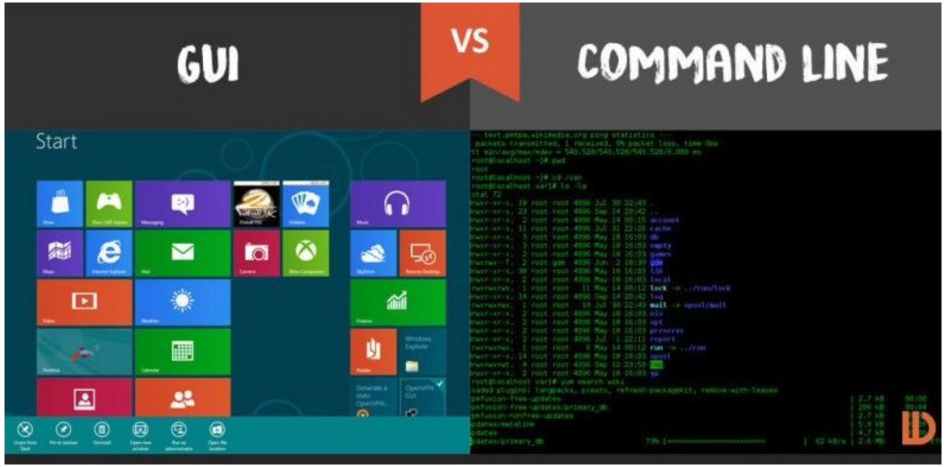
Graphical User Interfaces(GUI) Review

What is a GUI?

- Graphical User Interface
- User friendly way to interact with a program
- Allows the use of buttons and images rather than text



https://www.google.com/url?sa=i&source=images&cd=&ved=2ahUKEwjarPH76t_kAhXkw1kKHfDXAbcQjRx6BAgBEAQ&url=https%3A%2F%2Fsciencerack.com%2Fdifferences-between-command-line-interface-and-gui%2F&psig=AOvVaw3N6O00TkvhmFaOgUROsbpd&ust=1569083841961145

Creating a GUI in Java

- The class of a GUI in Java must be defined in a different way than regular Java programs

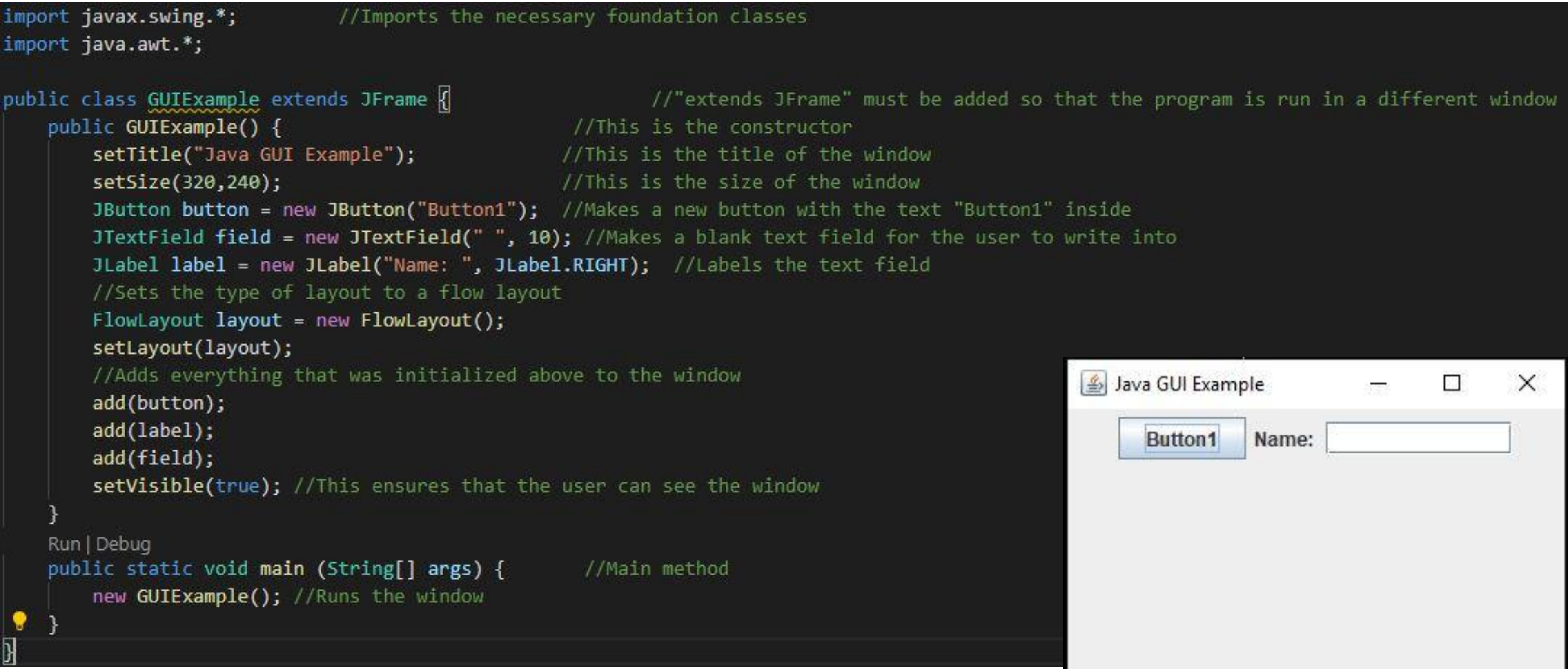
- Java organizes components in the application window in very specific ways

