```java
/*
 * Filename:         proj4.java
 * Included Files:   MyPanel.java
 *                   MyFrame.java
 *                   MyPoint.java
 *
 * Author:           Chris Troutner
 * Date Created:     8/4/05
 *
 * Notes:
 * ------
 * This program is Project #4 for my CS161 course taken
 * summer term '05. It creates a GUI takes coordinate inputs
 * and plots the points on a grid.
 */

//Imported files
import java.awt.event.*;

public class proj4 implements ActionListener, MouseListener
{
    private MyFrame gui;    //gui is the graphical user interface

    //Main - this just calls itself
    public static void main(String[] args)
    {
        new proj4();
    }

    public proj4()
    {
        //Create a new gui (MyFrame)
        gui = new MyFrame(this);
        gui.setVisible(true);
    }

    // Here is the method you must include if you are an ActionListener
    // This method handles the action when the button is pressed.
    public void actionPerformed( ActionEvent ae )
    {
        int x = 0;
        int y = 0;

        try
        {
            x = Integer.parseInt(gui.xcoordTF.getText());
            y = Integer.parseInt(gui.ycoordTF.getText());
```

```
            if(x < 0 || x > 100 || y < 0 || y > 100)
                throw new ArithmeticException();
            MyPoint p = new MyPoint(x, y);
            p.setSize(10, 10);
            p.setVisible(true);
            gui.myPanel.addMyPoint(p, x, y);
            gui.xcoordTF.setText("");
            gui.ycoordTF.setText("");
            gui.xcoordTF.requestFocus();
            gui.errLabel.setVisible(false);
        }

        catch(NumberFormatException _ex)
        {
            gui.errLabel.setVisible(true);
            gui.errLabel.setText("You must enter two integers.");
        }

        catch(ArithmeticException _ex)
        {
            gui.errLabel.setVisible(true);
            gui.errLabel.setText("Your two integers must be in the range [0, 100]");
        }

    }

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

```java
/*
 * Notes:
 * ------
 * This part of the program was orgionally give to the class by the teacher
 * - with a lot of holes left to fill with code.
 */

//Imported Files
import java.awt.*;
import javax.swing.*;
import java.util.*;
import java.awt.event.*;

public class MyPanel extends JPanel {
    private int MARGIN = 35;
    private int yInc;
    private int xInc;
    private ArrayList aList = new ArrayList();

    //This method and the paintComponent method below create the GUI grid
    //on a JPanel object.
    public MyPanel( int width, int height )
    {
        // Since the constructor uses setSize, you don't have to do this again in your gui
        setSize(width, height);
        yInc = (height -  MARGIN - 10) / 10;
        xInc = (width -  MARGIN - 10) / 10;
        setLayout(null); // This tells Java that this container will not use a Layout Manager.
    }

    public void paintComponent( Graphics g )
    {
        //LOCAL VARIABLES
        FontMetrics fontM = g.getFontMetrics(); // fontM is an object that can tell us
        final int HALFTIC = 5;                  // the width and height of a String in pixels.
        int strHt = 0, strWth = 0;
        super.paintComponent( g );
        // draw vertical axis
        g.drawLine(  MARGIN, 10,  MARGIN, getHeight() - 5 );
        // draw horizontal axis
        g.drawLine( 10, getHeight() -  MARGIN, getWidth()-15, getHeight() -  MARGIN );
        MyPoint point1;
        MyPoint point2;

        //MAIN EXECUTION CODE
        for( int i=1; i<=10; i++ ) {
            // draw and label tic marks on horizontal axis.
```

