements CommandListener {
    …
     private Form mForm
     private Command mOKCommand;
     protected void startApp(){
              …
              mForm = new Form("Form and actions");
              mOKCommand = new Command("OK", Command.OK, 0);
              mForm.addCommand(mOKCommand);
              mForm.setCommandListener(this);
              …
     }
     public void commandAction(Command arg0, Displayable arg1){
              if (c == mOKCommand && d == mForm){
                  …
              }
…
}
```

Alert

- The `Alert` class is used to inform the user about errors and other exceptional conditions, but also to display a message to the user
- An `Alert` is defined by:
  - A title
  - A text content
  - An image to be displayed in the Alert
  - An `Alert` type – ALARM, CONFIRMATION, ERROR, INFO and WARNING
- Alerts have a timeout that determines how long the Alert will be displayed on the screen
- Code:

```
Alert a = new Alert("Alert", "This is the message
of the alert", image, AlertType.INFO);
a.setTimeout(3000);
```

Ticker

- The `Ticker` class permits to add text that scrolls across the screen
- The same ticker can be shared by different `Displayable` objects
- Code:

```
f = new Form("Form for Ticker");
f.setTicker(new Ticker("Mobile Programming Summer 2008"));
d = Display.getDisplay(this);
d.setCurrent(f);
```

References

- MIDP Profile API
  - http://java.sun.com/javame/reference/apis/jsr118/
- Kicking Butt with MIDP and MSA - Creating Great Mobile Applications, Jonathan Knudsen, Addison Wesley, Paperback, 2008