Graphical User Interfaces (GUI’s)

Objectives

- Learn about various GUI components.
- Be able to use layout managers to arrange components in a window.
Outline

- Java Packages
- Some simple GUI components
  - Labels
  - Text fields
  - Buttons
  - Check boxes and radio buttons
- Containers
- Layout Managers

Packages

- Related classes are grouped into packages.
- By convention developers use unique names for their packages starting with the organization’s domain, then sub-domains, separated by dots, e.g:
  - edu.umassd.cis.pbergstein
  - com.adobe.crypto
- The standard java library includes packages that start with java or javax, e.g:
  - java.lang
  - java.awt
  - javax.swing
Import Statements

- In order to use a class from a different package you must either:
  - Use the its full name, including the package, e.g.
    java.awt.Window
  - Or, use an import statement at the beginning of your source file:
    import java.awt.*;
- The only exception is the java.lang package which is always automatically imported.

GUI Classes
Visual Guide to Swing Components

http://java.sun.com/docs/books/tutorial/ui/features/components.html

Components

- The component class defines many useful methods inherited by all the other GUI classes.

<table>
<thead>
<tr>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ setSize(width: int, height: int) : void</td>
</tr>
<tr>
<td>+ setLocation(x: int, y: int) : void</td>
</tr>
<tr>
<td>+ setVisible(isVisible: boolean) : void</td>
</tr>
<tr>
<td>+ paint(g: Graphics) : void</td>
</tr>
<tr>
<td>+ repaint( ) : void</td>
</tr>
</tbody>
</table>
Labels

- A JLabel displays a single line of text.

```
JLabel
+ JLabel(s: String)
+ setText(s: String) : void
+ getText() : String
```

Text Components

- A JTextField provides box with a single line of editable text.
- A JTextArea provides a box for multi-line editable text.

```
JTextField
+ JTextField( )
+ JTextField(s: String)
+ JTextField(columns: int)
+ JTextField(s: String, columns: int)
```
```
JTextArea
+ JTextArea( )
+ JTextArea(s: String)
+ JTextArea(rows: int, columns: int)
+ JTextArea(s: String, rows: int, columns: int)
```
**Simple Buttons**

```
<table>
<thead>
<tr>
<th>AbstractButton</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ setText(s: String) : void</td>
</tr>
<tr>
<td>+ getText() : String</td>
</tr>
<tr>
<td>+ isSelected() : boolean</td>
</tr>
<tr>
<td>+ setSelected(b: boolean): void</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>JButton</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ JButton( )</td>
</tr>
<tr>
<td>+ JButton(s: String)</td>
</tr>
</tbody>
</table>
```

**Checkboxes and Radio Buttons**

```
<table>
<thead>
<tr>
<th>AbstractButton</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ JButton(s: String)</td>
</tr>
<tr>
<td>+ JButton(s: String, isSelected: boolean)</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>JToggleButton</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ JButton(s: String)</td>
</tr>
<tr>
<td>+ JButton(s: String, isSelected: boolean)</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>JCheckbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ JButton(s: String)</td>
</tr>
<tr>
<td>+ JButton(s: String, isSelected: boolean)</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>JRadioButton</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ JButton(s: String)</td>
</tr>
<tr>
<td>+ JButton(s: String, isSelected: boolean)</td>
</tr>
</tbody>
</table>
```
**Button Groups**

- A button group manages a set of radio buttons, so that when one member of the group is turned on, all the others are turned off. The ButtonGroup object is not visible.

```java
ButtonGroup
+ ButtonGroup( )
+ add(button: AbstractButton) : void
+ remove(button: AbstractButton) : void
```

**Radio Button Example**

```java
// Let the user pick a color by selecting a radio button

JRadioButton red = new JRadioButton("Red", true);
JRadioButton yellow = new JRadioButton("Yellow", false);
JRadioButton purple = new JRadioButton("Purple", false);

ButtonGroup colorButtons = new ButtonGroup();
colorButtons.add(red);
colorButtons.add(yellow);
colorButtons.add(purple);

// We still need to arrange the buttons in a window to make
// them visible
```
Containers

- Used to group and arrange components for display purposes.
- Since Container is a subclass of Component containers can contain other containers.
- Container objects must belong to a subclass of Container such as:
  
<table>
<thead>
<tr>
<th>Container</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JFrame</td>
<td>(Top-level application windows)</td>
</tr>
<tr>
<td>JDialog</td>
<td>(Dialog boxes)</td>
</tr>
<tr>
<td>JPanel</td>
<td>(Invisible containers)</td>
</tr>
</tbody>
</table>

```java
+ add(comp: Component): void
+ remove(comp: Component): void
+ setLayout(mgr: LayoutManager): void
```

Application Windows (JFrames)

```java
// Open a window on the screen
JFrame window = new JFrame("My App");
window.setSize(800, 600);
window.setVisible(true);
```
**JPanel**

<table>
<thead>
<tr>
<th>JPanel</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ JPanel ()</td>
</tr>
<tr>
<td>+ JPanel (layout: LayoutManager)</td>
</tr>
</tbody>
</table>

- A JPanel is an invisible component that represents a rectangular region of a window.
- JPanels are useful for arranging other components, especially when using layout managers.

