# GUI Programming with Java

**Lecture 2**
**Starting with SWING**

- We will look at…

- Creating Frames in SWING
- Labels
- Buttons
- Basic Text

# Basic Components

# Creating a Window (Frame)

- Before we can draw a SWING component on screen we need a window to place it into.
- A top level window (that is, a window that is not contained inside of another window) is called a frame in JAVA.
- The AWT library has a peer based class called Frame.
- The SWING version of this class is called JFrame.
- JFrame extends the frame class and is one of the few SWING components that is not drawn on the canvas.
- Frames are examples of containers. This means that a frame can contain other user interface components such as buttons and text fields.

## Creating a Frame

- Look at the sample program contained in sample 1 folder.

Conclusions

- JAVA swing packages are contained in the javax.swing package. The package name javax indicates a JAVA extension package and not a core package.
- In the program we define a class, Firstframe, that behaves exactly like the class JFrame in the javax.swing package with two exceptions.
- The constructor of the of FirstFrame sets the title bar to the sting “FirstFrame” and the size of the frame to 300 X 200 pixels.
- The frame has no other methods.
- The main method just shows the frame and exits. Note that it does not terminate the program, just the main thread.

## Creating a Frame (2)

- To actually show a frame, you perform the following steps.
  - Create a frame object by a call to new.
  - Optionally position it on screen by using the setLocation method.
  - Call the show method to make the frame visible and to bring it to the front if it behind another window.

- The key line in the main method is
  - JFrame frame = new FirstFrame.

  This line makes a new frame by giving all the necessary information to the underlying windowing system to display a window.

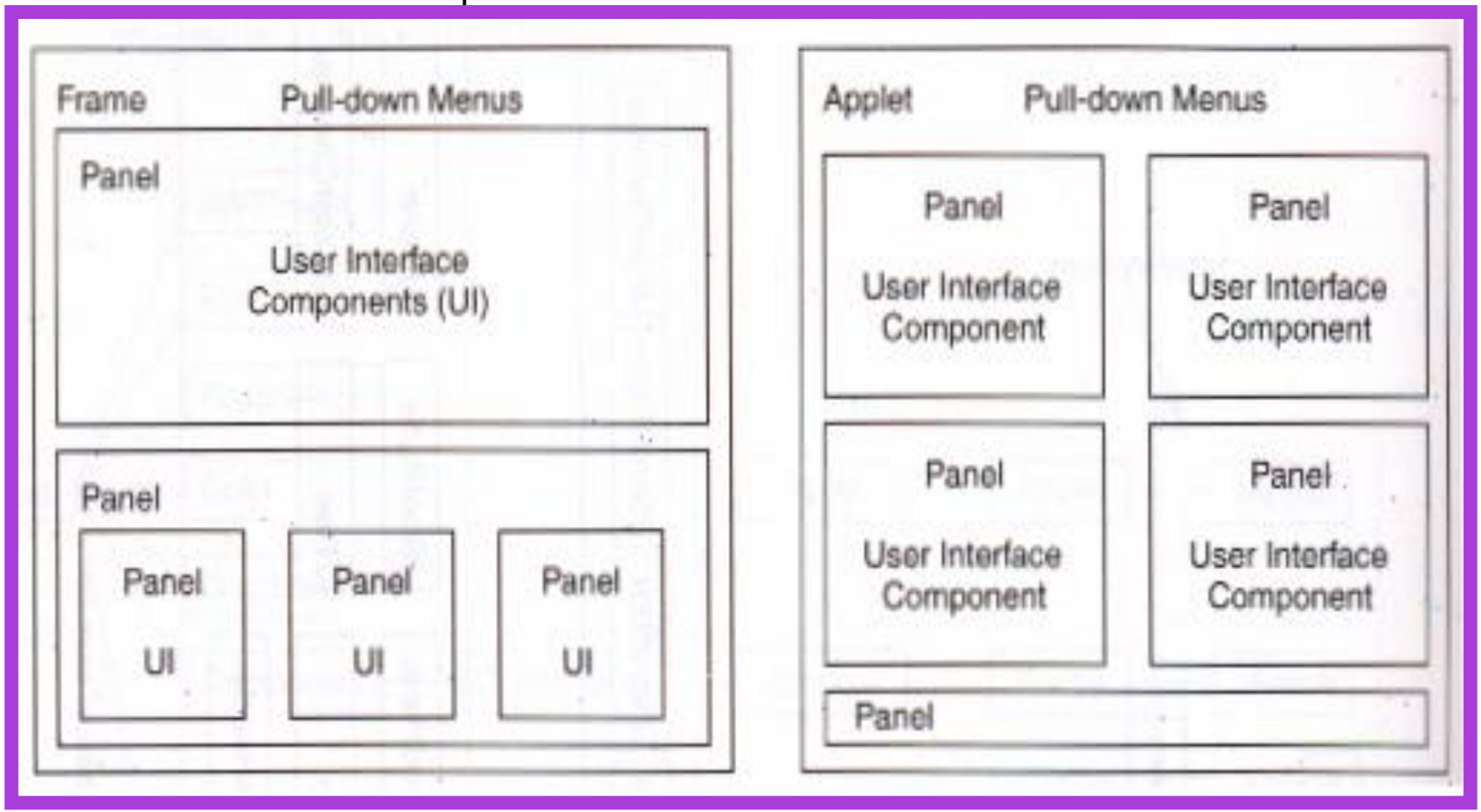
- Remember that new does not display the frame. We must use either the show or setVisible methods to make it appear on screen.

