

More About GUIs

COMP163 Introduction to
Computer Programming

“An ounce of practice is worth more than tons of preaching.”

Mahatma Gandhi

Quiz

- There will be a quiz in the COMP163 recitation this week
- The quiz will cover all the material since the beginning of the semester
- The questions will be
 - Complete a program
 - Show what a program displays
- There is a sample quiz on Blackboard under course materials

Exam

The first exam in COMP163 will be on
Monday, September 23, 2019

Inherited Consistency

- Because many of the capabilities of a GUI object come from parent classes, you use the same methods to specify these capabilities for all objects
- An object of the class JButton or JTextField or any of the other GUI classes is also a member of the JComponent class
- Methods to set the color, font, size, position and many other properties are common for all components

Making a GUI

1. Create a class to extend JFrame
2. Create component objects
3. Specify the layout manager
4. Add the component objects to the container
5. Make the GUI visible and stoppable
6. Specify which objects respond to user input

Extending JFrame

- You can create a GUI by extending `javax.swing.JFrame`
- The class `JFrame` provides the “frame” where the GUI will be located

```
public class MyGUI extends  
    javax.swing.JFrame {  
  
}
```

Initializing

- GUIs should have a constructor method

```
public MyGUI () {  
    // GUI initialization  
}
```

- Your main method calls the constructor

```
public static void main(String[] rat) {  
    MyGUI frog = new MyGUI ();  
}
```


Creating a GUI

```
public class MyGUI extends javax.swing.JFrame {  
  
    public MyGUI() {  
        setSize(250, 200);  
    }  
  
    public static void main(String[] args){  
        MyGUI frog = new MyGUI();  
    }  
}
```

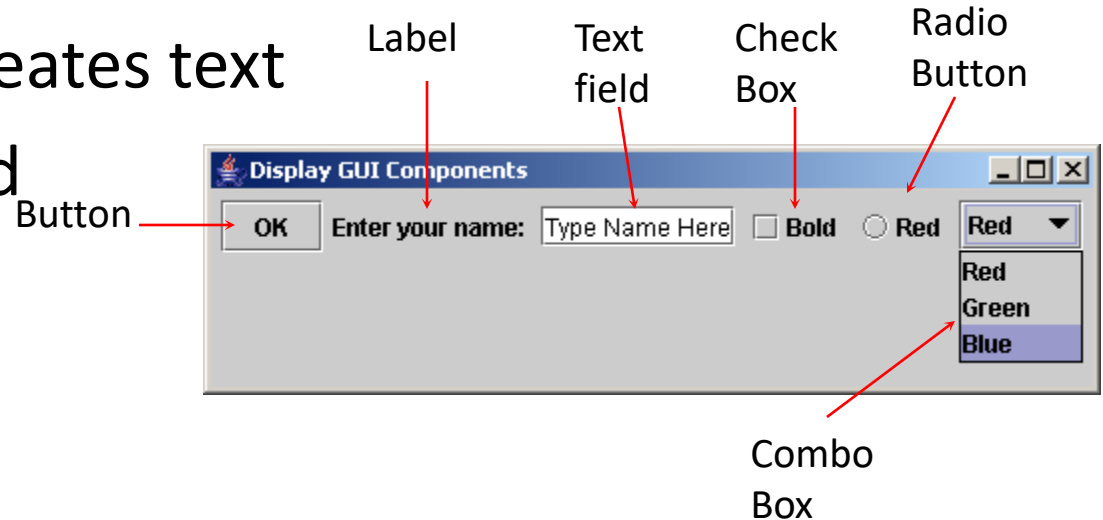
Setting the Frame Size

- You can specify how big the frame will be with the **setSize(xSize, ySize)** method
- The sizes are specified in pixels the same as the graphics methods used

Creating GUI Objects

All of the parts of a GUI are Java objects.

