Week 8 – Graphical User Interfaces

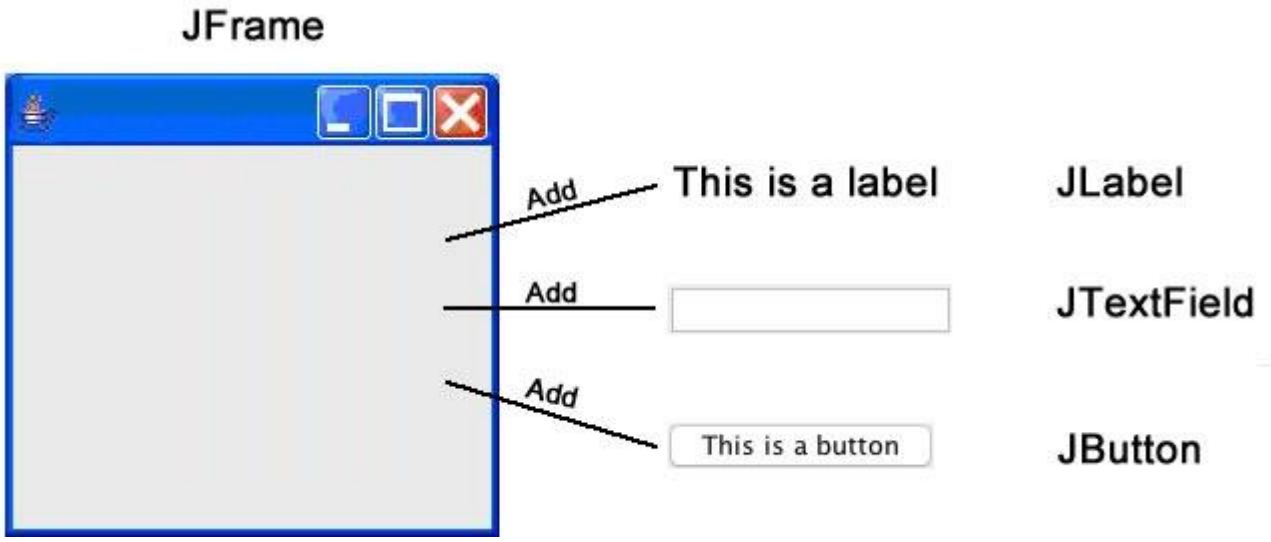
Up until this stage, you have been developing “console applications.” Generally, it is better to provide users with an easier to use and more visually appealing ways of interacting with your program.

This lesson aims to allow you to develop simple, single screen GUIs using the **Swing** library

JFrame

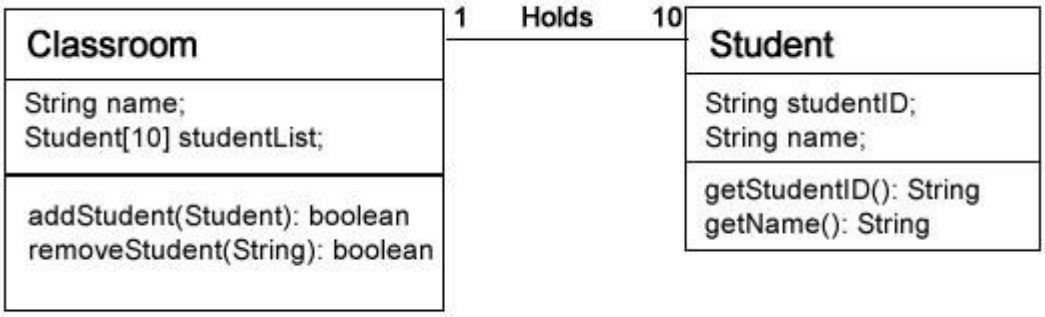
First, we will create a window, to do this we use the swing library’s JFrame class.

A JFrame acts as a window, which you can add components to:



This Weeks Example

This example will be based off a very simple classroom management system:



Author: Darren Cunningham
Version: 1.0

Week 8 – Graphical User Interfaces

We will be adding a GUI to the above system that looks like this:

Creating a JFrame

First, we will create a class to represent our GUI screen. This will inherit from JFrame (which will need to be imported)

```
import javax.swing.JFrame;

public class MainScreen extends JFrame
{

}
```

We can then add the components in as attributes. The screen shot of the GUI above suggests we will need:

- 3 JLabels
- 2 JTextFields
- 2 JButtons

These will be added as attributes of our MainScreen class:

```
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class MainScreen extends JFrame
{
	private JLabel lblTitle;
	private JLabel lblStudentID;
	private JLabel lblStudentName;

	private JTextField txtStudentID;
	private JTextField txtStudentName;

	private JButton btnAdd;
	private JButton btnRemove;
}
```

Week 8 – Graphical User Interfaces

Constructor

The constructor will play a big role in this class.
We are going to first make it set our application to terminate if the window is closed. Then we are going to set the “layout manager” (explained below)

```
public MainScreen()
{
	this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	this.setLayout(new GridBagLayout());
}
```

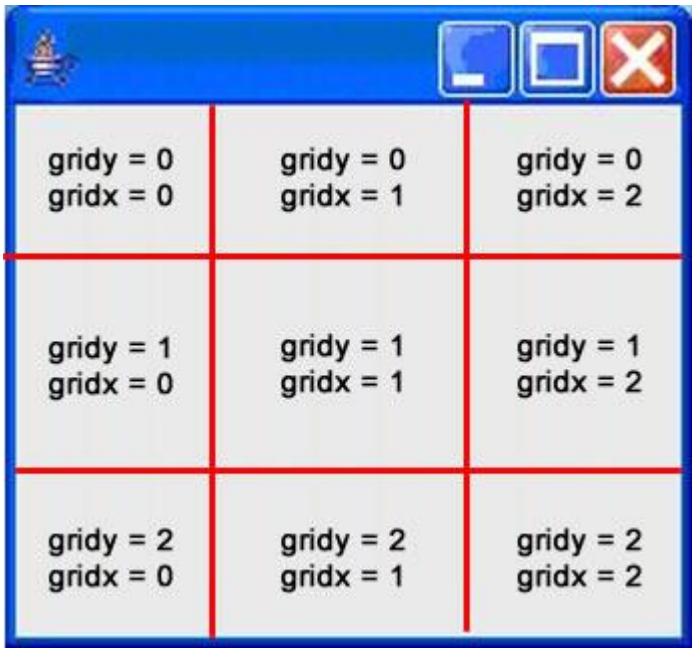
Layout Managers and the GridBagLayout

When working with swing, we typically apply layout managers to our container that helps position the components on the screen.

Swing has a number of layout managers, each of which behave in their own individual way.

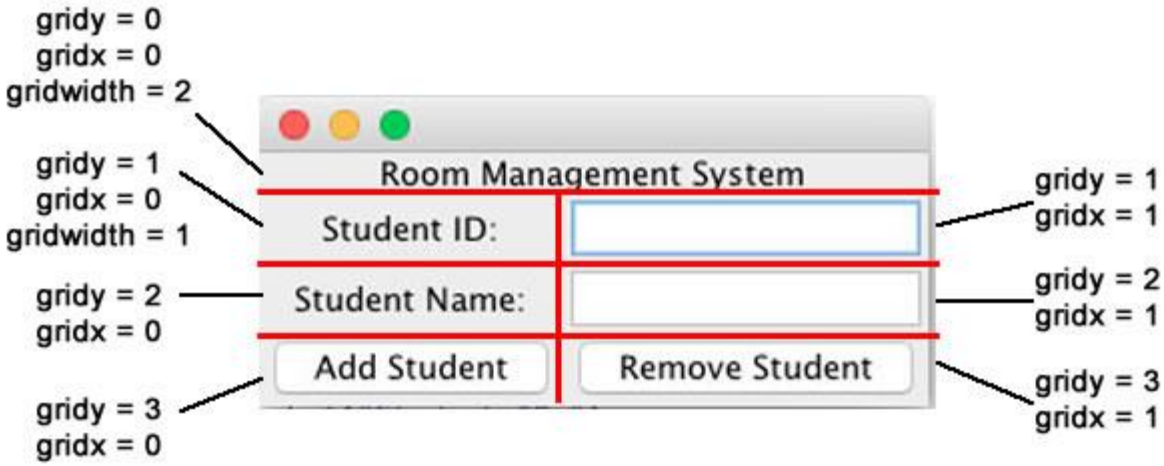
We will be focusing on a layout manager called GridBagLayout.

