



# **Mobile Information Device Programming (16)**

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

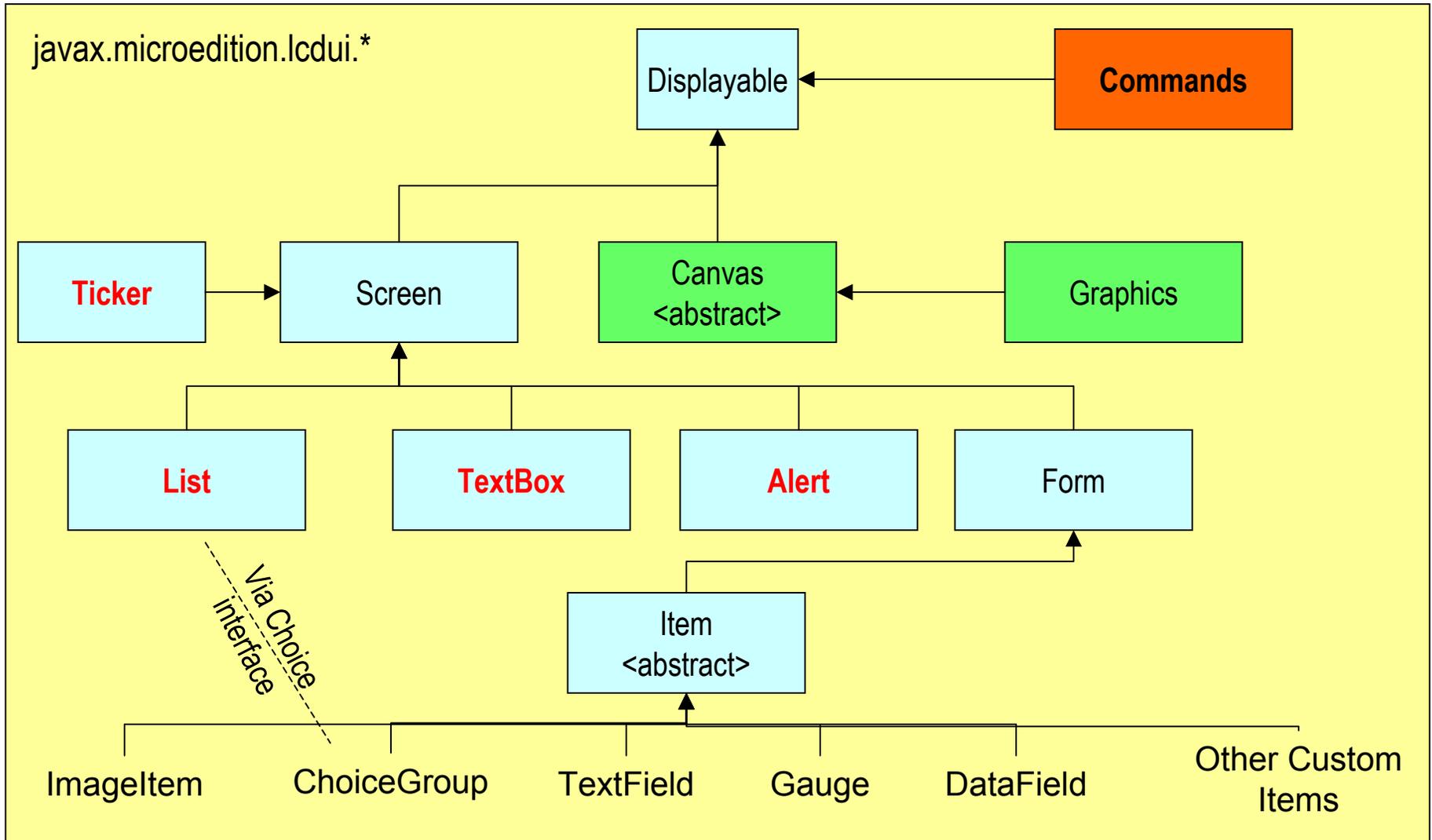


# Animation

- MIDlets for **Animation** applications
- **Display Manager** API (Manage Displayable objects)
- **Animating** a series of Images
- Examples