- `javax.swing.JButton` creates buttons
- `javax.swing.JLabel` creates text
- `javax.swing.JTextField` creates input boxes



- `javax.swing.JCheckBox` creates check boxes
- `javax.swing.JRadioButton` creates radio buttons
- `javax.swing.JComboBox` creates menu boxes

Creating Components

- Most Swing components have a constructor that takes one String parameter specifying the text to be displayed in that component

```
javax.swing.JButton dog =  
    new javax.swing.JButton ("Go") ;  
javax.swing.JLabel cat =  
    new javax.swing.JLabel ("COMP163") ;
```

Components as Instance Variables

- The GUI component variables should be defined in the class and **not** in the constructor method
- Calls to methods of the components are often put in the constructor method of a GUI class

Creating Components

```
public class MyGUI extends javax.swing.JFrame {  
  
    javax.swing.JButton dog = new javax.swing.JButton( "Go" );  
    javax.swing.JLabel cat    = new javax.swing.JLabel("COMP163");  
  
    public MyGUI() {  
        setSize(250, 200);  
    }  
  
    public static void main(String[] rat){  
        MyGUI frog = new MyGUI();  
    }  
}
```

Content Pane

- A frame has a content pane that holds the visible components of the GUI, like buttons
- Programs need to get the content pane using the `getContentPane()` method of frame

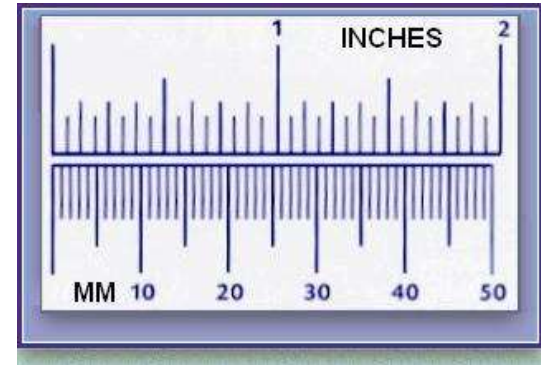
```
java.awt.Container  goat = getContentPane();
```

Creating Components

```
public class MyGUI extends javax.swing.JFrame {  
    javax.swing.JButton dog = new javax.swing.JButton( "Go" );  
    javax.swing.JLabel cat    = new javax.swing.JLabel("COMP163");  
  
    public MyGUI() {  
        setSize(250, 200);  
        java.awt.Container goat = getContentPane();  
    }  
    public static void main(String[] rat){  
        MyGUI frog = new MyGUI();  
    }  
}
```


The size of GUI components is measured in

- A. inches
- B. millimeters
- C. pixels
- D. pixies



Putting Them Where You Want Them

- The **Layout Manager** determines where components will appear in a GUI
- There are several ways to specify position and size
- Some are more portable than others
- Some are more frustrating than others

BoxLayout

- The BoxLayout is relatively simple
- It positions components as “boxes” either top to bottom (Y_AXIS) or left to right (X_AXIS)
- You must create a BoxLayout object

```
javax.swing.BoxLayout whale = new  
javax.swing.BoxLayout(goat, javax.swing.BoxLayout.Y_AXIS);
```

or

```
import javax.swing.*;
```

```
BoxLayout whale =
```

```
    new BoxLayout(goat, BoxLayout.Y_AXIS);
```

Specifying the Layout Manager

- You must specify what layout manager your program will use with the **setLayout** method
- The parameter of the **setLayout** method is any layout manager object

```
javax.swing.BoxLayout whale = new  
    javax.swing.BoxLayout(goat, javax.swing.BoxLayout.Y_AXIS);
```

```
setLayout( whale );
```

Adding Layout Manager

```
public class MyGUI extends javax.swing.JFrame {
    javax.swing.JButton dog = new javax.swing.JButton( "Go" );
    javax.swing.JLabel cat    = new javax.swing.JLabel("COMP163");
    public MyGUI() {
        setSize(250, 200);
        java.awt.Container goat = getContentPane();
        javax.swing.BoxLayout whale = new
            javax.swing.BoxLayout(goat,javax.swing.BoxLayout.Y_AXIS);
        setLayout( whale );
    }
    public static void main(String[] rat){
        MyGUI frog = new MyGUI();
    }
}
```

GUI components should be instance variables of the class declared

- A. in the main method
- B. in the constructor method
- C. in the action performed method
- D. outside of any method

