cis3120
design and implementation of software applications I
spring 2015
lecture # II.1: Java GUI API

topics:

- Java GUI API:
  - Containers
  - Components
- *Reference: A Programmers Guide to Java Certification by Mughal and Rasmussen*

GUIs

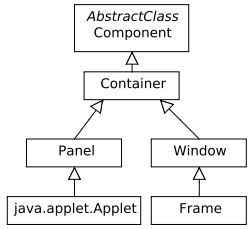
- GUI = Graphical User Interface
- The *Java Foundation Classes (JFC)* provide two frameworks for building GUI based applications:
  - *Abstract Windowing Toolkit* (AWT) relies on the underlying windowing system on a specific platform to present its GUI components. package:`java.awt`.
  - *Swing* implements a new set of *lightweight* (does not rely on the underlying windowing system) GUI components that are written in Java and have a pluggable look and feel. Built on AWT event handling model, provides lightweight counterparts to AWT GUI components. package:`javax.swing`.

GUI Building Blocks

- In Java, GUI elements can be classified into following parts:
  - `Containers`: components that can accomodate other components (e.g.top-level windows, intermediate-level containers for component organization)
  - GUI Control `Components`: primary elements of the user interface. (e.g. buttons, labels, drop-down menus, radio buttons etc.)
  - `LayoutManagers`: policies that define how components are displayed within containers
  - `Events`: refers to user actions (e.g. button click, radio button checked, mouse click, key stroke etc.)

Containers (AWT)

- *containers* are special components that can hold other components



- `Window`: top-level window without title, menus or borders (rarely used)
- `Frame`: window with title and border, usually the starting point of an application
- `Panel`: intermediate-level container used for organizing components

java.awt.Component class

- All non-menu related GUI elements are derived from `Component` class
- It is an abstract class specifies a large number of methods related to position and display of components. Some methods commonly overriden/used by its subclasses:
  - `paint(Graphics g)`
  - `setVisible(boolean b)`
  - `setSize(int width, int height)`
  - `setForeground(Color c)`
  - `setBackground(Color c)`
  - `setEnabled(boolean b)`

Frame

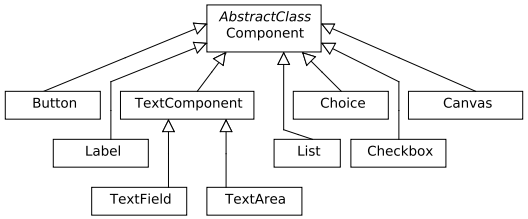
```
import java.awt.*;

public class FrameTest {
  public static void main( String[] args ) {
    Frame frame = new Frame( "myframe!" );
    MyCanvas c = new MyCanvas();
    frame.add(c);
    frame.setSize( 100,100 );
    frame.setBackground( Color.red );
    frame.setVisible(true);
  }
}

public class MyCanvas extends Canvas {
  public void paint( Graphics g ) {
    setBackground( Color.blue );
    g.drawString( "hiho",10,10 );
  }
}
```


AWT GUI Control Components

- *components* are primary elements of any GUI :



- to make a GUI control component:
  - construct: `Button guiButton = new Button("Ok");`
  - add to container: `guiFrame.add(guiButton);`
  - register listeners to capture user action events

AWT GUI Control Components (2)

- `Button`: A button with a textual label designed to invoke an action when pushed
  - C'tors:
    `Button()`
    `Button(String label)`
  - Some useful methods:
    `String getLabel()`
    `void setLabel(String label)`
- `Canvas`: A generic component for drawing and designing new GUI components. No default graphical representation. Usually subclassed to customize GUI components. Its `paint()` method is overriden to render graphics.
  - C'tor:
    `Canvas()`

AWT GUI Control Components (3)

- `Checkbox`: A checkbox with a textual label that can be toggled on and off.
  - C'tors:
    ```
    Checkbox()
    Checkbox(String label)
    Checkbox(String label, boolean state)
    Checkbox(String label, boolean state, CheckboxGroup group)
    ```
  - Some useful methods:
    ```
    boolean getState()
    void setState(boolean state)
    CheckboxGroup getCheckboxGroup()
    void setCheckboxGroup(CheckboxGroup g)
    ```


AWT GUI Control Components (4)

- `CheckboxGroup`: CheckboxGroup is just a class to implement mutual exclusion among a set of checkboxes (e.g. checkboxes can be grouped to represent radio buttons). Doesn't have a graphical representation. Not a subclass of `Component`.
  - C'tors:
    ```
    CheckboxGroup()
    ```
  - Some useful methods:
    ```
    Chekcbox getSelectedCheckbox()
    void setSelectedCheckbox(Checkbox box)
    ```


AWT GUI Control Components (5)

- `Choice`: A component that provides a pop-up menu of choices. Only the current choice is visible. Constructing a menu of choices involves, creating a `Choice` object using the single default constructor and adding items using `add()` method.
  - C'tors:
    ```
    Choice()
    ```
  - Some useful methods:
    ```
    void add(String item)
    int getItemCount()
    String getItem(int index)
    String getSelectedItem()
    void select(String str)
    ```


AWT GUI Control Components (6)

- `Label`: A label is a component that displays a single line of read-only, non-selectable text.
  - C'tors:
    ```
    Label()
    Label(String text)
    Label(String text, int alignment)
    ```
  - Some useful methods:
    ```
    String getText()
    void setText(String text) int getAlignment()
    void setAlignment(int alignment)
    ```

AWT GUI Control Components (7)

