```java
/**
 * @author Chris Troutner
 *
 * Created on Jul 25, 2005
 *
 * Project: Project #3
 *
 */
import java.awt.event.*;
import java.awt.*;
import java.util.Random;

public class proj3 implements ActionListener, MouseListener
{

        // classes can implement multiple interfaces. A KidsGame object is both
        //   an ActionListener AND a MouseListener. Can you look those interfaces up on the Java API
        //   and determine what methods this class MUST include?
        private MyFrame gui;
        private String player;

        public proj3()
        {
            gui = new MyFrame(this);
            gui.setVisible(true);
        }

        public static void main (String[] args)
        {
            new proj3();
        }

        // Here is the method you must include if you are an ActionListener
        public void actionPerformed( ActionEvent ae )
        {
            // This method is automatically called by Java when the button is clicked because the
            //   button over in the gui was told that an object from this class would be its actionListener.
            roundOne();
        }

        public void roundOne()
        {
            // Get the first name that the user typed in
            player = gui.getName();

            // Instantiate lots of MyCircle objects and tell the gui to add them to its panel
            // The addToPanel is a method I put in the MyFrame class.
```

```
            MyCircle cir1 = new MyCircle(Color.red, 60, 100,10);
            gui.addToPanel(cir1);
            MyCircle cir2 = new MyCircle( Color.blue, 30, 10, 100);
            gui.addToPanel(cir2);
            MyCircle cir3 = new MyCircle( Color.green, 70, 120, 200);
            gui.addToPanel(cir3);
            MyCircle cir4 = new MyCircle( Color.yellow, 50, 80, 170);
            gui.addToPanel(cir4);
            MyCircle cir5 = new MyCircle( Color.orange, 80, 220, 250);
            gui.addToPanel(cir5);
            MyCircle cir6 = new MyCircle( Color.magenta, 60, 250, 150);
            gui.addToPanel(cir6);
            MyCircle cir7 = new MyCircle( Color.pink, 40, 270, 30);
            gui.addToPanel(cir7);
            MyCircle cir8 = new MyCircle( Color.cyan, 50, 350, 170);
            gui.addToPanel(cir8);
            gui.setDirections(player + ", touch the yellow circle with the mouse.");

            // Assign a mouse listener to every MyCircle object
            cir1.addMouseListener(this);
            cir2.addMouseListener(this);
            cir3.addMouseListener(this);
            // Create an instance of the inner class defined below and use that obj as your listener.
            cir4.addMouseListener( new MyRoundOneListener() );
            cir5.addMouseListener(this);
            cir6.addMouseListener(this);
            cir7.addMouseListener(this);
            cir8.addMouseListener(this);
        }

        public void roundTwo()
        {
            //These instructions were supplied on the class webpage:
            gui.removeComponents();
            gui.setDirections(player + ", touch the square with the mouse.");


            //Note: The frame dimenions are 460 X 400
            //MyPolygon(Color c, int width, int height, int Locx, int Locy, int[] xpoints, int[] ypoints, int npoints)
            MyPolygon poly1 = new MyPolygon(Color.red, 40, 150, 70, 230, new int[] {0,40,40,0}, new int[]
{0,0,150,150}, 4);
            gui.addToPanel(poly1);
            MyPolygon poly2 = new MyPolygon(Color.red, 70, 80, 30, 75, new int[] {35,70,50,20,0}, new int[]
{0,30,80,80,30}, 5);
            gui.addToPanel(poly2);
            MyPolygon poly3 = new MyPolygon(Color.red, 60, 30, 175, 30, new int[] {0,60,60,0}, new int[] {0,0,30,30},
4);
```

```
            gui.addToPanel(poly3);
            MyPolygon poly4 = new MyPolygon(Color.red, 50, 100, 350, 250, new int[] {25,50,0}, new int[] {0,100,100},
3);
            gui.addToPanel(poly4);
            MyPolygon poly5 = new MyPolygon(Color.red, 120, 160, 190, 230, new int[] {0,120,120}, new int[] {0,0,160},
3);
            gui.addToPanel(poly5);
            MyPolygon poly6 = new MyPolygon(Color.red, 40, 40, 350, 100, new int[] {0,40,40,0}, new int[] {0,0,40,40},
4);
            gui.addToPanel(poly6);
            MyCircle cir1 = new MyCircle( Color.red, 70, 200, 120);
            gui.addToPanel(cir1);


            // Assign a mouse listener to every object
            poly1.addMouseListener(this);
            poly2.addMouseListener(this);
            poly3.addMouseListener(this);
            poly4.addMouseListener(this);
            poly5.addMouseListener(this);
            cir1.addMouseListener(this);

            //For the square, I'll assign a separte listener class
            poly6.addMouseListener( new MyRoundTwoListener() );
        }

        public void roundThree()
        {
            //Local Variables
            int posx, posy, panelx, panely;
            Random generator = new Random();
            final int circdiameter = 60;

            //Initialize the screen (same as for round 2)
            gui.removeComponents();
            gui.setDirections(player + ", touch the circles with the mouse.");

            //Retrieve the width and height of the panel
            panelx = gui.myPanel.getWidth();
            panely = gui.myPanel.getHeight();

            //Generate the three colored circles:
            //The width of the shape is subtracted from the panel width so that
            //the object stays within the visible area.