- But we're not finished. The default size for a new frame is 0x0 pixels. We must use the setSize method to give the frame (window ) a size.

- We haven't taken into account a method for terminating the window. We'll do that later when we examine event handling!

## Frame Contents

- A frame or an applet can contain menus, panels, and user interface components.
- Panels are used to group user interface components.
- Panels can contain other panels.



## Drawing on a Frame?

- It would be possible to draw on or place buttons, labels etc, directly onto a frame. However this is not considered good programming practice. In JAVA frames are designed to be containers for components.
- Normally we draw/place objects on another component which we call a panel. This is added to the frame.
- A frame is made up of many panes (menu bar, lass, content, root etc). As programmers we are only concerned with the content pane.
- When developing, we add components into the content pane, using code such as the following

```
Container contentPane = frame.getContentPane();
Component c = ……;
contentPane.add(c);
```

## Our first component - JLabel

- Labels provide text instructions or information on a GUI. Labels are defined with the class JLabel – a subclass of JComponent.

- A JLabel is a single line label similar to java.awt.Label. Additional functionality that a JLabel has is the ability to:
  - Add an Icon
  - Set the vertical and horizontal position of text relative to the Icon
  - Set the relative position of contents within component

- Our first simple JLabel program is available for review in the sample2 folder - LabelPanel.java.

## Conclusions regarding the JLabel sample

- We use both the swing and awt classes.
- Our class LabelFrame is based on JFrame (we are going to create a window)
- We call the super class constructor to give our frame a name in the title bar using the code

  **super("My Panel");**
- We can get the content pane by using the code Container

  **Container contentPane = getContentPane();**
- We create a panel for placing GUI components on by using the code

  **JPanel panel = new JPanel();**

## Conclusions regarding the JLabel sample

- We create the first label using the code

```
JLabel plainLabel = new JLabel("Plain Small Label");
panel.add(plainLabel);
```

- Notice that we add the label to the panel and not to the content pane. At a later stage we will add the panel with the GUI components to the content pane.

- We create a new font for our next label using the font class (this is part of the awt package).

```
Font fancyFont = new Font("Serif", Font.BOLD | Font.ITALIC, 32);
```

- We then set the font for the label using the code

```
fancyLabel.setFont(fancyFont);
```

## Conclusions regarding the JLabel sample (2)

- We are going to add an icon to the label. This involves two steps
  - Create an instance of the icon class
  - Assign an image to the instance of the icon class
  - Add the icon instance to the label object using the setIcon method.

```
// Create an Icon
        Icon tigerIcon = new ImageIcon("bug1.gif");

// Place the Icon in the label
        fancyLabel.setIcon(tigerIcon);

// Align the text to the right of the Icon
        fancyLabel.setHorizontalAlignment(JLabel.RIGHT);
```

# Conclusions regarding the JLabel sample (3)

- We add the second label to the panel using the code

  **panel.add(fancyLabel);**

  Now that we have added all of our GUI components to the panel we will add the panel to the frame.

  **contentPane.add(panel);**

- We then set the size of the frame and display it on screen.

  **setSize(300,200);**
  **setVisible( true );**

## Buttons - JButton

- A button is a component that a user clicks to trigger a specific action. A JAVA program can use several types of buttons, including command buttons, check boxes, toggle buttons and radio buttons.
- The SWING version of a button is called JButton.
- As with labels, we can specify a size, text and an icon.
- To view a simple button application view the sample contained in sample 3 - ButtonFrame.java

## Expanding JButton

- As with JLabel, we can add an icon to a button.
- To view a simple button application with an icon view the sample contained in sample 4 - ButtonIcon.java
- Notice how we add the icon. We do not use the setIcon function. Instead we use the constructor by passing it values for a string and an icon

  **JButton myButton = new JButton("The Bug", bugIcon);**
- JButton and JLabel are two very simple yet powerful GUI components. Just take a look at all of the methods and properties available to us through their use.
- For a full list of methods/properties make sure to check the JDK documentation stored in the JAVA documentation folder.