Adding Components to a GUI

- To make them appear in a frame the components must be added to the content pane
- The add method links a component to a content pane object

```
goat.add( cat );  
goat.add( dog );
```
- The order in which the components are added determines the order in which they will appear in the GUI

Adding Components

```
public class MyGUI extends javax.swing.JFrame {
    javax.swing.JButton dog = new javax.swing.JButton( "Go" );
    javax.swing.JLabel  cat = new javax.swing.JLabel("COMP163");
    public MyGUI() {
        setSize(150, 100);
        java.awt.Container goat = getContentPane();
        javax.swing.BoxLayout whale = new
            javax.swing.BoxLayout(goat,javax.swing.BoxLayout.Y_AXIS);
        setLayout( whale );
        goat.add( cat );
        goat.add( dog );
    }
    public static void main(String[] rat){
        MyGUI frog = new MyGUI();
    }
}
```


Making It Start and Stop

- To make your program terminate if you press the red **X** in the upper right corner, include

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

- Call `setVisible(true)` to make the frame appear onscreen

The Full GUI Initialization

```
public class MyGUI extends javax.swing.JFrame {
    javax.swing.JButton dog = new javax.swing.JButton( "Go" );
    javax.swing.JLabel  cat = new javax.swing.JLabel("COMP163");
    public MyGUI() {
        setSize(150, 100);
        java.awt.Container goat = getContentPane();
        javax.swing.BoxLayout whale = new
            javax.swing.BoxLayout(goat,javax.swing.BoxLayout.Y_AXIS);
        setLayout( whale );
        goat.add( cat );
        goat.add( dog );
        setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
        setVisible( true );
    }
    public static void main(String[] rat){
        MyGUI frog = new MyGUI();
    }
}
```

Jar Files



- A jar file is a collection of compiled Java classes

```
>jar tf BMPfile.jar // display contents of jar
```

```
  META-INF/MANIFEST.MF
```

```
  BMPcomponent.class
```

```
  BMPfile.class
```

- The manifest file contains information about the files in the jar

Nothing Happens?

- When you press the button of our example GUI program, nothing happens
- You can specify a method to be called whenever a button is pressed or something is done to a component

Steps to Respond to Actions

To make an GUI respond in user input:

- Implement `java.awt.event.ActionListener`
- Add an action listener to the component
- Create an `actionPerformed` method

Implementing Interfaces

- Implementing an interface is similar to extending a class
- Interfaces are an advanced feature to be explained in detail in COMP167

```
public class MyGUI  
    extends javax.swing.JFrame  
    implements java.awt.event.ActionListener {
```

Listeners



- Java has the concept of listeners
- Listener methods will be called when something happens
- There is a `MouseListener`, `KeyListener`, `FocusListener`, `CellEditorListener` and many more
- An `ActionListener` calls a method when the user takes some action with a component in the GUI

Adding a Listener

- The method **addActionListener** should be called on each GUI component that can do something
- Do not call `addActionListener` on components that do not do anything, like `JLabel`

```
javax.swing.JButton kangaroo = new  
    javax.swing.JButton("Go");
```

...

```
kangaroo.addActionListener( this );
```


Who Does It

- The parameter to the addActionListener method is the name of an object that implements ActionListener and has an actionPerformed method
- If the actionPerformed method is in the same class that calls addActionListener, the usual parameter is **this**
- **this** is the name of *this* object

What Happens

- When a user clicks an object with an ActionListener, Java calls the method

```
public void actionPerformed(  
    java.awt.event.ActionEvent chicken) {  
    // do something here  
}
```

- This method should do what the user expects to happen when the button is pressed

GUI Structure

- Class extending JFrame and implementing ActionListener
- GUI component objects defined
- Constructor method
 - Initializes the GUI
 - Called only once at the beginning
- ActionPerformed method
 - Gets input and sets output
 - Called each time something is clicked

When a button is pressed, Java calls

- A. actionPerformed
- B. add
- C. actionPerformed
- D. constructor

Changing a Component's Text

- A useful method is `setText`. It can be used to change the text of any object that has text, such as `JButtons`, `JLabels` and `JTextfields`.

```
javax.swing.JLabel trout =  
    new javax.swing.JLabel("Do something");
```

```
trout.setText("Don't do something");
```