            posx = generator.nextInt(panelx - circdiameter);
            posy = generator.nextInt(panely - circdiameter);
            MyCircle cir1 = new MyCircle(Color.green, circdiameter, posx, posy);
```

```
        gui.addToPanel(cir1);

        posx = generator.nextInt(panelx - circdiameter);
        posy = generator.nextInt(panely - circdiameter);
        MyCircle cir2 = new MyCircle(Color.blue, circdiameter, posx, posy);
        gui.addToPanel(cir2);

        posx = generator.nextInt(panelx - circdiameter);
        posy = generator.nextInt(panely - circdiameter);
        MyCircle cir3 = new MyCircle(Color.yellow, circdiameter, posx, posy);
        gui.addToPanel(cir3);

        //Assign a new listener class to each circle.
        cir1.addMouseListener(new MyRoundThreeListener());
        cir2.addMouseListener(new MyRoundThreeListener());
        cir3.addMouseListener(new MyRoundThreeListener());

    }

    // Here are the methods you must include if you are a MouseListener
    public void mouseEntered( MouseEvent me )
    {
        Toolkit.getDefaultToolkit().beep();
    }
    public void mouseExited( MouseEvent me ) {}
    public void mouseClicked( MouseEvent me ) {}
    public void mousePressed( MouseEvent me ) {}
    public void mouseReleased( MouseEvent me ) {}


    // This is an inner class, a class definition within another class definition
    // Inner classes are often used as listeners. This class inherits from MouseAdaptor.
    // The MouseAdaptor class implements MouseListener and includes all 5 methods
    // that MouseListeners must have. But its method bodies are empty. You just need
    // to override the one you want to use.
    class MyRoundOneListener extends MouseAdapter
    {
        public void mouseEntered( MouseEvent me )
        {
            roundTwo();
        }
    }

    //This class was just copied and modified from the one above.
    class MyRoundTwoListener extends MouseAdapter
    {
        public void mouseEntered( MouseEvent me )
```

```
            {
                roundThree();
            }
        }

        //This mouse listener class generates the new random corrdinates
        //and moves the circles around the panel to these new locations
        //when the mouse passes over the circle.
        //Note: Variable names are chosen from the instructions given on the
        //class web page/project directions.
        class MyRoundThreeListener extends MouseAdapter
        {
            public void mouseEntered( MouseEvent me )
            {
                //Local Variables
                Component shape = (Component) me.getSource();       //shape is the object that called the mouse event.
                int panelx, panely, shapex, shapey;
                Random generator = new Random();

                //This block generates a new random corrdinate for the shape.
                //The width of the shape is subtracted from the panel width so that
                //the object stays within the visible area.
                panelx = shape.getParent().getWidth();  //Retrieve the width of the panel
                panely = shape.getParent().getHeight(); //Retrieve the hight of the panel
                shapex = generator.nextInt(panelx - shape.getWidth());  //Generate a random horizontal coordinate.
                shapey = generator.nextInt(panely - shape.getHeight()); //Generate a random verticle coordinate.

                //Move the object to the new location
                shape.setLocation(shapex, shapey);

            }
        }
}
```

```java
/**
 * @author Chris Troutner
 *
 * Created on Jul 25, 2005
 *
 * Project: Project #3
 *
 * Note: The code and comments downloaded from the class web page have
 * not been changed.
 */

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


public class MyFrame extends JFrame
{
    // To be able to see your components, you have to do 5 things. Memorize these.
    // #1 you have to instantiate the component
    JButton button1 = new JButton();

    JTextField nameTF = new JTextField();
    JLabel directions = new JLabel("Tell me your first name.");
    // JPanel objects are used as containers that you put other components in to.
