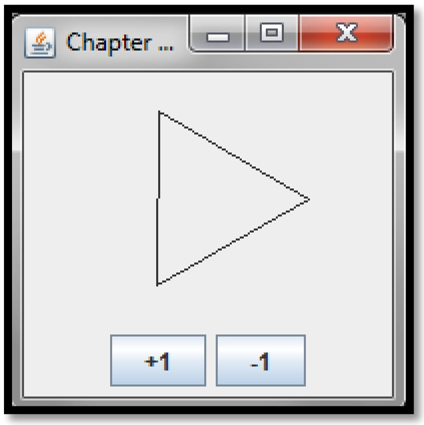
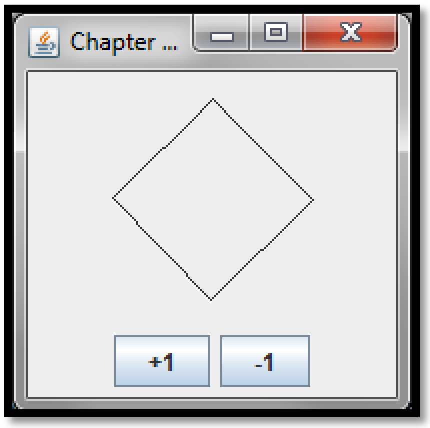
CIT 249: Java II
Chapter 16 Lab 1

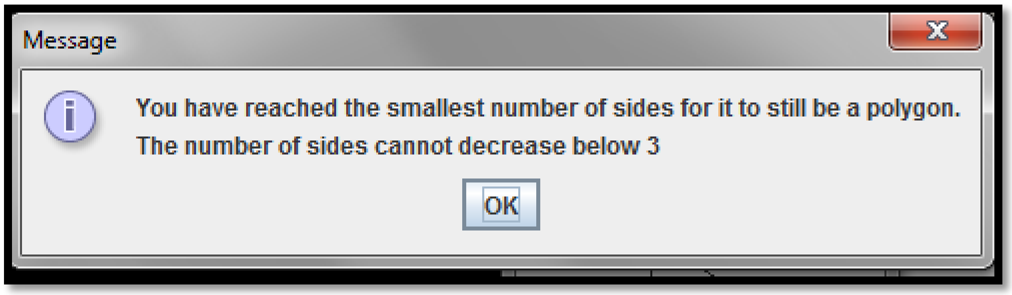
In this program we will complete #31 under Programming exercises on page 635. When the program is run it will display as:



If the +1 is pressed the number of sides of the Polygon increase by one each time it is pressed:



If the -1 is pressed the number of sides of the Polygon will decrease by one, as long as the number of sides is not less than 3 since if it is less than 3 it is no longer a polygon. If you attempt to decrease the number of sides to less than 3 a popup box will display with the message of:



```
You have reached the smallest number of sides for it to still be a polygon.
The number of sides cannot decrease below 3
```

1. Open a new document and save the file as Ch16Lab1.java. Type the following comments that tell the purpose of the program.

```
/*  Purpose:  Draw a Polygon with three sides.  Increase the number of sides when +1 is
*   pressed.  Decrease the number of sides when -1 is pressed except when it will decrease
*   the number of sides to less than 3.
*/
```

2. First we will need to import all classes required to display our different components. Type:

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JOptionPane;
import javax.swing.JButton;
import java.awt.Polygon;
import java.awt.BorderLayout;
import java.awt.event.*;
import java.awt.Graphics;
import java.awt.Dimension;
```

- Notice that I am using the * to import all classes in a package less often? If I need no more than 3-4 classes in a package I simply import the individual classes. If I need to import more than this I use the asterisk.

3. Type the class header and opening brace having the class extend the JFrame class. To extend means to inherit. Type:

```
public class Ch16Lab1 extends JFrame
{
```

- We need to construct two buttons and a RegularPolygonPanel object. Type:

```
//create our components
private JButton jbtEnlarge = new JButton("+1");
private JButton jbtShrink = new JButton("-1");
private RegularPolygonPanel canvas = new RegularPolygonPanel();
```

- The JButton class constructs a button, here with a label of +1 and another with a label of -1
- The RegularPolygonPanel class will follow this class. Its purpose will be to draw the actual polygon.

4. Type the constructor method of:

```
public Ch16Lab1()
{
```

- Remember that the constructor method always has the same name as the class.

5. Within the constructor method we create a panel and add the two buttons to the panel. Type:

```
JPanel panel = new JPanel(); // Use the panel to group buttons
```

```
panel.add(jbtEnlarge);
panel.add(jbtShrink);
```

6. We add the RegularPolygonPanel object to the frame, as well as add the panel. Type:

```
add(canvas, BorderLayout.CENTER); // Add canvas to center
add(panel, BorderLayout.SOUTH); // Add buttons to the frame
```

7. Our next step is to add a listener to the buttons and specify exactly what should occur when the buttons are pressed. This will be done entirely different than we did in Java I. Type:

```
jbtEnlarge.addActionListener(new ActionListener()
{

     public void actionPerformed(ActionEvent e)
    {
        canvas.setNumberOfSides(canvas.getNumberOfSides() + 1);
        canvas.requestFocusInWindow();
    }
});
```

- We add a listener to the jbtEnlarge button. In Java I we simply had addActionListener(this). When doing so we implemented the ActionListener. Here we do not implement the ActionListener but instead add the actionPerformed() method within the addActionListener(). This will be done for each button.
- We invoke the setNumberOfSides() method in the RegularPolygonPanel class and increase the number of sides by 1 by getting the value from the getNumberOfSides() method and adding 1.
- The requestFocusInWindow() makes sure that this button gets the primary focus the moment the window is activated.

8. The other button is program the exact same way. Type:

```
jbtShrink.addActionListener(new ActionListener()
{

  public void actionPerformed(ActionEvent e)
  {
    canvas.setNumberOfSides(canvas.getNumberOfSides() - 1);

    if(canvas.getNumberOfSides() <= 3)
    {
                JOptionPane.showMessageDialog(null, "You have reached the smallest
number of sides for it to still be a polygon.\nThe number of sides cannot decrease below 3");
                canvas.setNumberOfSides(3);
         }
    canvas.requestFocusInWindow();
  }
 });
```

- The one difference here is the message that will display if you attempt to decrease the number of sides when the sides are already set at 3.

9. We also add a mouse listener so that if you press the left mouse while it hovers over the campus it will increase the number of sides. If you press the right mouse button the number of sides will decrease. Type:

```
canvas.addMouseListener(new MouseAdapter()
{
  public void mouseClicked(MouseEvent e)
  {
    if (e.getButton() == MouseEvent.BUTTON1)
      canvas.setNumberOfSides(canvas.getNumberOfSides() + 1);
    else if (e.getButton() == MouseEvent.BUTTON3)
      canvas.setNumberOfSides(canvas.getNumberOfSides() - 1);
  }
 });
```

- Notice that the listener is applied to the RegularPolygonPanel object.

10. We next place focus on the canvas. Type:

```
canvas.setFocusable(true);
 canvas.requestFocusInWindow();
```

11. We also add a listener to the canvas for when either the UP and Down arrow buttons are pressed. Type:

```
canvas.addKeyListener(new KeyAdapter()
 {
  public void keyPressed(KeyEvent e)
  {
    if (e.getKeyCode() == KeyEvent.VK_UP)
      canvas.setNumberOfSides(canvas.getNumberOfSides() + 1);
    else if (e.getKeyCode() == KeyEvent.VK_DOWN)
      canvas.setNumberOfSides(canvas.getNumberOfSides() - 1);
    }
  });
}
```

- If the UP arrow is pressed it increases the number of sides.
- If the DOWN arrow is pressed it decreases the number of sides.

12. The main method is written as in past labs. Type:

```
public static void main(String[] args)
{
 JFrame frame = new Ch16Lab1();
 frame.setTitle("Chapter 16 Lab 1");
 frame.setLocationRelativeTo(null); // Center the frame
```

```
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  frame.setSize(200, 200);
  frame.setVisible(true);
}
```

13. Close the class.
14. The RegularPolygonPanel class will be responsible for creating the polygon and painting it onto the screen. Type the class header and opening brace, omitting the access modifier of "public", having the class extend the JPanel class.
15. Construct a private integer named numberOfSides set to 3.
16. We will create the constructor method and overload it once. Type:

```
 public RegularPolygonPanel()
{
}

public RegularPolygonPanel(int numberOfSides)
{
  setNumberOfSides(numberOfSides);
}
```

- Notice that the overloaded method passes the number of sides to the setNumberOfSides() method.

17. The accessor method is simple to understand. Type:

```
 public int getNumberOfSides()
{
  return numberOfSides;
}
```

18. The mutator method is a little different. Type:

```
 public void setNumberOfSides(int numberOfSides)
{
  this.numberOfSides = numberOfSides < 3 ? 3 : numberOfSides;
  repaint();
}
```

- The ? is a ternary operator. This operator is used instead of if, else if statements. First < 3 is tested. If is true than 3 is returned. If the statement is not true then the integer passed to the method is returned. This would be the same as writing:

```
if(numberOfSides < 3)
    this.numberOfSides = 3;
else
    this.numberOfSides = numberOfSides;
```

- We then repaint the screen with the new number of sides.

19. Next is our paintComponent() method. Type:

```
 protected void paintComponent(Graphics g)
{
  super.paintComponent(g);

  int xCenter = getWidth() / 2;
  int yCenter = getHeight() / 2;
  int radius = (int) (Math.min(getWidth(), getHeight()) * 0.4);

  double angle = 2 * Math.PI / numberOfSides;

  // Create a Polygon object
  Polygon polygon = new Polygon();

  // Add points to the polygon
  for (int i = 0; i < numberOfSides; i++)
  {
    polygon.addPoint((int)(xCenter + radius * Math.cos(i * angle)),
        (int)(yCenter - radius * Math.sin(i * angle)));
  }

  // Draw the polygon
  g.drawPolygon(polygon);
}
```

- We invoke the superclass' paintComponent() method.
- We create 2 integers used to center the polygon.
- We create a integer for the radius of the polygon.
- We create a Polygon object. This is a predefined class.
- Using a for loop we set the number of sides, centering it and determine the radius.
- We draw the polygon.

20. Our last method is used to set the preferred size. Type:

```
 public Dimension getPreferredSize()
{
  return new Dimension(200, 200);
}
```

21. Close the class.
22. Compile the program and fix any errors.
23. Run the program and test it.
24. Compress ALL files, you will have a class file for the RegularPolygonPanel class.
25. Submit to the drop box for Chapter 16 Lab 1.