GUI Output

- When a program with a GUI wants to display a result, it usually puts the value in a JLabel object.

```
JLabel aardvark = new JLabel();
```

```
...
```

```
double answer = some equation;
```

```
aardvark.setText("The answer is "+answer);
```

```

public class MyGUIaction extends javax.swing.JFrame implements
                                java.awt.event.ActionListener {
    javax.swing.JButton dog = new javax.swing.JButton( "Go" );
    javax.swing.JLabel  cat = new javax.swing.JLabel("COMP163");
    int counter = 0;
    public MyGUIaction() {
        setSize(200, 100);
        java.awt.Container goat = getContentPane();
        javax.swing.BoxLayout whale = new
            javax.swing.BoxLayout(goat, javax.swing.BoxLayout.Y_AXIS);
        setLayout( whale );
        goat.add( cat );
        goat.add( dog );
        dog.addActionListener( this );
        setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
        setVisible( true );
    }
    public void actionPerformed(java.awt.event.ActionEvent thing) {
        counter = counter + 1;
        cat.setText("pressed " + counter + " times");
    }
    public static void main(String[] rat){
        MyGUIaction frog = new MyGUIaction();
    }
}

```

Making a GUI

1. Create a class that extends JFrame and implements `java.awt.event.ActionListener`
2. Create component objects
3. Specify the layout manager
4. Add the component objects to the container
5. Add an action listener to the components that respond to user input
6. Create an `actionPerformed` method

Change of Thinking



- Input is usually from JTextField objects
 - java.util.Scanner is **not** used
- Output is usually to JLabel objects
 - System.out is **not** used

GUI Execution

Start in the main method
Create object of the class

Constructor method
Initializes the GUI

ActionPerformed method
Do what the program is supposed
to do when you press a button

String and Numbers

- A String can contain any character on the keyboard, including the numbers

```
String myGrade = "4.0";
```

```
double myScore = 4.0;
```

- You cannot do arithmetic with Strings
- A string containing numerical characters must be converted to a double or an int for use in any calculations

Converting Strings to Numbers

- A String that contains numerical characters can be converted using

```
String four = "4", deuce = "2.25";
```

```
double cow = Double.parseDouble( deuce );
```

```
int goat = Integer.parseInt( four );
```

- These methods will throw an exception if the string does not contain a number

Inputting Numbers

- The `getText()` method returns a `String`
- If a number is entered in a `JTextField`, it will have to be converted to an `int` or `double`

```
JTextField gekko = new JTextField();
```

```
// in the actionPerformed method
```

```
String dragon = gekko.getText();
```

```
double lizard = Double.parseDouble( dragon );
```

getText and setText

- The `getText` and `setText` methods can be called on all of the GUI components
- The `actionPerformed` method of most programs
 - Call `getText` to get the input
 - Calculate a value
 - Call `setText` to display the result in the GUI

Example Numeric Input

```
import javax.swing.*;
public class GUImult extends JFrame implements
        java.awt.event.ActionListener {
    JTextField inAmount = new JTextField();
    JLabel      answer   = new JLabel("Enter a number ");
public GUImult() {
    setSize(200, 70);
    java.awt.Container pane = getContentPane();
    BorderLayout where = new BorderLayout(pane, BorderLayout.X_AXIS);
    setLayout( where );
    pane.add( answer );
    pane.add( inAmount );
    inAmount.addActionListener( this );
    setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
    setVisible( true );
}
public void actionPerformed(java.awt.event.ActionEvent thing) {
    String numStr = inAmount.getText();
    double amount = Double.parseDouble( numStr );
    answer.setText("Twice the number "+amount*2.0+" ");
}
public static void main(String[] cow) {
    GUImult horse = new GUImult();
}
```

Quiz

- There will be a quiz in the COMP163 recitation this week
- The quiz will cover all the material since the beginning of the semester
- The questions will be
 - Complete a program
 - Show what a program displays
- There is a sample quiz on Blackboard under course materials

Exam

The first exam in COMP163 will be on
Monday, September 23, 2019