Prerequisite 1

# As part of the requirements, we will use Java Swing to create a GUI. Using this GUI we will be able to render shapes. Those shapes will utilize inheritance, abstract classes, polymorphism, and (hopefully) interfaces to provide us with practical experience of these concepts.

I will attempt to provide you with some basic building blocks that should be relatively simple to understand and utilize for our needs (if you are not familiar with Swing.)

The appendices in this document provide you with such a set of building blocks to get you started. The comments in the code should help you understand the basic idea. Additionally, there are a few exercises contained within the comments of the JPanel class for you to attempt.

## public class WindowCreator {

public static void main(String[] args) {

*//Create and configure our JFrame (window)*

CustomWindow customWindow = new CustomWindow(); customWindow.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*); customWindow.setTitle("Test Window"); customWindow.setVisible(true);

}

}

import java.awt.\*;

*/\*\**

* *We can think of a JFrame as a window*
* *By extending (inheriting from) the class javax.swing.JFrame we can*
* *define what goes into our window - in this case a single JPanel which is*
* *a component container for GUI elements.*

*\*/*

public class CustomWindow extends JFrame { private CustomPanel mainPanel;

public CustomWindow()

{

mainPanel = new CustomPanel();

*//add our new panel to the frame*

add(mainPanel, BorderLayout.*CENTER*);

*//set the dimensions of the frame/window*

setSize(Consts.*FRAME\_WIDTH*, Consts.*FRAME\_HEIGHT*);

## }

}

import javax.swing.\*; import java.awt.\*;

//import java.awt.event.MouseAdapter;

//import java.awt.event.MouseEvent;

//import java.util.ArrayList;

/\*\*

* *This allows us to create a panel which we can add to a frame/window*
* *Oftentimes, you would then add standard GUI components, e.g. JButton, JLabel, to the*
* *panel.*
* *In our case, though, we will want to draw shapes, so we override the paintComponent() method*
* *that we inherit from the javax.swing.JPanel class.*
* *The graphics system passes us a java.awt.Graphics object and this has methods which allows us*
* *to draw shapes.*

\*/

public class CustomPanel extends JPanel {

@Override

protected void paintComponent(Graphics g) {

//The superclass does some important work in the method we've overridden, so we

//should invoke it.

super.paintComponent(g);

g.setColor(Color.*blue*);

//Here's an example of a shape.

//For our panels, the origin (0,0) is the top left corner

//This means that the +ve Y axis is "down", i.e. it's the opposite

//of the familiar cartesian coordinate system.

g.drawRect(5,20, 40,60);

//Exercises:

//1: Draw an oval that fits exactly within the rectangle above

//2: Draw an oval that fills the entire panel. NOTE: the panel

// class has getters for its width and height.

//3: Draw a circle of radius 25 that is centered in the center of

// the panel. Make the window bigger/smaller and

// verify that it remains centered.

//4: See if you can find (using intellisense) the method required to

// draw a filled version of the circle from exercise 3.

g.setColor(Color.*red*);

int circleRadius = 25; g.drawOval(this.getWidth()/2 - circleRadius,

this.getHeight()/2 - circleRadius, circleRadius \* 2, circleRadius\*2);

}

}

*/\*\**

*Collected constants of general utility.*

*<P>All members of this class are immutable.*

*<P>(This is an example of*

*<a href='*[*http://www.javapractices.com/Topic2.cjp'*](http://www.javapractices.com/Topic2.cjp%27)*>class for constants</a>.)*

*\*/*

public final class Consts {

public static final int *FRAME\_WIDTH* = 640; public static final int *FRAME\_HEIGHT* = 480;

}