## Text Fields – JTextField & JTextArea

- A text field is an input area where the user can type in characters. Text fields enable the user to enter variable data, such as names or descriptions (JTextField).

- A text area allows the user to enter multiple lines of text (JTextArea).

- Both of these are sub classes JTextComponent. is a generalized text class that contains all the features you would expect from a simple editor. Some of its methods include:

| | | | |
|---|---|---|---|
| copy() | cut() | paste() | getSelectedText() |
| setSelectionStart() | setSelectionEnd() | selectAll() | replaceSelection() |
| getText() | setText() | setEditable() | setCaretPosition() |

## Text Fields – JTextField & JTextArea (2)

- JTextComponent objects in Swing can be placed in a panel in a fashion nearly identical to AWT text widgets.
- There are three basic subclasses of JTextComponent: JTextField, JTextArea, and JEditorPane. JPasswordField and JTextPane are sub-subclasses that are also of interest.
- If you want your users to be able to see content that exceeds the screen display area, you must place the component inside of a JScrollPane to support scrolling to the extra content.
- Other than having to add a JTextArea to a JScrollPane for scrolling, JTextField and JTextArea behave very similarly to their AWT counterparts: java.awt.TextField and java.awt.TextArea:

## Text Fields – JTextField & JTextArea (3)

- To view a simple text application view the sample contained in sample 5 - TextDemo.java

- The main methods that we use are the setText methods.

```
tf.setText("TextField");
ta.setText("JTextArea\n Allows Multiple Lines");
```

- Note that we added the text area to a scroll pane using the code

```
panel.add(new JScrollPane(ta));
```

- This is an abstract object. We do not give it an identifier
  **e.g. myScroll = new scrollPane()**
  as we will not be making further reference to the scroll pane.

## Summary

- **The SWING API was introduced to improve the process of creating graphical user interfaces.**
- **It makes use of lightweight components, many of which inherit methods/behaviours from AWT classes.**
- SWING offers developers a number of advantages including
  - Swing has a much richer and convenient set of user interface elements.
  - Swing depends much less on the underlying platform, therefore is much less prone to platform specific bugs.
  - SWING will give a consistent user experience across platforms.

## Summary (2)

- Swing is built on top of AWT, it DOES NOT REPLACE AWT.
- Some of the more common GUI components we have examined include JLabel, JButton, JTextField and JTextArea.
- Before we can draw a SWING component on screen we need a window to place it into.
- A top level window (that is, a window that is not contained inside of another window) is called a frame in JAVA.
- The SWING version of this class is called JFrame.
- Frames are examples of containers. This means that a frame can contain other user interface components such as buttons and text fields.

## Summary (3)

- Normally we draw/place objects on another component which we call a panel. This is added to the frame.
- Labels provide text instructions or information on a GUI. Labels are defined with the class JLabel – a subclass of JComponent.
- A button is a component that a user clicks to trigger a specific action. A JAVA program can use several types of buttons, including command buttons, check boxes, toggle buttons and radio buttons.
- The SWING version of a button is called JButton.

## Summary (4)

- A text field is an input area where the user can type in characters. Text fields enable the user to enter variable data, such as names or descriptions (JTextField).
- A text area allows the user to enter multiple lines of text (JTextArea).
- Both of these are sub classes JTextComponent. is a generalized text class that contains all the features you would expect from a simple editor.
- Extensive documentation is available on the SWING API and is available in the JAVA documentation folder for this course.

Labwork 2:

**Task 2-1**

Write a class that creates a JFrame window with the following properties

- Title: First Java GUI Window
- Size: 440 X 380
- Location: 100 x 300 (use setLocation(x,y) method)
- Allow Resize: No (use the setResizable(true/false) method)

**Task 2-2**

Write a class that contains a window. The window should contain 2 labels with the following properties

Text: My first label

Font: Arial, italic, size 14

(use setFont method and create a font using new Font)

Text: My second label

Font: Times, Bold, size 12

Labwork 2

**Task 2-3**

Write a class that creates a window containing 3 buttons

- Button 1 | Text: My first Button | Enabled: true
- Button 2 | Text: My second Button | Enabled: true | Font: Arial, bold, 12 |
- Button 3 | Text: My third Button | Enabled: false | Font: Serif, italic, 14 |

Labwork 2:

**Task 2-4**

Write a class that creates a window containing 2 text areas

Text area 1

- Set the text area to contain the following text “This is my first text area”
- Allow the text area to be edited by the user (setEditable method, ‘true’)

Text area 2

- Set the text area to contain the following text “This is my second text area”
- Prevent the text area from being edited by the user (setEditable method, ‘false’)

# END OF SESSION