- To create the different components of the interface, different built-in methods are used

Coding a GUI in Java

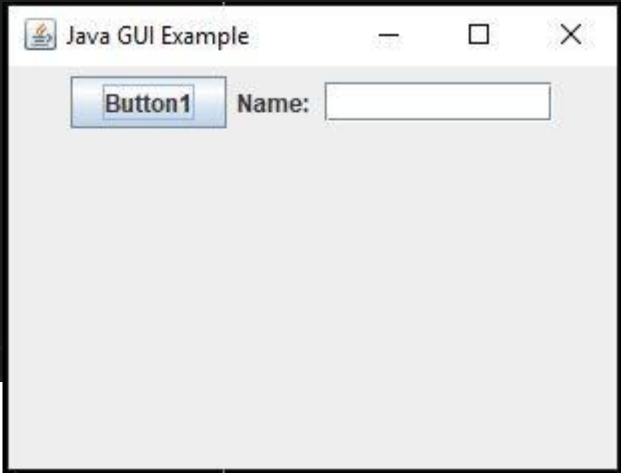
- Java foundation classes must be imported
  - javax.swing.*;
  - java.awt.*;
- Class/Instance variables can go outside of the GUI class (defining buttons, etc.)
- Methods that initialize the frame in a special method called the *constructor*, which has the same name as the GUI class
  - setTitle(“text”);
  - setSize(x, y);
  - setVisible(true);
  - add();
- Instance and class methods go afterwards

Sample Program For a GUI in Java



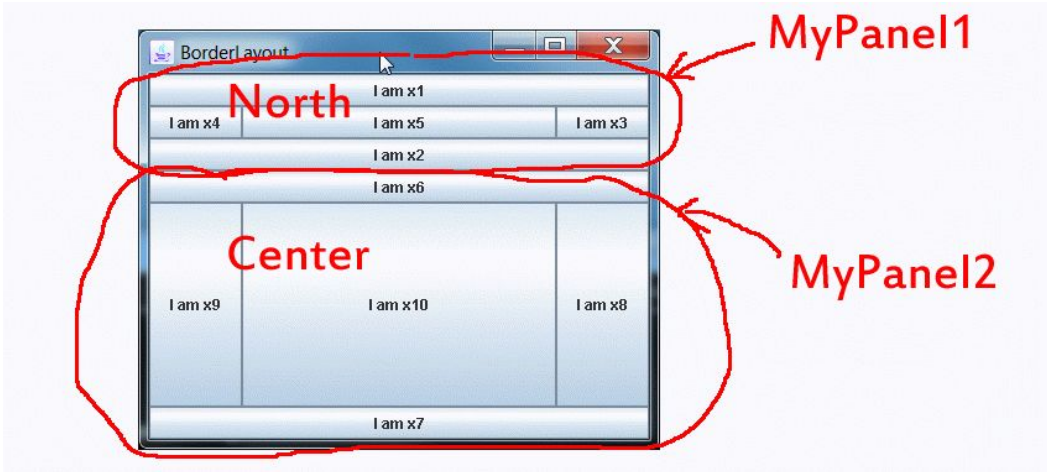
```
import javax.swing.*;         //Imports the necessary foundation classes
import java.awt.*;

public class GUIExample extends JFrame {                    //"extends JFrame" must be added so that the program is run in a different window
    public GUIExample() {                                  //This is the constructor
        setTitle("Java GUI Example");                     //This is the title of the window
        setSize(320,240);                                 //This is the size of the window
        JButton button = new JButton("Button1");  //Makes a new button with the text "Button1" inside
        JTextField field = new JTextField(" ", 10); //Makes a blank text field for the user to write into
        JLabel label = new JLabel("Name: ", JLabel.RIGHT);  //Labels the text field
        //Sets the type of layout to a flow layout
        FlowLayout layout = new FlowLayout();
        setLayout(layout);
        //Adds everything that was initialized above to the window
        add(button);
        add(label);
        add(field);
        setVisible(true); //This ensures that the user can see the window
    }
    Run | Debug
    public static void main (String[] args) {       //Main method
        new GUIExample(); //Runs the window
    }
}
```



