Java™ Platform, Micro Edition

Part 2 – High Level UI

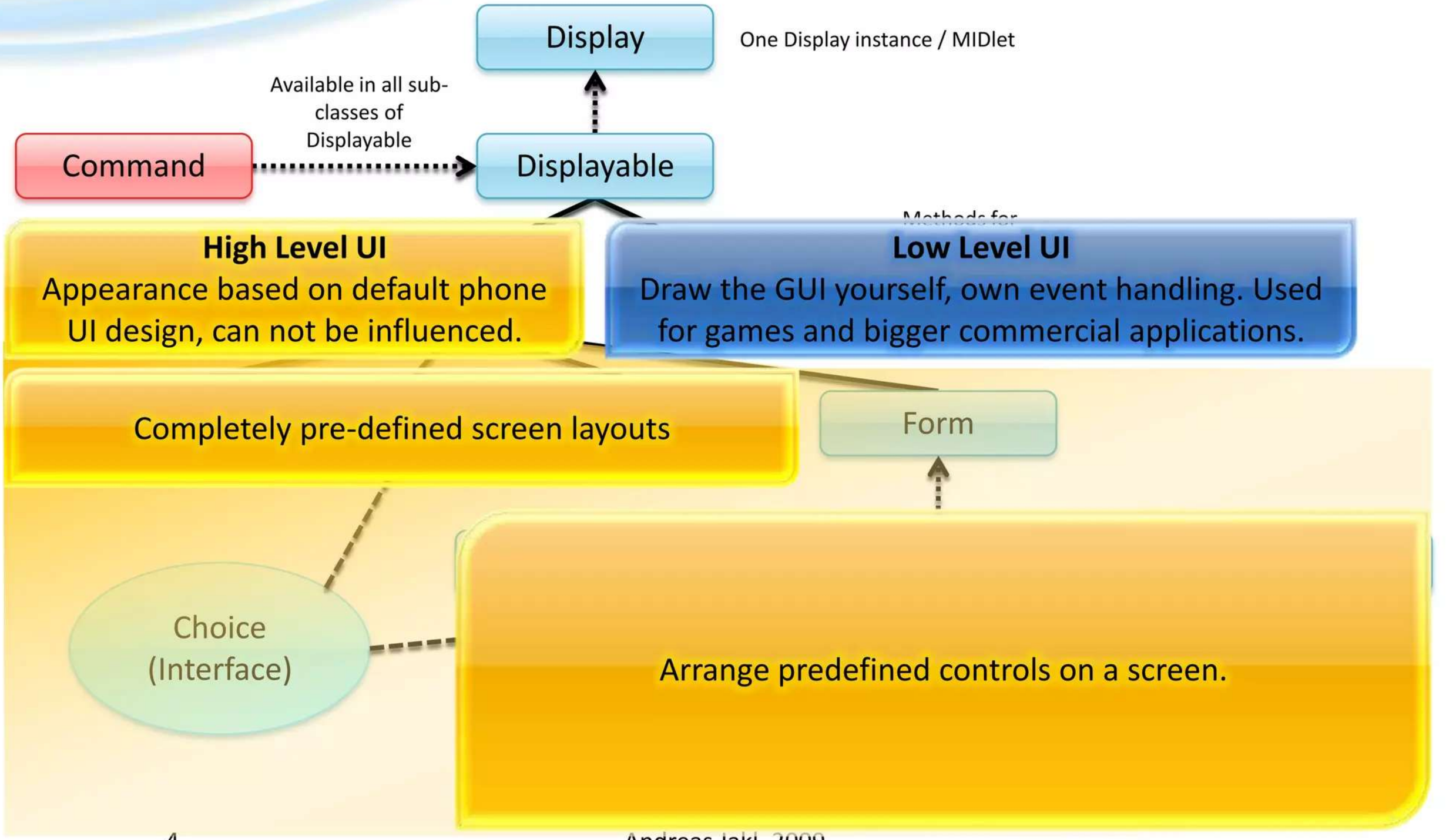
Disclaimer

- **These slides are provided free of charge at http://www.symbianresources.com and are used during Java ME courses at the University of Applied Sciences in Hagenberg, Austria at the Mobile Computing department (http://www.fh-ooe.at/mc)**
- **Respecting the copyright laws, you are allowed to use them:**
  - **for your own, personal, non-commercial use**
  - **in the academic environment**
- **In all other cases (e.g. for commercial training), please contact andreas.jakl@fh-hagenberg.at**
- The correctness of the contents of these materials cannot be guaranteed. Andreas Jakl is not liable for incorrect information or damage that may arise from using the materials.
- This document contains copyright materials which are proprietary to Sun or various mobile device manufacturers, including Nokia, SonyEricsson and Motorola. Sun, Sun Microsystems, the Sun Logo and the Java™ Platform, Micro Edition are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