```
            g.drawLine(  MARGIN + i*xInc, getHeight()- MARGIN-HALFTIC,  MARGIN + i*xInc, getHeight() - MARGIN+HALFTIC
);
            strHt = fontM.getHeight();
            strWth = fontM.stringWidth( "" + i*10 );
            g.drawString(""+i*10, MARGIN + i*xInc - strWth/2, getHeight() - MARGIN + strHt );
        }

        for( int i=1; i<=10; i++ ) {
            g.drawLine( MARGIN-HALFTIC, getHeight()- MARGIN - i*yInc, MARGIN+HALFTIC, getHeight()- MARGIN - i*yInc );
            strHt = fontM.getAscent();
            strWth = fontM.stringWidth( "" + i*10 );
            g.drawString(""+i*10, MARGIN - strWth - HALFTIC, getHeight() - MARGIN + strHt/2 - i*yInc );
        }

        // draw lines connecting points
        for( int i=1; i<aList.size(); i++ ) //Loop through the entire arrayList.
        {
            //This code retireves the coordinates of two points and draws a line between them.
            point2 = (MyPoint)aList.get(i);     //Retrieve point2
            point1 = (MyPoint)aList.get(i - 1); //Retrieve point 1
            g.drawLine(point1.getWidth()/2 + point1.getX(), point1.getHeight()/2 + point1.getY(), point2.getWidth()/2 +
point2.getX(), point2.getHeight()/2 + point2.getY());
        }
    }

    public void addMyPoint(MyPoint p, int x, int y)
    {
        /*
           This method accepts a MyPoint object and the coordinates from the user.
           It converts the ints sent by the user to those needed by the setLocation method.
           You will add code that:
             1. adds p to this MyPanel object
             2. registers the correct kind of listener to p so that it can tell someone when the user clicks on it.
                 Obviously, you will be writing an inner class to serve as this listener.
             3. repaints this MyPanel object. (This will cause the paintComponent method above to be run.)
                 Don't call the paintComponent method yourself. Java will do it for you when you call repaint.
             4. adds p to the ArrayList object named aList. (See the instance field at the top of the class.)
                 You will have to add p to the correct index in aList. Don't just add it to the end everytime.
                 You should look at the x coordinate of p and insert it into aList immediately before the first
                 MyPoint object that has a bigger x coordinate. If none do, then insert it at the end.
        */

        //LOCAL VARIABLES
        boolean addFlag = false;

        //EXECUTION BODY
        x = ((x*xInc)/10) - p.getWidth()/2 + MARGIN;
```

```
        y = getHeight() - (y*yInc)/10 - p.getHeight()/2 - MARGIN;

        //Modify p
        p.setLocation(x, y);

        //p's mouse listener
        p.addMouseListener(new MouseAdapter()
        {

            public void mouseClicked(MouseEvent me)
            {
                //Remove a point from the list when it get's clicked
                aList.remove(aList.indexOf(me.getSource())); //remove from aList
                remove((MyPoint)me.getSource());    //Remove from the GUI

                repaint(); //Repaint the panel.
            }

        }); //end of the embedded class.

        //These two instructions add apoint to the panel.
        add(p);     //add the point.
        repaint();  //Repaint the panel.

        //Add a new point into aList at the right location.
        for(int i = 0; i < aList.size(); i++)   //Loop through the list until we reach the
        {                                       //right spot.
            //if the x-coordinate of the point in aList(i) > our new point...
            if(p.getXPnt() >= ((MyPoint)aList.get(i)).getXPnt())    //then continue.
                continue;
            //Once we find the point where we should insert our new point...
            aList.add(i, p);    //add it in.
            addFlag = true;     //Set our flag.
            break;              //Break out of the loop.
        }

        //If we looped through the whole list and didn't find any points with a...
        if(!addFlag)        //x-coord bigger than our current point,
            aList.add(p);   //then add the new point at the end of the list.
    }

}
```

```java
/**
 * @author Chris Troutner
 *
 * Created on Jul 25, 2005
 *
 * Project: Project #4
 *
 * Most of the code below is taken directly from project #3.
 */

//Imported Files
import java.awt.*;
import javax.swing.*;