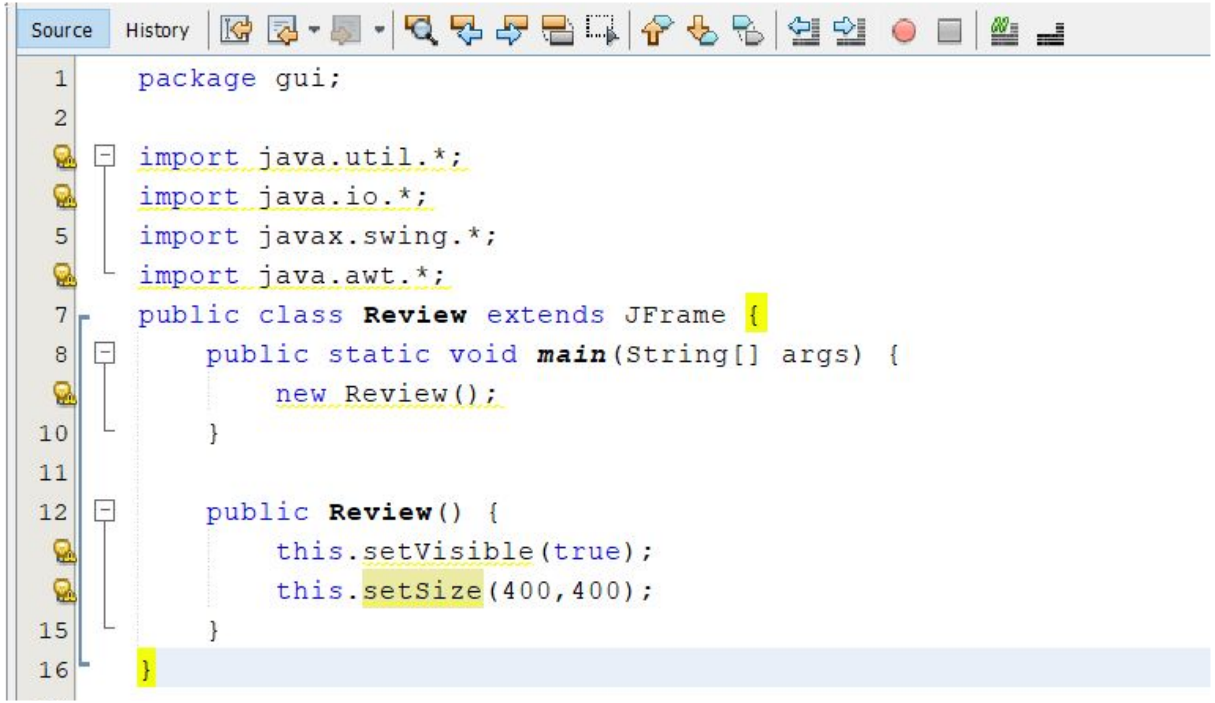
Frames and Panels

- JFrame: A window/container for other elements (buttons, images, etc)
- JPanel: A container for elements that can go inside a JFrame



https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjd2O23rePkAhWxLn0KHSUpAGAQjhx6BAgBEAI&url=http%3A%2F%2Fwww.mathcs.emory.edu%2F~cheung%2FCourses%2F377%2FSyllabus%2F8-JDBC%2FGUI%2Flayout.html&psig=AOvVaw24SEFOLYPMEviXUOe1ZRHv&ust=1569204751552610

Creating a JFrame



```
package gui;

import java.util.*;
import java.io.*;
import javax.swing.*;
import java.awt.*;
public class Review extends JFrame {
    public static void main(String[] args) {
        new Review();
    }

    public Review() {
        this.setVisible(true);
        this.setSize(400,400);
    }
}
```

Containers - Labels

- Text that is displayed on the GUI
    - Labels are often used to label text fields
- Code to create a label :
    - JLabel name = new JLabel(“Name: “, JLabel.RIGHT);
      Variable name = name
      “Name: “ will be displayed on the screen
- Placement and the name are set