# Displayable Hierarchy





# Facts

1. There is little to no complexity in moving from a **Form** to an **Alert** or **List**
2. There may be applications that you want to move between a number of *Displayables*
3. Going forward is normally straightforward but rewinding could be challenging – Hence
4. **DisplayManager** deals with MIDlets with a large number of *Displayables*



# The Display Manager Class

- It leaves a **trail** as you move along the *Displayables* in your application.
- This **trail** makes moving back through *Displayables* much simpler.
- The *DisplayManager* class is not part of MIDP it is designed (by J. W. Muchow 2002)



# Display Manager API

- The *DisplayManager* is implemented as a stack under `java.util.Stack`, The methods are:

<i>Method</i>	<i>Description</i>
<b><i>Constructor</i></b>	
<i>DisplayManager(Display display, Displayable mainDisplayable)</i>	Creates a new DisplayManager
<b><i>Methods</i></b>	
<i>void pushDisplayable(Displayable new Displayable)</i>	Push a Displayable object on to the stack
<i>void popDisplayable( )</i>	Pop a Displayable object off the stack
<i>void home( )</i>	Pop all Displayable objects and return to the "main" displayable object

REF: J. W. Muchow (2002), *J2ME Technology & MIDP*, The Sun Microsystems Press

# Example16-1: Coding an Animation Application



**Note: this example is extracted from J.W. Muchow (2002) with minor modifications**

- We need to create 5 MIDlets for this example:
  - A DisplayManager
  - A Main MIDlet called AnimTimer.java
  - A TimerCanvas to create the graphics and placing the images on display (we will use clipping techniques [why?])
  - OptionsList to choose the actions to be taken e.g. Start, Stop and Set Timer
  - SleepForm a form containing a gauge to control the speed of animation, and other commands to save the settings, going to the main menu and back to the previous display.
- Creating the images



# Results





## Ex 16-1: General

- Use a stack to push and pop Displayable objects
- The constructor needs to refer to a Displayable object e.g. Canvas, Form, ...
- Define an *Alert* component to pop up error messages if they occur
- Create a *Form* and define it as the main Displayable



## Ex 16-1: General cont.

- Create the main MIDlet
- Show a Canvas with animated time
- Include start and stop configurations for the timer
- Adjust the sleep interval of the thread (setting the time the animation thread sleeps between repainting the display)