**Layout Managers**

- By default, containers use layout managers to keep their contents nicely arranged.
- Different kinds of containers have different defaults for their layout managers.
  - JFrames use a BorderLayout by default
  - JPanels use a FlowLayout by default
- You can change the default by sending a setLayout message:
  ```java
  JFrame frame = new JFrame();
  frame.setLayout( new FlowLayout() );
  ```
- Various layout manager classes are defined in the java.awt package.
FlowLayout

- A flow layout arranges components left to right and top to bottom in the order they are added to the container.
  - Similar to the way a word processor arranges words as you type them.
  - If there are too many components to fit in one row, the flow layout “wraps” to the next row.
  - By default, each row is centered horizontally.
  - If the container changes shape (e.g. because the user resized a window), the components are automatically rearranged.
  - The layout manager sets the size of each component to its preferred size.

FlowLayout in Action
**FlowLayout Constructors**

<table>
<thead>
<tr>
<th>FlowLayout</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ FlowLayout ( )</td>
</tr>
<tr>
<td>+ FlowLayout (align: int)</td>
</tr>
<tr>
<td>+ FlowLayout (align: int, hgap: int, vgap: int)</td>
</tr>
</tbody>
</table>

- For the align parameter use: FlowLayout.CENTER, FlowLayout.RIGHT, or FlowLayout.LEFT
- By default the horizontal and vertical gaps are 5 pixels

**GridLayout**

- A GridLayout divides its container into rows of equal height and columns of equal width.
  - Components are added left to right and top to bottom.
  - Each component is sized to completely fill one rectangle of the grid.
GridLayout in Action

GridLayout Constructors

- The default constructor produces a 1x1 grid.
- In the other constructors, one (but not both) of rows and cols may be zero.
  - If rows is zero, the number of rows will be determined by the number of components in the container.
  - If cols is zero, the number of columns will be determined by the number of components in the container.
**BorderLayout**

- A BorderLayout divides its container into 5 regions:
  
<table>
<thead>
<tr>
<th>North</th>
<th>West</th>
<th>Center</th>
<th>East</th>
<th>South</th>
</tr>
</thead>
</table>

- Each region holds at most one component.
  - But the component could be another container!
- The north and south components are given their preferred heights and fill the width of the container.
- The east and west components are given their preferred widths and fill the space between north and south in the vertical direction.
- The center fills up all the remaining space.

**BorderLayout (2)**

- If there is no component in the north, the north region will have a height of zero.
  - same for south
- If there is no component in the west, the west region will have zero width.
  - same for east
- When adding a component to a container using a BorderLayout, you should specify the region, e.g:
  
  ```java
  myContainer.add(myComponent, BorderLayout.NORTH)
  ```
  - Use the constants defined in the BorderLayout class to specify the region.
BorderLayout in Action

BorderLayout Constructors

<table>
<thead>
<tr>
<th>BorderLayout</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ BorderLayout ( )</td>
</tr>
<tr>
<td>+ BorderLayout (hgap: int, vgap: int)</td>
</tr>
</tbody>
</table>
**Layout Example 1**

```java
JFrame frame = new JFrame("Example One");
frame.setLayout(new BorderLayout());

JTextArea text = new JTextArea();
JScrollPane scroll = new JScrollPane(text);
frame.add(scroll, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();
buttonPanel.setLayout(new FlowLayout(FlowLayout.LEFT));
buttonPanel.add(new JButton("OK");
buttonPanel.add(new JButton("Cancel");
frame.add(buttonPanel, BorderLayout.SOUTH);

frame.setSize(50, 300);
frame.setVisible(true);
```

**Layout Solution 1**

```java
JFrame frame = new JFrame("Example One");
frame.setLayout(new BorderLayout());

JTextArea text = new JTextArea();
JScrollPane scroll = new JScrollPane(text);
frame.add(scroll, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();
buttonPanel.setLayout(new FlowLayout(FlowLayout.LEFT));
buttonPanel.add(new JButton("OK");
buttonPanel.add(new JButton("Cancel");
frame.add(buttonPanel, BorderLayout.SOUTH);

frame.setSize(50, 300);
frame.setVisible(true);
```
Layout Example 2

![Diagram of Example Two]

Layout Solution 2

```java
JFrame frame = new JFrame("Example Two");
frame.setLayout(new GridLayout(1,2));

JPanel leftHalf = new JPanel(new BorderLayout());
JPanel rightHalf = new JPanel(new BorderLayout());
frame.add(leftHalf);
frame.add(rightHalf);

JPanel topLeft = new JPanel(new GridLayout(2,1));
topleft.add(new JButton("A"));
topleft.add(new JButton("E"));
leftHalf.add(topLeft, BorderLayout.NORTH);

JPanel topRight = new JPanel(new GridLayout(2,3));
topRight.add(new JButton("B"));
topRight.add(new JButton("C"));
topRight.add(new JButton("D"));
topRight.add(new JButton("E"));
topRight.add(new JButton("F"));
topRight.add(new JButton("G"));
topRight.add(new JButton("H"));
rightHalf.add(topRight, BorderLayout.NORTH);

frame.setSize(50, 300);
frame.setVisible(true);
```
Some Other Layout Managers

- BoxLayout
- CardLayout
- GridBagLayout