Containers - Text Fields and Buttons

- User can type inside of text fields
- Code to create a text field:
  - JTextField nameField = new JTextField(“ “, 30);
- If text is put in the quotations the text field will not be blank
  - JTextField nameField = new JTextField(“Bob”, 30);
  - “Bob” will appear in the text field
- User can click on buttons
- Code to create a button:
  - JButton button = new JButton(“OK”);
  - Makes button called “OK”

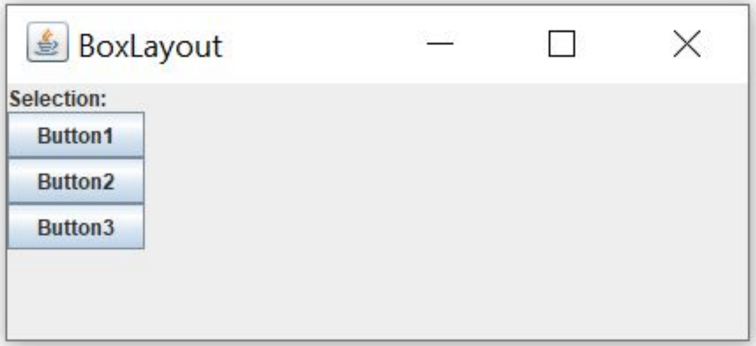
Layouts

- Layouts must be easy to understand and use for the user
- Helps to organize the components on the frame
- Java includes layout managers:
    - BoxLayout
    - GridLayout
    - FlowLayout

BoxLayout

- Every item will be placed in a single row or column
- BoxLayout can also be set up with rigid areas and glue areas. This allows you to add some space between items and/or force items to one side of the area

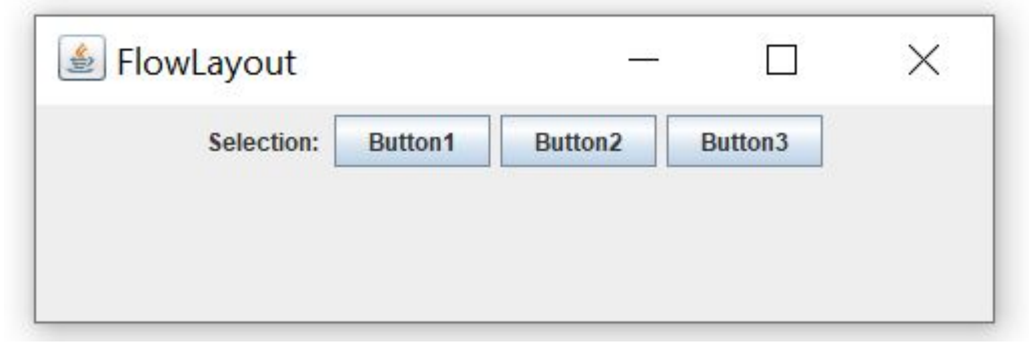
```
BoxLayout layout1 = new BoxLayout(panel,BoxLayoutLayout.Y_AXIS);
//Add in containers to panel
panel.setLayout(layout1);
```



FlowLayout

- Puts each item into a single row and starts a new row where there is no more space left
- Can be setup with alignment details, horizontal spacing, and vertical spacing

```
FlowLayout layout2 = new FlowLayout();
setLayout(layout1);
```

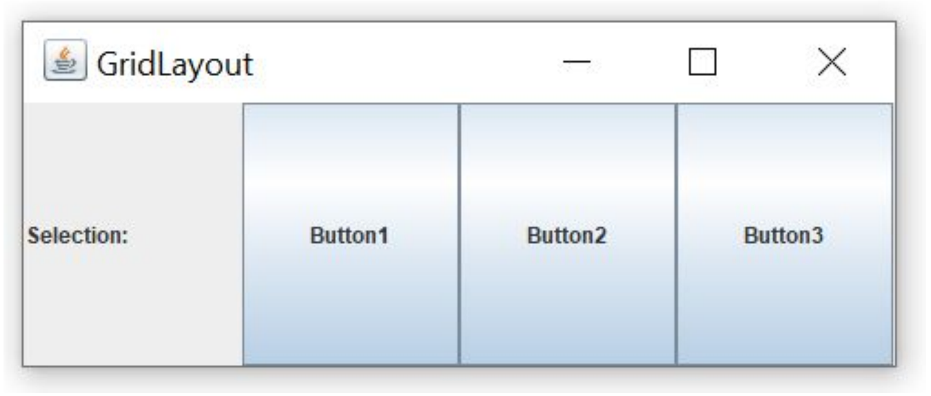


Grid Layout

- Puts all items in rows and columns and makes them all equal in size
- Can be set up with the number of rows and columns required and the spacing details

```
GridLayout layout3 = new GridLayout(2,1);
setLayout(layout3);
```