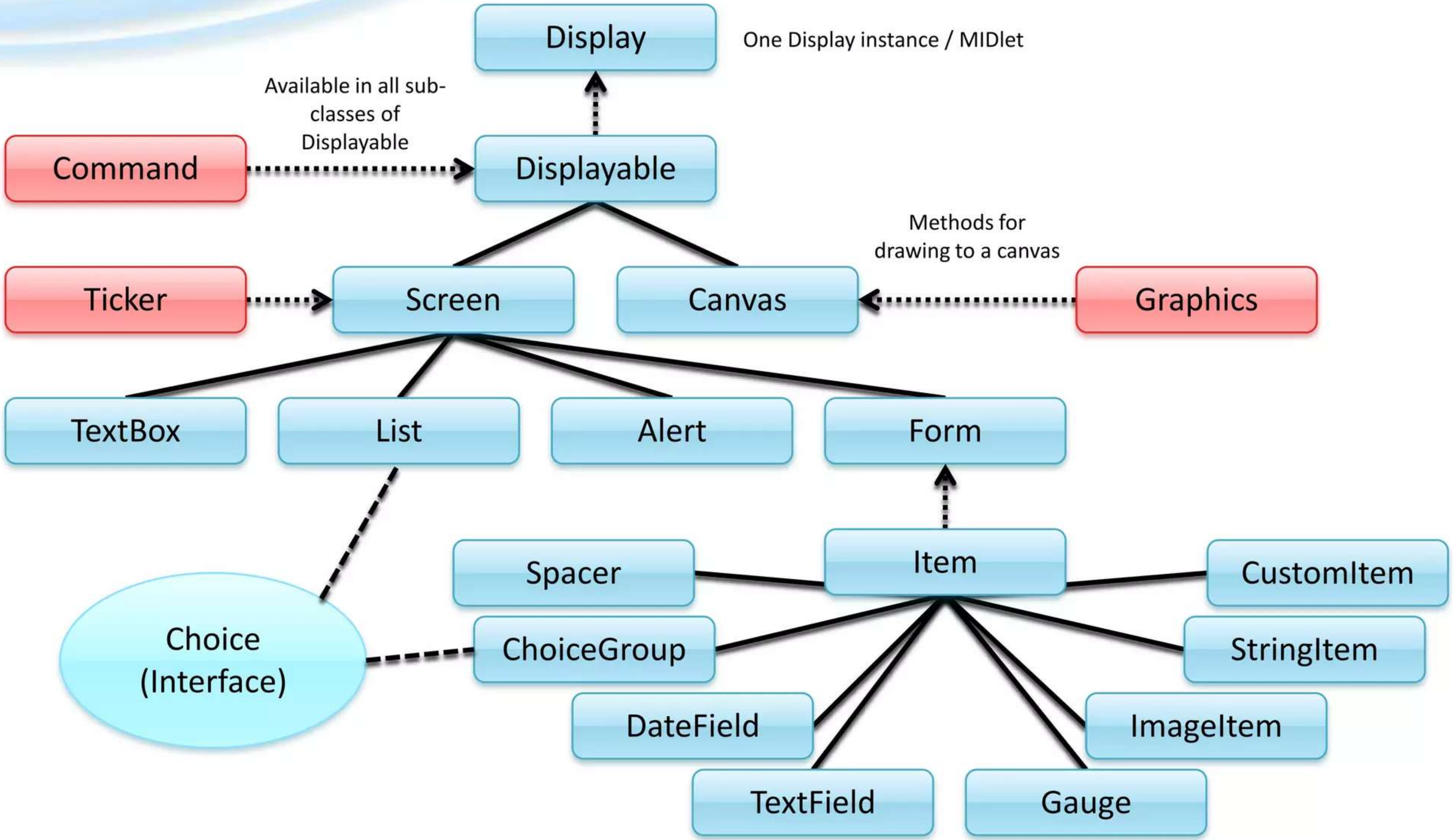
High-Level GUI

GUI Elements

Hierarchy of Displayables



Hierarchy of Displayables



How to create your own layouts

Forms and Items

Form

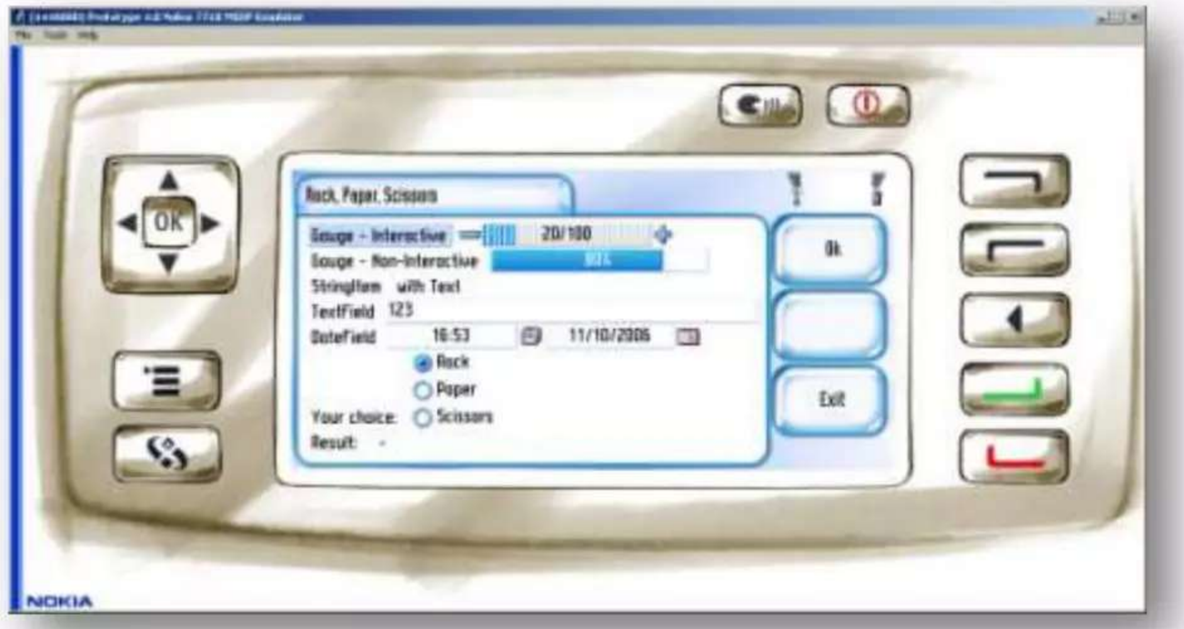
- “Container” for items
- Displays multiple items below each other on the screen
- Automatic scrolling

Most important methods	**Description**
int **append**(Item item)	Add an item at the end
void **insert**(int itemIndex, Item item)	Insert an item before the specified item position
void **set**(int itemIndex, Item item)	(Re)place an item at a specified position
void **delete**(int itemIndex)	Delete an item
void **setItemStateListener** (ItemStateListener itmListener)	Add a listener

Form and Items

WTK emulator 2.5 (DefaultColorPhone)

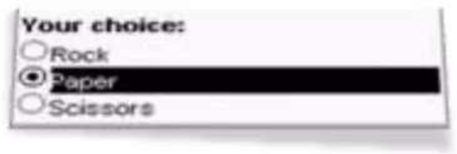
Nokia 7710 Emulator



Individual items, automatically arranged below each other. The appearance depends on the phone.