- `List`: A component that defines a scrollable list of text items.
  - C'tors:
    ```
    List()
    List(int rows)
    List(int rows, boolean multipleMode)
    ```
  - Some useful methods:
    ```
    void add(String item)
    void add(String item, int index)
    int getRows()
    String[] getItems()
    String[] getSelectedItems()
    void select(String str)
    boolean isMultipleMode()
    ```


AWT GUI Control Components (8)

- `TextField`: A component that implements a single line of optionally editable text.
  - C'tors:
    ```
    TextField()
    TextField(String text)
    TextField(int columns)
    TextField(String text, int columns)
    ```
- `TextArea`: A component that implements multiple lines of optionally editable text.
  - C'tors:
    ```
    TextArea()
    TextArea(String text)
    TextArea(int rows, int columns)
    TextArea(String text, int rows, int columns)
    TextArea(String text, int rows, int columns, int scrollbars)
    ```


AWT GUI Control Components (9)

- Both *TextField* and *TextArea* have the following methods:
  ```
  int getColumns()
  void setColumns(int columns)
  ```
- They inherit the following methods from their parent class, `TextComponent`:
  ```
  String getText()
  void setText(String t)
  String getSelectedText()
  boolean isEditable()
  void setEditable(boolean b)
  ```


applets (1)

- *application:*
  - executed using the *java* command
  - server and client can be the same machine or different machines
  - client invokes JVM which interprets classes and runs them
- *applet:*
  - must be executed using a browser, or the *appletviewer* command
  - server sends applet to the client, in the form of class files; applet invokes JVM which interprets classes and runs them on the client
  - there are two parts:
    - an HTML file used to invoke the applet
    - Java class file(s) that contain the applet code

applets (2)

- file name = hi.html

```
<html>
<title>
sample applet page
</title>

the applet will be shown below...

<applet code="HiWorldApplet.class" width=400 height=400>
</applet>

</html>
```


applets (3)

- file name = Hi.java

```
import java.awt.*;
import java.applet.Applet;

public class HiWorldApplet extends Applet {
  public void paint( Graphics g ) {
    g.drawString( "hi",10,10 );
  }
}
```

- Running using appletviewer: `appletviewer hi.html`
- or open the .html file in a Java enabled browser.

applets (4)

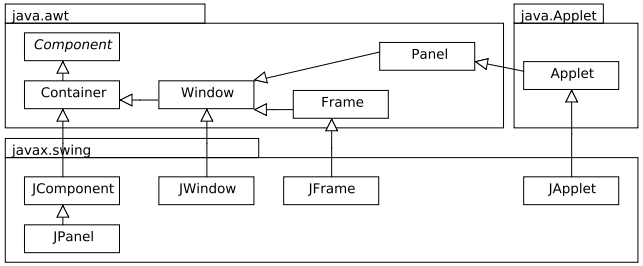
- `java.awt` package
  - includes `java.awt.Component` method:
    - `public void paint()`
- `java.applet.Applet` class
  - `public void init()`
  - `public void start()`
  - `public void stop()`

Swing

- Swing is not a replacement for AWT but rather and extension that provides lightweight components that do not depend on the native windowing system.
- Because it is not constrained by the native windowing system, it has more advanced components that are available in AWT (e.g. JTree, JTable, JFileChooser).
- Swing uses many of the same concepts as AWT
- Uses AWT's event model and layout managers.
- Most common AWT containers and components have Swing counterparts having a 'J' prefix to their names (e.g. JFrame, JButton)

Swing Containers

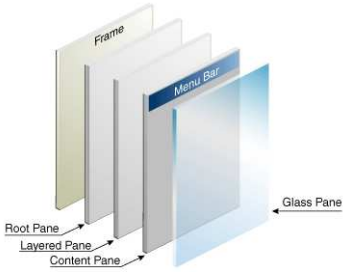
- At the top of the javax.swing package hierarchy is the `JComponent`, the base class for all Swing components.



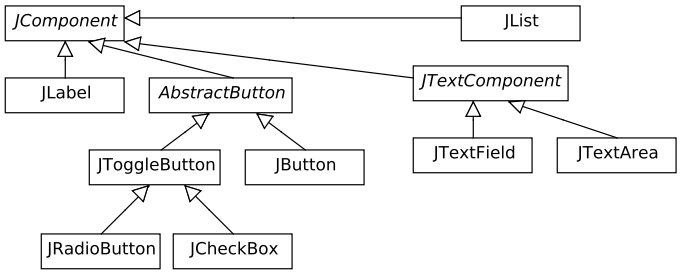
- `JComponent`, `JWindow`, `JFrame` and `JApplet` classes are called *root component* classes, extending from respective AWT classes and are heavy-weight.
- `JPanel`, similar to `Panel` is generally used to organize primary components.

RootPane container model

- *content pane*: all components should be added to this pane.
- *layered pane*
- *glass pane*
- and an optional menu-bar.



Swing GUI Control Components



Swing GUI Control Components

- Every AWT component has a Swing counterpart
  - `JLabel` → `Label`
  - `JButton` → `Button`
  - `JCheckbox` → `Checkbox` with `CheckboxGroup`
  - `JRadioButton` → `Checkbox`
  - `JComboBox` → `Choice`
  - `JPanel` → `Canvas`
  - `JList` → `List`
  - `JTextField` → `TextField`
  - `JTextArea` → `TextArea`
- Swing has more additional components: `JToolBar`, `JProgressBar`, `JSeperator`