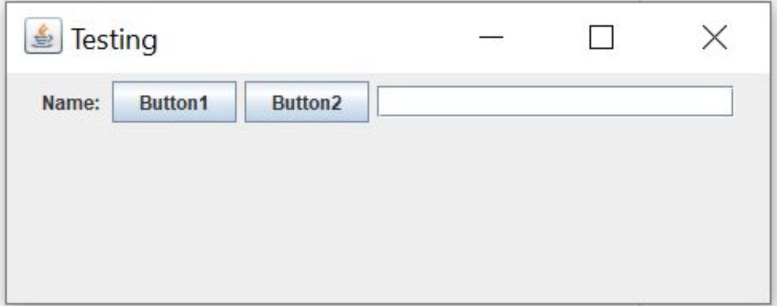
rows columns



```java
//Imports needed to use GUI
import javax.swing.*;
import java.awt.*;
public class ReviewGUI4 extends JFrame{
  public static void main (String[]args){
    JFrame newFrame = new JFrame ("Testing"); //Creates a new frame
    newFrame.setVisible(true); //Makes it so that the user can see the frame
    newFrame.setSize(500,200); //Sets size of the frame

    JLabel name = new JLabel("Name: "); //Creates a label that will display "Name: "
    JButton button = new JButton("Button1"); //Creates a button called "Button1"
    JButton button2 = new JButton("Button2"); //Creates a button called "Button2"
    JTextField nameField = new JTextField(" ", 20); //Creates a text field
    JPanel panel = new JPanel(); //Creates a new panel
    FlowLayout layout1 = new FlowLayout(); //Flow layout

    panel.add(name); //Adds Label
    panel.add(button); //Adds button to panel
    panel.add(button2); //Adds button
    panel.add(nameField); //Adds text field to
    panel.setLayout(layout1); //Sets layout
    newFrame.add(panel); //Adds panel to frame
  }
}
```



Type Casting

- Converting from one data type to another
  - Memory efficiency, some data types occupy less memory than others
  - Position of the original variable is lost
- Widening/automatic/implicit conversion
  - Two data types are automatically converted when:
    - Two data types are compatible.
    - Assign value of a smaller data type to a bigger data type

Byte –> Short –> Int –> Long – > Float –> Double

Widening or Automatic Conversion

- Narrowing/explicit conversion
  - Two data types have to be manually converted
    - Incompatible data types
    - Specify desired type

Double –> Float –> Long –> Int –> Short –> Byte

Narrowing or Explicit Conversion

```
class Test
{
        public static void main(String[] args)
        {
                int x = 100;

                //automatic type conversion
                long y = x;

                //automatic type conversion
                double z = y+0.4;
                System.out.println("Int value "+x);
                System.out.println("Long value "+y);
                System.out.println("Float value "+z);
        }
}
```

Output:

```
Int value 100
Long value 100
Float value 100.4
```

Byte –> Short –> Int –> Long – > Float –> Double

Widening or Automatic Conversion

```
class Test
{
        public static void main(String[] args)
        {
                double d = 100.04;

                //explicit type casting
                long l = (long)d;

                //explicit type casting
                int i = (int)l;
                System.out.println("Double value "+d);

                //fractional part lost
                System.out.println("Long value "+l);

                //fractional part lost
                System.out.println("Int value "+i);
        }
}
```

Output:

```
Double value 100.04
Long value 100
Int value 100
```

Double –> Float –> Long –> Int –> Short –> Byte

Narrowing or Explicit Conversion

Parsing

- Return type method that converts the string into its integer equivalent
    - String to integer

      ```
      int number = Integer.parseInt(stringVariable);
      //The I in Integer is capitalized
      ```

    - String to double

      ```
      double decimal = Double.parseDouble(stringVariable);
      ```

```java
String number = "10";
int result = Integer.parseInt(number);
System.out.println(result);
```

Output:

```
10
```

Action Listeners

- Java uses action listeners to detect user interaction (button presses)
- When the user performs an action, java automatically calls the action listener method
- You must implement actionlistener into your class
- It is important that you add an actionlistener to each UI element

```
button.addActionListener(this);
```

Action Performed Method

```
public void actionPerformed(ActionEvent ae) {
    String action= ae.getActionCommand();
    if (action.equals(“Button")) {
      //Action Detected!
    }
}
```