    JPanel myPanel = new JPanel();

    public MyFrame(proj3 kg)
    {
        // #2 you have to set the component's location
        button1.setLocation(20, 520);
        // #3 you have to set the component's size
        button1.setSize(90, 40);
        button1.setText("Enter");
        // #4 you have to set the component's visibility
        button1.setVisible(true);

        // button1 needs a listener; someone to tell when it has been clicked.
        // I am using the KidsGame object as the listener.
        // That means that the KidsGame class MUST implement the interface ActionListener
        button1.addActionListener(kg);

        nameTF.setLocation(20, 480);
        nameTF.setSize(220, 30);
        nameTF.setVisible(true);

        directions.setFont( new Font("Arial", Font.BOLD, 18) );
        directions.setForeground( Color.BLUE );
```

```
    directions.setLocation(20, 445);
    directions.setSize(500, 30);
    directions.setVisible(true);

    myPanel.setBackground(Color.white);
    myPanel.setLocation(15, 15);
    myPanel.setSize(460,400);
    myPanel.setVisible(true);
    myPanel.setLayout(null);

    // #5 you have to add the component to a container.
    //    the object returned by getContentPane() is the container of a JFrame obj.
    this.getContentPane().add(button1);
    this.getContentPane().add(nameTF);
    this.getContentPane().add(directions);
    this.getContentPane().add(myPanel);


    // This line tells the container to not use a layout manager
    // This means that we have to set the size and location of the components ourselves
    this.getContentPane().setLayout(null);
    // This puts the top left corner of our MyFirstFrame object at (0,0)
    //       that is the top left corner of our screen
    this.setLocation(0, 0);
    this.setTitle("MyFirstFrame");
    // This sets the size of our MyFirstFrame object.
    this.setSize(500, 650);

    // event handling
    // This is an advanced topic--an anonymous inner class
    //        not something you need to worry about right now
    this.addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(java.awt.event.WindowEvent e) {
            setVisible(false);
            dispose();
            System.exit(0);
        }
    });

}

public void addToPanel( JComponent myComponent ) {
    // myPanel is a container that holds all the components you are creating
    // This method tells myPanel to add your component and then it tells
    //  your component to repaint itself.
    myPanel.add( myComponent );
    myComponent.repaint();
```

```
    }

    public String getName() {
        // Return the name that the user typed in and also make the JTextField and JButton invisible
        nameTF.setVisible( false );
        button1.setVisible( false );
        return nameTF.getText();
    }

    public void setDirections( String str ) {
        // Tell the JLabel object to change its text
        directions.setText( str );
    }

    public void removeComponents() {
        // The getComponents method returns an array of all the components currently in this container
        Component[] array = myPanel.getComponents();
        for( int i=0; i<array.length; i++ )
            myPanel.remove( array[i] );

        // repaint the empty panel
        myPanel.repaint();
    }

}
```

```java
/**
 * @author Chris Troutner
 *
 * Created on Jul 25, 2005
 *
 * Project: Project #3
 *
 * This is the code to generate the circle objects
 */

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

public class MyCircle extends JComponent
{

    public MyCircle(Color c, int s, int x, int y)
    {
        this.setSize(s, s);
        this.setLocation(x,y);
        this.setVisible(true);
        this.setForeground(c);
    }

    public void paintComponent( Graphics g )
    {
        g.fillOval( 0,0,getWidth(), getHeight() );
    }

}
```

```java
/**
 * @author Chris Troutner
 *
 * Created on Jul 25, 2005
 *
 * Project: Project #3
 *
 * Note: This code is copied and modified from the MyCircle.java
 * class. Instead of drawing a circle, it draws a polygon. A
 * polygon can be a rectangle, square, or any n-pointed object.
 */

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


public class MyPolygon extends JComponent
{
    private int[] Xpoints;
    private int[] Ypoints;
    private int Npoints;

    public MyPolygon(Color c, int width, int height, int Locx, int Locy, int[] xpoints, int[] ypoints, int npoints)
    {
        this.setSize(width, height);
        this.setLocation(Locx,Locy);
        this.setVisible(true);
        this.setForeground(c);

        Xpoints = xpoints;
        Ypoints = ypoints;
        Npoints = npoints;
    }

    public void paintComponent( Graphics g )
    {
        g.fillPolygon( Xpoints, Ypoints, Npoints );
    }

}
```