JFrame and JPanel

CSCI 201L

Jeffrey Miller, Ph.D.
jeffrey.miller@usc.edu

HTTP://WWW-SCF.USC.EDU/~CSCI201
Outline

▪ JFrame Class
▪ Adding Components
▪ Panels
▪ Program
JFrame Overview

- javax.swing.JFrame inherits from java.awt.Frame
- The JFrame is typically the outermost container used in GUI applications
  - JApplet could also be the outermost container depending on your application
- JFrame contains a JRootPane, which is where all of the components are added
  - The JRootPane is shared among all container objects to support good OO design
  - When components are added to a JFrame, they are actually added to the content pane (JRootPane)
- To display a window in Java, create a JFrame, set the size, set the location, and set it visible
import javax.swing.JFrame;

public class MyFirstGUI extends JFrame {
    public MyFirstGUI() {
        super("First Frame");
        setSize(500, 300);
        setLocation(100, 100);
        setVisible(true);
    }

    public static void main(String[] args) {
        MyFirstGUI mfgui = new MyFirstGUI();
    }
}
Closing a GUI

- The default operation of the X on the GUI is to set the visibility of the frame to false
  - Note that this does not terminate the program
- Since there may be some clean up that needs to occur before a GUI can be closed, we can add custom code to the close operation
  - We will learn more about this when we talk about event-based programming
- Java gives us the ability to just make the program terminate when the X is clicked on a GUI
  - On the frame, call the `setDefaultCloseOperation` method with `JFrame.EXIT_ON_CLOSE` as a parameter

```java
mfgui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```
Outline

- JFrame Class
- Adding Components
- Panels
- Program
Adding Components to a JFrame

- To add a component to a JFrame, create the component, set its properties, and pass the instance into the `add` method of the JFrame.
- The component will be added to the content pane of the frame.
- If you only add one component, it will take up the entire frame.
Adding Component Example

```java
import javax.swing.JButton;
import javax.swing.JFrame;

public class Test extends JFrame {

    public Test() {
        super("My Frame");
        setSize(300, 400);
        setLocation(200, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JButton helloButton = new JButton("Hello");
        add(helloButton);

        setVisible(true);
    }

    public static void main(String[] args) {
        Test t = new Test();
    }
}
```
import javax.swing.JButton;
import javax.swing.JFrame;

public class Test extends JFrame {

    public Test() {
        super("My Frame");
        setSize(300, 400);
        setLocation(200, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JButton helloButton = new JButton("Hello");
        add(helloButton);

        JButton worldButton = new JButton("World");
        add(worldButton);

        setVisible(true);
    }

    public static void main(String[] args) {
        Test t = new Test();
    }
}
BorderLayout Description

- If we want to add components to different locations on the frame, we need to specify where to add them
- Layout managers allow us to do that
  - This is the topic of our next lecture
- The default layout manager for a JFrame is BorderLayout
  - When you add a component, you can specify you want to add it to the BorderLayout.NORTH, BorderLayout.SOUTH, BorderLayout.EAST, BorderLayout.WEST, or BorderLayout.CENTER
  - Components in the north and south will have their height acknowledged but the width will be expanded to the width of the frame
  - Components in the east and west will have their width acknowledged but the height will be expanded to the height of the frame
  - Components in the center will have their heights and widths expanded to take up whatever space is remaining in the frame after the other four regions are rendered
import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

public class Test extends JFrame {
    public Test() {
        super("My Frame");
        setSize(300, 400);
        setLocation(200, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JButton northButton = new JButton("North");
        add(northButton, BorderLayout.NORTH);
        JButton southButton = new JButton("South");
        add(southButton, BorderLayout.SOUTH);
        JButton eastButton = new JButton("East");
        add(eastButton, BorderLayout.EAST);
        JButton westButton = new JButton("West");
        add(westButton, BorderLayout.WEST);
        JButton centerButton = new JButton("Center");
        add(centerButton, BorderLayout.CENTER);

        setVisible(true);
    }

    public static void main(String[] args) {
        Test t = new Test();
    }
}

Adding Components
Outline

- Java GUI Organization
- JFrame Class
- Adding Components
- Panels
- Program
Panel Overview

- With **BorderLayout**, we can only add one component to each of the five locations.
- If we want to have more than one component in one of the locations, we need to create a panel.
  - We use the `JPanel` class for this.
- A panel is a container (and therefore also a component).
- A panel has its own layout manager, so you can add components to the panel similar to adding them to the frame.
  - The default layout manager of a panel is a `FlowLayout`, which adds components from left to right, top to bottom and acknowledges the component’s requested height and width.
  - A panel can also be added to a panel.
- The panel can then be added to one of the locations in the frame.
import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Test extends JFrame {
    public Test() {
        super("My Frame");
        setSize(300, 400);
        setLocation(200, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel jp = new JPanel();

        JButton helloButton = new JButton("Hello");
        jp.add(helloButton);
        JButton worldButton = new JButton("World");
        jp.add(worldButton);

        add(jp, BorderLayout.NORTH);

        JButton eastButton = new JButton("East");
        add(eastButton, BorderLayout.EAST);
        JButton westButton = new JButton("West");
        add(westButton, BorderLayout.WEST);
        JButton centerButton = new JButton("Center");
        add(centerButton, BorderLayout.CENTER);

        setVisible(true);
    }
}

public static void main(String [] args) {
    Test t = new Test();
}
Outline

- Java GUI Organization
- JFrame Class
- Adding Components
- Panels
- Program
Program

- Write a program that produces the GUI below.