Example: ChoiceGroup

- Selection from multiple elements
- Either exclusive or multi-selection



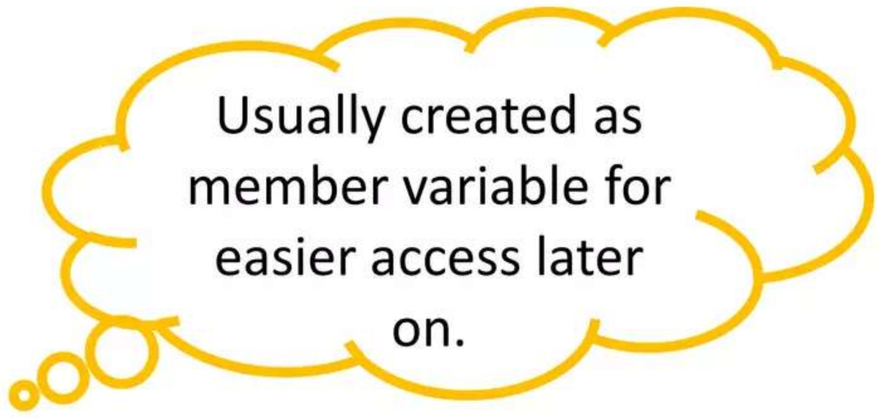
Type: EXCLUSIVE

Type: MULTIPLE

Type: POPUP

Most important methods	Description
ChoiceGroup(String label, int choiceType)	Create an empty choice group item
ChoiceGroup(String label, int choiceType, String[] stringElements, Image imageElements)	Create a choice group item and fill it with data
int **append**(String text, Image image)	Add an option to the end
int **getSelectedIndex**()	Get the index of the currently selected item
int **getSelectedFlags**(boolean[] selectedArray_return)	Get the complete selection status in an array (used for multi-selection)
void **setSelectedIndex**(int elementNum, boolean selected)	Select the specified item (additionally or exclusive)

Process



1. **Create the Form**

   - `Form frmMain = new Form ("Title");`

2. **Create the ChoiceGroup**

```
- ChoiceGroup itmCG = new ChoiceGroup ("Your choice:",
        ChoiceGroup.EXCLUSIVE);
  itmCG.append ("Rock", null);
  itmCG.append ("Paper", null);
```

3. **Add the ChoiceGroup to the Form**

   - `frmMain.`**`append`**` (itmCG);`

4. If necessary: to instantly get information about item changes, register the class as **ItemStateListener**

   - `frmMain.setItemStateListener (this);`

itemStateChanged()

- Will be called by the framework when the item selection changes:
  -
```
public void itemStateChanged (Item item)
{
    if (item == itmCG)
    {
        if (itmCG.getSelectedIndex() == 1) { … }
    }
}
```

- MIDP doesn’t require that this has to be called for every single change.

Item: StringItem

- For (modifiable) **display of text**
- Contains **label** (usually highlighted) and **message**
- **Not editable** by the user
- **Create it explicitly:**
  - ```
    StringItem itmSI = new StringItem("StringItem",
                                      "with Text");
    frmMain.append(itmSI);
    ```
- **Implicit creation:**


  - ```
    // Not possible to set label
    frmMain.append("with Text");
    ```
  - … but more work when you want to change the text later on

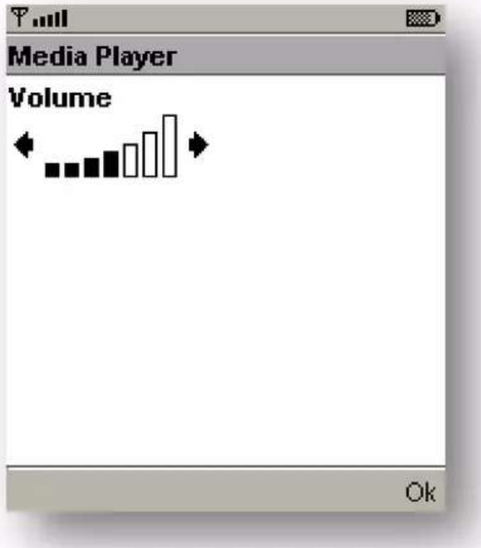
Item: TextField

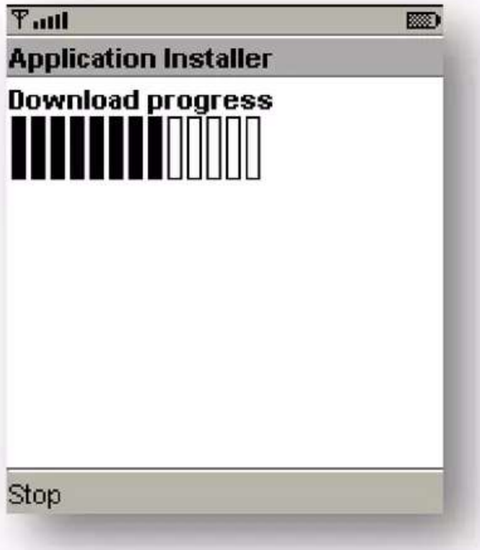
- Single- or multiline text input
- Optional filters to restrict the input:

Filter	Description
ANY	Everything is allowed
EMAILADDR	Only allow valid characters for email addresses
NUMERIC	Whole numbers, includes negative numbers
DECIMAL	Floating point numbers
PHONENUMBER	Chars allowed in telephone numbers (includes +, ...)
URL	Chars allowed in URLs
Important modifiers	**Can be set to additionally influence the text input**
PASSWORD	Display all chars as *
NON_PREDICTIVE	Do not use T9 for text entry (e.g. when it would be useless)
SENSITIVE	Text entered by the user may not be saved to a dictionary
UNEDITABLE	Content may not be edited at this time

Item: Gauge

- **Interactive:**
  - Allow the user to modify the value (e.g. volume)
- **Non-interactive:**
  - Progress bar (e.g. download)
  - Infinitely running activity bar (unknown duration, e.g. when establishing a connection to a server)





Item: DateField

- To modify a Java `Date` object (date + time)
- Implementation of the date/time-editor highly dependent on device manufacturer
  - e.g. date entry: graphical calendar view in Sun WTK or just plain text entry on many devices

Item: ImageItem

- Display a .png image
- Can also be used as button or hyperlink
- Various methods for positioning
- **Code:**

```
Image im = Image.createImage("/img.png");
frmMain.append(new ImageItem("ImageItem", im,
      ImageItem.LAYOUT_DEFAULT, null));
```

- Graphics can also be used for `ChoiceGroup`s, `Alert`s, and `List`s

Other Items

- **Spacer**
  - Separate two items by a specified distance
  - Works vertically and horizontally
- **CustomItem**
  - Base class to develop your own UI items
  - These have to be implemented in a low-level way (you draw the item content directly using lines, …)
  - **Disadvantage:** No automated adaption to the colours / layout of the system (= used by standard items) in MIDP 2.0

Commands for Items

- Set `Commands` specifically for items
- Analogous to all objects derived from `Displayable`:
  - `item.addCommand(myCmd);`
- Handling through **ItemCommandListener** Interface