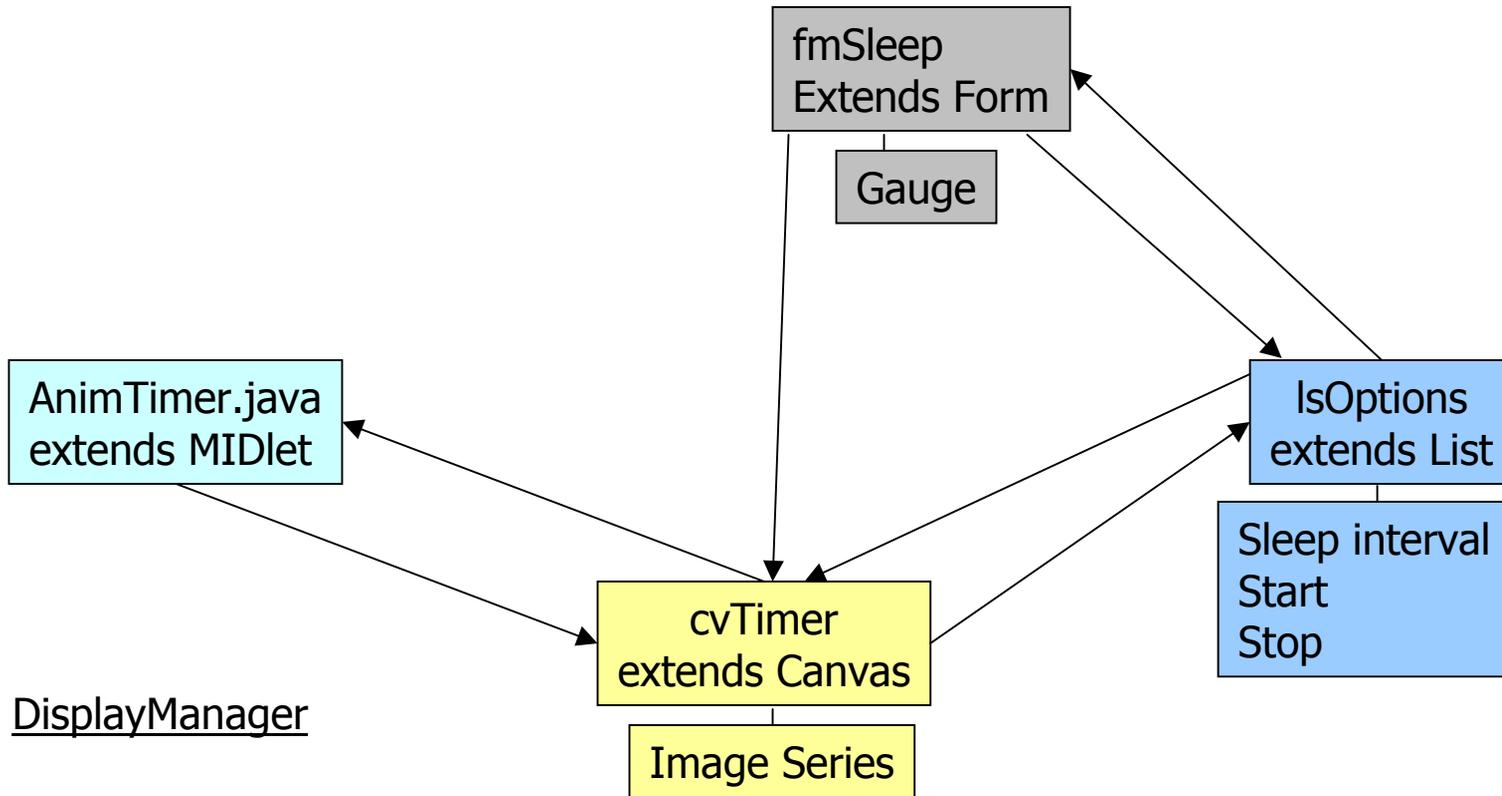
# Ex 16-1: General cont.

- On the Sleep Form you will have 3 options:
  1. Back: returns to the Gauge
  2. Save: saves the current speed setting (the higher the gauge the slower the animation)
  3. Home: to return to the animation.





# Animation MIDlet Design



DisplayManager

pushDisplayable( )  
popDisplayable( )  
home( )



# Application Design

## AnimTimer class

- *AnimTimer* is the main MIDlet class, its task is to initialise other components and manage the interface with application manager.

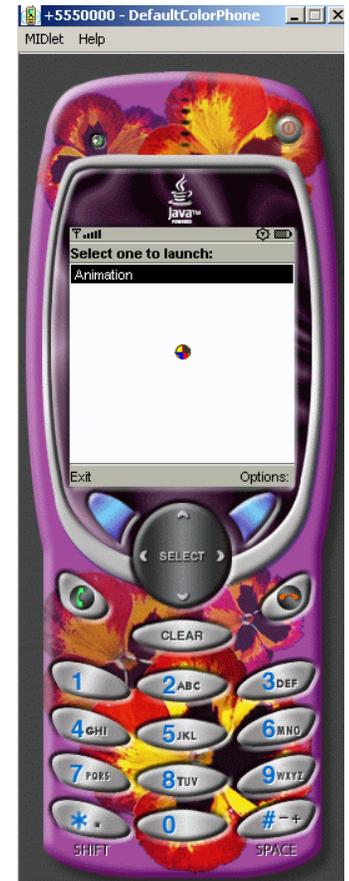
# Application Design

## cvTimer Class



*cvTimer* corresponds to the first instance of running *the application*.

- You can either Exit the application or choose from a list of options
- Here is the first encounter with *pushDisplayable()* Calling the *DisplayManager* for the first time
- This call is to push a reference to the current *Displayable(cvTimer)* onto a stack





# Application Design

## IsOptions & fmForm Class

- From *IsOptions* you can return to *cvTimer* or Display Form (fmSleep)
- Note that when we move to fmSleep we push The current *Displayable* onto a stack
- From fmSleep you can either return to *IsOptions* (*popDisplayable()* from the stack) or *cvTimer* using *home()* method