//This is where I construct the graphical user interface
public class MyFrame extends JFrame
{
    /*
     * Note:
     * -----
     * To be able to see your components, you have to do 5 things. Memorize these:
     * #1 you have to instantiate the component
     * #2 you have to set the component's location
     * #3 you have to set the component's size
     * #4 you have to set the component's visibility
     * #5 you have to add the component to a container.
     */

    //Local Object Variables (instantiations)
    JButton     button1     = new JButton();
    JTextField  xcoordTF    = new JTextField();
    JTextField  ycoordTF    = new JTextField();
    JLabel      xlabel      = new JLabel("X Coordinate");
    JLabel      ylabel      = new JLabel("Y Coordinate");
    JLabel      errLabel    = new JLabel("");
    // JPanel objects are used as containers that you put other components in to.
    // MyPanel is a separate class that the teacher created to generate the GUI grid.
    MyPanel     myPanel     = new MyPanel(480, 450);

    public MyFrame(proj4 caller)
    {
        //INITIALIZE GUI OBJECTS
        // button1 needs a listener - someone to tell when it has been clicked.
        // I am using the calling object as the listener.
        // That means that the calling object's class MUST implement the interface ActionListener
        button1.addActionListener(caller);
        button1.setLocation(200, 530);
```

```
button1.setSize(100, 30);
button1.setText("Add Point");
button1.setVisible(true);

xcoordTF.setLocation(110, 480);
xcoordTF.setSize(100, 30);
xcoordTF.setVisible(true);
xcoordTF.addActionListener(caller);

ycoordTF.setLocation(340, 480);
ycoordTF.setSize(100, 30);
ycoordTF.setVisible(true);
ycoordTF.addActionListener(caller);

//xlabel.setFont( new Font("Arial", Font.BOLD, 18) );
//directions.setForeground( Color.BLUE );
xlabel.setSize(100, 30);
xlabel.setLocation(30, 480);
xlabel.setVisible(true);

ylabel.setSize(100, 30);
ylabel.setLocation(260, 480);
ylabel.setVisible(true);

//All the other settings for myPanel are taken care of in the
//MyPanel class.
myPanel.setBackground(Color.white);
myPanel.setLocation(30, 15);

errLabel.setSize(300, 30);
errLabel.setLocation(100, 560);
errLabel.setVisible(false);
errLabel.setForeground(Color.red);

//Finally I can add all the GUI objects to my frame.
//The object returned by getContentPane() is the container of a JFrame obj.
this.getContentPane().add(button1);
this.getContentPane().add(xcoordTF);
this.getContentPane().add(ycoordTF);
this.getContentPane().add(xlabel);
this.getContentPane().add(ylabel);
this.getContentPane().add(myPanel);
this.getContentPane().add(errLabel);


// 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("Project 4 Plot Program");
        // This sets the size of our MyFirstFrame object.
        this.setSize(550, 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
        xcoordTF.setVisible( false );
        button1.setVisible( false );
        return xcoordTF.getText();
    }

    public void setDirections( String str ) {
        // Tell the JLabel object to change its text
        xlabel.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 8/4/05
 *
 * Project: Project #4
 *
 * Notes: This file creates the points on the graph.
 */

import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JComponent;

public class MyPoint extends JComponent
{
    //LOCAL VARIABLES
    private int xpnt;
    private int ypnt;

    public class PointListener extends MouseAdapter
    {

        public void mouseEntered(MouseEvent me)
        {
            setToolTipText("(" + xpnt + ", " + ypnt + ")");
        }

        public PointListener()
        {
        }
    }


    public MyPoint(int x, int y)
    {
        addMouseListener(new PointListener());
        xpnt = x;
        ypnt = y;
    }

    public int getXPnt()    //This method just returns the x coordinate point.
    {
        return xpnt;
    }
```

```
    public void paintComponent(Graphics g) //Paint the point.
    {
        g.setColor(Color.blue); //Make the points blue.
        g.fillOval(0, 0, getWidth(), getHeight());  //call fillOval since we want a circle.
    }



}
```