**Tutorial 3**

At the end of this tutorial you should be comfortable with the following:-

- Create an ArrayList of shapes in your drawing program that use polymorphism (watch the previous week's lecture)
- Write a simple Graphical User Interface using SWING

**Ex1 Adding a List of Shapes**

1. Open the ShapeDraw program that you wrote in Tutorial 2.
2. In the constructor of the Drawing class, create an ArrayList<Shape> and write loops to add a number of objects of class Circ, Rect and Square to the list.
3. In the Canvas's paint method, call the draw methods for each element in the ArrayList.
4. Rather that use an abstract method 'draw' in Shape, implement it as an `interface`.

We are now going to create a Graphical User Interface (GUI) using a library called SWING

**Ex2 Create a Simple Window**

We are going to create a JFrame. This is a component of the SWING user interface library that implements a window.

1. Create a new Gradle project and add it to Github – call it something like SimpleUI.
2. Add the application plugin to build.gradle as in the previous tutorial, and set source compatibility 1.8.
3. Create a new class in the src->main->java folder in the project window called SimpleUI.
4. Add the following code to the class:-

```
static GraphicsConfiguration gc;   // Class field containing config info
public static void main(String[] args) {
    JFrame frame= new JFrame(gc);  // Create a new JFrame
    frame.setSize(500,300);
    frame.setVisible(true);
    // This next line closes the program when the frame is closed
    frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

}
```

5. IntelliJ should ask you whether it can include the correct imports – agree. Build the program from View->Tool Windows->Gradle window by running the build task.
6. Run the program.

**Ex3 Add a Panel Containing a Button**

A JPanel is a window that can reside inside a frame and can contain controls – or other JPanels. We are going to create one and add a button to it. We are going to create also the code that is called when the button is pressed.

1. Create a new class called something like ButtonPanel that extends JPanel.
2. Create a field of class JButton and instantiate it in the constructor. The constructor of JButton can take a character string which will be the text on the button.
3. Call a method of JButton called AddActionListener – pass into the method: "new ActionListener()" – but as you start to type ActionListener IntelliJ should prompt you with a drop down list – select ActionListener in the list and it will automatically add some code so that you get the following:-

```
but1=new JButton("Press me");
but1.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {

    }
});
```

   This has created the code which will be called when the button is pressed.
4. Add the following line which calls the JPanel.add method, adding the button to the JPanel.

```
add(but1);
```

5. Inside the actionPerfomed method, do something like print a message to the console.

6. Finally, in the SimpleUI class, before `frame.setVisible(true);` put:-
   `frame.getContentPane().add(new ButtonPanel());` This adds the JPanel to the JFrame
7. Build and run the program – you should see a button inside the window and pressing it should execute the action you put in actionPerformed.

**Ex4 Controlling Layout**

The layout of controls within a JFrame and a JPanel can be controlled by a layout manager. We will now use an object of class GridLayout, whose constructor takes the number of rows and columns of the grid and where each box in the grid can contain a control.

1. In the ButtonPanel constructor, put the following and then build and run the program:-
   `setLayout(new GridLayout(1,1)); // One row, one column`
2. Try creating a larger grid, and add more buttons to it.

**Ex5 Further Controls**

A description of all SWING components can be found here:-
https://docs.oracle.com/javase/tutorial/uiswing/components/componentlist.html
Examples can be seen here:-
http://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html

1. Try adding different components – eg check boxes, radio buttons, text areas - to the grid layout.
2. There are also layout managers other than GridLayout – try a couple of different ones:-

https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

**Ex6 Menus**

Here is a class that creates a simple JMenuBar with one menu and one menu item.

```
// MainMenu is a JMenuBar that also implements the ActionListener interface by
// having a concrete actionPerformed method
public class MainMenu extends JMenuBar implements ActionListener{
    JMenu menu;         // One menu
    JMenuItem item;     // Which has one item
    public MainMenu(){
        menu=new JMenu("The Menu");
        item=new JMenuItem("Item 1");
        item.addActionListener(this);
        add(menu);                          // Add the JMenu to the JMenuBar
        menu.add(item);                     // Add the JMenuItem to the JMenu
    }

    @Override
    // If 'this' is set as the ActionListener to all menu items, this method will
    // be called whenever any menu item is selected, but the ActionEvent
    // command will be the text of the menu item selected
    public void actionPerformed(ActionEvent e) {
        System.out.println(e.getActionCommand());
    }
}
```

1. Add this class to your project, instantiate it and add it to the frame using the `setJMenuBar` method of frame.
2. Expand on your menu to include more items.

**Ex7 Adding a Canvas**

Going back to the shape drawing program, the Drawing class extended the Canvas class. Canvases can also be added to JPanels as if they were components. Try it! Make sure that your shapes are drawing in the visible area of the canvas. Try adding your own menu items to perform various graphical operations of your own invention – eg adding more shapes, enlarging shapes, changing colours.