Predefined standard components

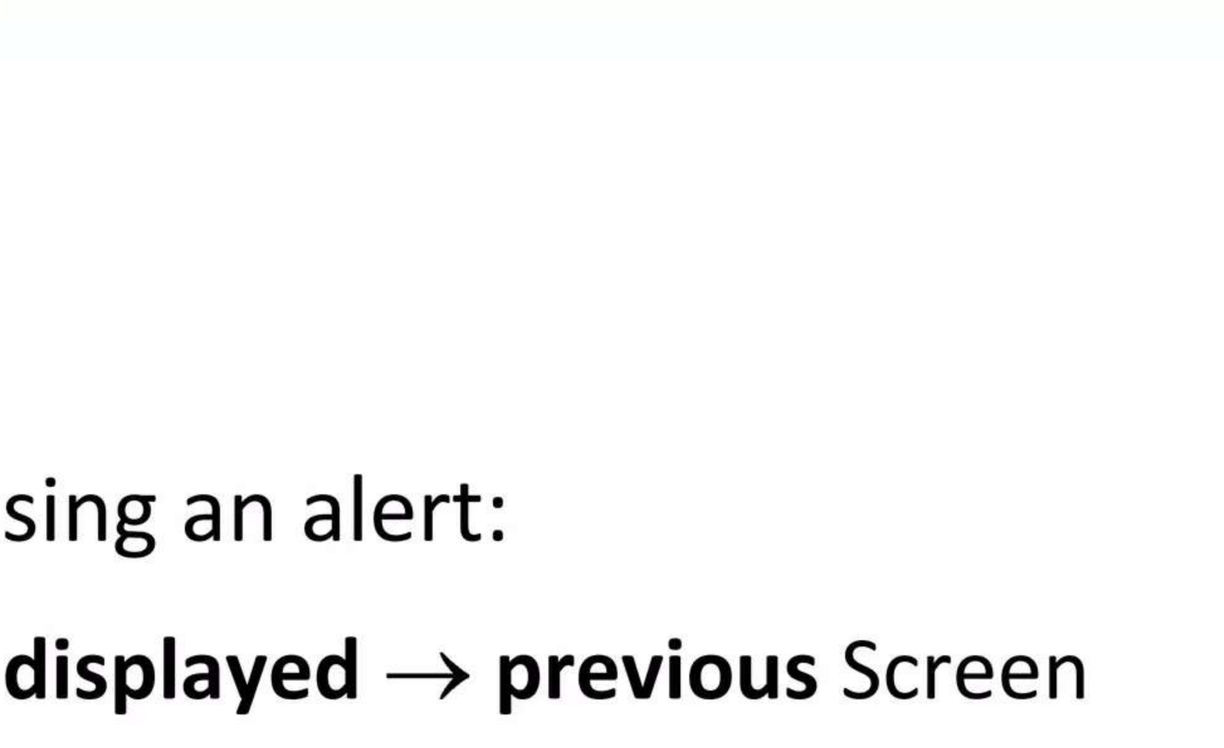
Screens

Alert

- Simple dialog box
- Managed completely by the device, e.g. no own Commands are possible
- 4 **visual attributes** can be specified:
  - Title
  - Image (optional)
  - Progress bar (optional)
  - Text (optional)
- Two **Alert types:**
  - **Modal:** Has to be dismissed by the user
    `al.setTimeout(Alert.FOREVER);`
  - **Timed:** Displayed for *x* milliseconds
    `al.setTimeout(1000);`

Alert – Display

- Display-sequence when using an alert:
  - Screen → **Alert is being displayed** → **previous** Screen
    `display.setCurrent(al);`
  - Screen → **Alert is being displayed** → **new** Screen
    `display.setCurrent(al, nextScreen);`



Alert – with Sound!

- Predefined sounds:
  - **Display an alert with a sound:**
    ```
    Alert al = new Alert("Alert", "Message", null,
    AlertType.WARNING);
    ```
  - **Or play the sounds without displaying the Alert dialog:**
    ```
    AlertType.INFO.playSound(display);
    ```

Alert Type	Description
ALARM	Information about an event that has been agreed upon (no surprise)
CONFIRMATION	After a task has been finished
ERROR	A serious error has occurred
INFO	General, non-critical information
WARNING	Potential problem or a harmless warning

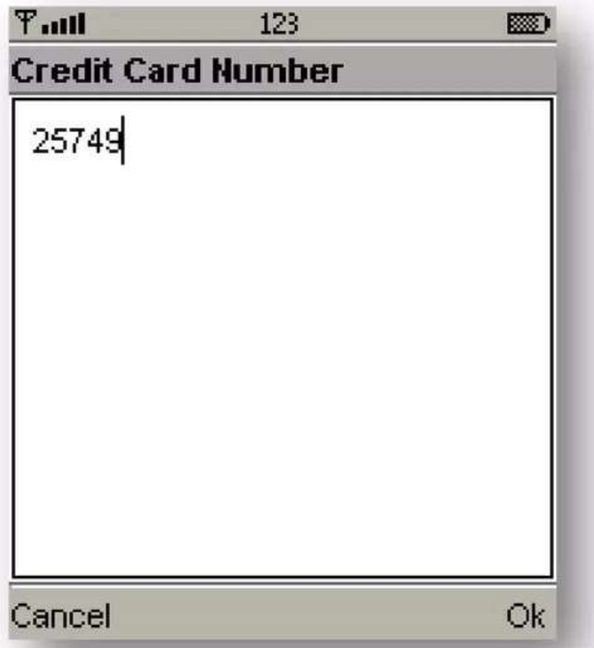
List

- Similar to a (full-screen) **ChoiceGroup** item: select items from a list that you define
- Available in addition to `EXCLUSIVE` and `MULTIPLE`:
  - **IMPLICIT-Lists:** selection triggers an action that can be defined by you.

```
- list = new List("Animals", List.IMPLICIT);
  list.append("Dogs", null);
  list.append("Cats", null);
  list.append("Snakes", null);
  list.setSelectCommand (cmdSelect);
  list.addCommand (cmdInfo);
  list.setCommandListener (this);
```

TextBox

- Full screen text input
- Mostly behaves like a TextField-Item




```
txtBox = new TextBox("Credit Card Number", null, 16,
TextField.NUMERIC|TextField.SENSITIVE|TextField.NON_PREDICTIVE);
```