As the name suggests, GridBagLayout divides the screen in to grids and positions the components within this grid



Author: Darren Cunningham
Version: 1.0

This is how the GridBagLayout translates in to our design:



Initialising Components

Once we have set the layout manager, we can now initialise our components and then position them on the screen.
We could just do this in the constructor, but due to the amount of code, it would be easier to read if we split this in to methods. We can then call these methods from the constructor

Our first method will be called initComponents() and will simply populate our attributes with objects:

```
public void initComponents()
{
	lblTitle = new JLabel("Room Management System");

	lblStudentID = new JLabel("Student ID:");
	lblStudentName = new JLabel("Student Name:");

	txtStudentID = new JTextField();
	txtStudentName = new JTextField();

	btnAdd = new JButton("Add Student");
	btnRemove = new JButton("Remove Student");
}
```

We can now call this method from our constructor:

```
public MainScreen()
{
        this.setLayout(new GridBagLayout());
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Set up components
        initComponents();
}
```

GridBagConstraints

Now it's time to add our components to the screen in the correct position.

First, you will need to know that GridBagLayout uses an object to manage coordinates and other information

This requires us to do a bit of set up. Firstly, we should import the GridBagConstraints class

```
import java.awt.GridBagConstraints;
```

Include a GridBagConstraints object as an attribute of our class:

```
private GridBagConstraints constraints;
```

Finally, initialise it in our constructor:

```
constraints = new GridBagConstraints();
```

Week 8 – Graphical User Interfaces

Creating the Layout

With the GridBagConstraints object set up, we can now create a method that positions our components.

```
public void layoutComponents()
{
	constraints.gridy = 0;	// Set the row
	constraints.gridx = 0;	// Set the column
	// This column will span two columns
	constraints.gridwidth = 2;

	this.add(lblTitle, constraints); // Add the component

	constraints.gridy = 1;		// New row
	constraints.gridx = 0;
	constraints.gridwidth = 1;
	this.add(lblStudentID, constraints);

	constraints.gridx = 1;
	// Makes the textbox fill the width of the column
	constraints.fill = GridBagConstraints.HORIZONTAL;
	this.add(txtStudentID, constraints);

	constraints.fill = GridBagConstraints.NONE;
	constraints.gridy = 2;
	constraints.gridx = 0;
	this.add(lblStudentName, constraints);

	constraints.fill = GridBagConstraints.HORIZONTAL;
	constraints.gridx = 1;
	this.add(txtStudentName, constraints);

	constraints.fill = GridBagConstraints.NONE;
	constraints.gridy = 3;
	constraints.gridx = 0;
	this.add(btnAdd, constraints);

	constraints.gridx = 1;
	this.add(btnRemove, constraints);
}
```

Author: Darren Cunningham
Version: 1.0

We can now call this method from our constructor:

```
public MainScreen()
{
	this.setLayout(new GridBagLayout());
	this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	constraints = new GridBagConstraints();

	// Set up our components
	initComponents();

	// Create layout
	layoutComponents();
}
```

The Main Method

We can now create our GUI in the main method:

```
public static void main(String[] args)
{
	MainScreen gui = new MainScreen();
	gui.pack();
	gui.setVisible(true);
}
```

Note. pack() simply auto sizes the GUI

Making the GUI Work

The GUI should now display, but the trouble is, if you were to press any of the buttons, nothing will happen.
We now have to write code that tells the program to do something if the user clicks a button:

First, remember that this system is for a classroom management system (see class diagram above)
For the system to be able to do anything, we need a Classroom object. We shall add this to the class as an attribute

```
private Classroom room;
```

This should be created in your constructor

```
room = new Classroom("Room Name");
```

Week 8 – Graphical User Interfaces

ActionListeners

ActionListeners allow our GUI to respond to events.

First, we will make our MainScreen class implement ActionListener

```
public class MainScreen extends JFrame implements ActionListener
```

When implementing ActionListener, you MUST override it's actionPerformed method

```
public void actionPerformed(ActionEvent ev)
{

}
```

Upon doing this, we have given our class the ability to listen out for events. We now need to tell our buttons that when they are clicked, the "actionPerformed" method of our class should be called:

```
btnAdd.addActionListener(this)
```

ActionPerformed

Now we can write what happens when a button is clicked. Our actionPerformed method should detect what button was clicked and then carry out the appropriate action

```
public void actionPerformed(ActionEvent ev)
{
    if (ev.getSource().equals(btnAdd))
    {
            String id = txtStudentID.getText();
            String name = txtStudentName.getText();

            if(room.addStudent(new Student(id, name)))
            {
                    JOptionPane.showMessageDialog(null, "Student
has been added");
            }
            else
            {
                    JOptionPane.showMessageDialog(null, "The room
is full");
            }
    }
}
```

Author: Darren Cunningham
Version: 1.0

Week 8 – Graphical User Interfaces

Please find an accompanying page on SOL which contains a fully coded version of this class

Author: Darren Cunningham
Version: 1.0