Title

Default text

Max. length

Restrictions

Modifiers

Ticker

- Text as ticker for instances of Displayable-objects
- Can not be influenced, usage usually not recommended
  - `Ticker t = new Ticker ("Vienna 30 , Barcelona 37 , Hawaii 42 ");`

```
txtBox.setTicker (t);
[…]
// Modify the text
t.setString(„Munich 17 , …");
[…]
// Remove the ticker
txtBox.setTicker (null);
```

Now & the Future

New UI Toolkits

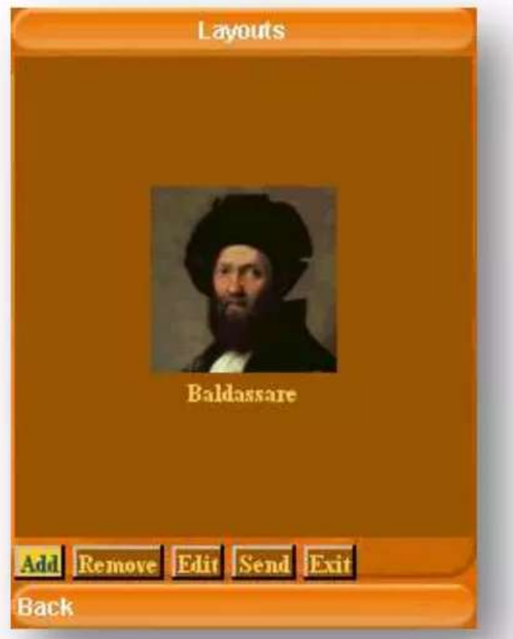
eSWT

- **Embedded Standard Widget Toolkit**
  - Cross platform toolkit
  - Part of Eclipse eRCP (embedded Rich Client Platform)
  - Shares most APIs with desktop SWT
- **Features (excerpt):**
  - Rich UI Component set
  - Flexible and scalable layout system via layout managers
  - Rich user interface events
  - Access to native UI functionality on par with smartphone UI frameworks

eSWT

- **Integrates operating system**
  - Most “real” work done by optimized, platform-specific code (no drawing directly through Java)
- **Traditional Java GUI library characteristics**
  - UI is constructed by widgets in containers
  - Containers use layout managers to scale the UI
- **Available since:**
  - S60 3rd Ed., FP2

LWUIT



- **LightWeight User Interface Toolkit**
  - Inspired by Swing
  - But designed for constrained devices
  - Can be added to any Java ME application (embedded .jar)
  - Drawing done in Java source code, without native peer rendering
- **Features (excerpt):**
  - Layouts
  - Themes, fonts
  - Animations & Transitions
  - 3D / SVG integration (optional)
  - Internationalization

JavaFX

- **JavaFX** (http://javafx.com – integrated in NetBeans 6.5+)
  - New **UI libraries** (graphics, media, web services)
  - **Consistent experience** across mobile, desktop, browser, TV, etc
  - **Plus:** use any Java library in JavaFX
  - Integrated with Java Runtime
- **JavaFX Script**
  - Simple declarative language, easier to learn
  - e.g., for artists to change sprite animation, without needing software developer
  - Advantage to JavaScript / ActionScript: integration with Java – reuse any Java library

JavaFX Mobile

- **Runs on Java ME (plus Android)**
  - Mobile content with same tools as Java FX
- **Availability?**
  - JavaFX Mobile Runtime needs to be pre-installed on the phone. None released yet.
  - Currently endorsed by: SonyEricsson, LG

eSWT vs. LWUIT vs. JavaFX Mobile?

- **Architecture**
  - **LWUIT & eSWT:** scene graph component model framework (like Swing / SWT)
  - **JavaFX:** more a vector graphics platform
- **Features**
  - **LWUIT:** Simulates UI design in pure Java-code, doesn't fit to phone UI. OK for games, not for business.
  - **eSWT:** Uses native code for drawing. More flexible (browser component, etc.). Good for business and speed, not so useful for games.
- **Support**
  - **LWUIT:** can be used today
  - **eSWT:** only on latest Nokia phones
  - **JavaFX:** Not supported yet by phones. Fully vector graphics based – will only work on newest phones
- **Conclusion**
  - Depending on use case: Use eSWT / LWUIT now. JavaFX Mobile will be the future.
  - http://weblogs.java.net/blog/terrencebarr/archive/2008/08/comparing_lwuit.html

That’s it!

Thanks for your attention