**Lab #3: JFrames, Graphics**

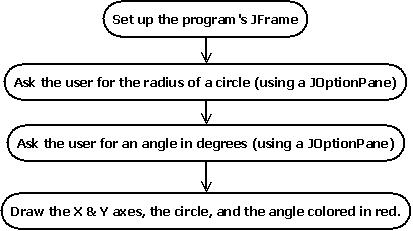
**Objectives**

* JFrames, graphics.

**Problem Description**

In this lab, we will write an application that accomplishes the items in Fig 1.

Fig 1:



**Part 1 of 4: Setting up the JFrame**

First, create a new project called IntroJFrameLab.

The file “IntroJFrameLab.java” should be open in the NetBeans editor. If it isn’t, find it in Source Packages->introjframelab as shown.



The file should have this code in it. Do the steps listed to this code to set up the JFrame.

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package introjframelab;

/\*\*

\*

\* @author \*\*Your e-Id here

\*/

public class IntroJFrameLab {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

}

}

Change this to your name, assignment, etc.

Erase this 4-line comment.

Erase this 3-line comment also.

Erase this comment also.

You should wind up with this:

Now add the red highlighted code to your file.

/\*

\* Your Name

\* Lab 3

\* Date

\*/

package introjframelab;

public class IntroJFrameLab {

public static void main(String[] args) {

}

}

Now, run the program and you should see a window like this:

/\*

\* Your Name

\* Lab 3

\* Date

\*/

package introjframelab;

import java.awt.Color;

import javax.swing.JFrame;

import java.awt.Graphics;

import javax.swing.JOptionPane;

public class IntroJFrameLab extends JFrame{

private static final int FRAME\_SIZE = 400;

private int radius, degrees;

public static void main(String[] args) {

IntroJFrameLab guiWindow = new IntroJFrameLab ();

guiWindow.setSize(FRAME\_SIZE, FRAME\_SIZE);

guiWindow.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

guiWindow.setVisible(true);

}

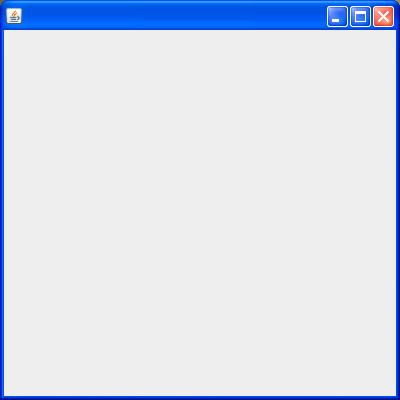
}

These lines add required imports.

This partial line causes our class to be usable as a JFrame. It lets us make a window for our program.

These lines define our program window’s size and two variables for our program to use.

These lines inside main() declare a window frame called guiWindow and set the size of the frame, the close operation, and then set the window to be visible.



This is what the ‘guiWindow’ frame looks like when the program is run. We will add some drawings later.

**Part 2 of 4: Asking the user for the radius of a circle using a JOptionPane**

Now that we have set up the JFrame part of the program, let’s ask the user for the radius of a circle.

To do this, we are going to use a JOptionPane. JOptionPanes are pop-up windows (called dialog boxes) that ask the user to enter some data.

Go to your main() method and add the red lines of code.

public static void main(String[] args) {

IntroJFrameLab guiWindow = new IntroJFrameLab ();

guiWindow.setSize(FRAME\_SIZE, FRAME\_SIZE);

guiWindow.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

String valueString;

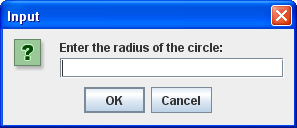
valueString = JOptionPane.showInputDialog("Enter the radius of the circle?");

guiWindow.radius = Integer.parseInt(valueString);

guiWindow.setVisible(true);

}

These lines will cause the following dialog box to appear:



Then the program will show the blank window we saw at the end of Part 1. For this lab, we will assume that the user will enter a valid number and not worry about bad inputs.

We will use the radius in Part 4.

**Part 3 of 4: Asking the user for an angle in degrees using a JOptionPane**

Now we will use another JOptionPane to get an angle in degrees from the user.

Again, add the red code shown below.

public static void main(String[] args) {

IntroJFrameLab guiWindow = new IntroJFrameLab ();

guiWindow.setSize(FRAME\_SIZE, FRAME\_SIZE);

guiWindow.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

String valueString;

valueString = JOptionPane.showInputDialog("Enter the radius of the circle?");

guiWindow.radius = Integer.parseInt(valueString);

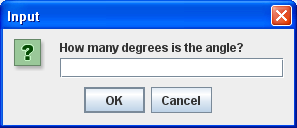
valueString = JOptionPane.showInputDialog("How many degrees is the angle?");

guiWindow.degrees = Integer.parseInt(valueString);

guiWindow.setVisible(true);

}

These lines will cause this dialog box to appear right after the first dialog box from Part 2:

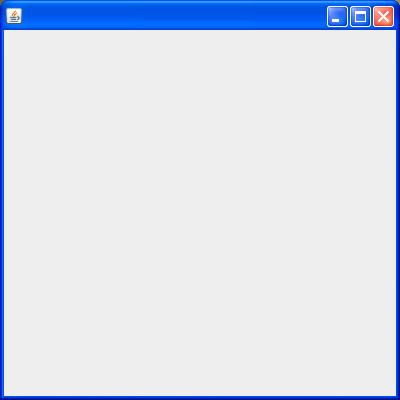


Once again, the program will simply show the blank window after the user enters a number of degrees. And once again, we will not worry about the possibility that the user will enter bad data, we will assume a number between 1 and 360 is given.

**Part 4 of 4: Drawing the X & Y axes, the circle, and the angle arc**

Now that we have set up the frame and received the radius and the number of degrees from the user, we will draw the circle and the arc.

In order to draw the shapes, we need to set up the frame’s Content Pane. The Content Pane is the part of the frame that we want to draw in.



Border

Content Pane

Title Bar.

Title Bar.

Whenever a frame is drawn on the screen, a special method called “paint” handles the drawing.

We will begin to set up the paint method by entering the following code below the main method.

@Override

public void paint(Graphics g) {

super.paint(g);

Graphics canvas = getContentPane().getGraphics();

}

“g” is the name of the graphics object for the whole frame.

Overrides the Frame’s paint method. We’ll learn this later.

Draws the basic parts of the frame (Title Bar, Border, Background, etc).

This line gets only the content pane’s graphics, so we can draw on the content pane and not on the other parts of the frame.

By drawing on the ‘canvas’ object instead of the object called ‘g’, we can draw only on the content pane and not worry about drawing under the Title Bar.

First let’s set up an x-axis and a y-axis.

@Override

public void paint(Graphics g) {

super.paint(g);

Graphics canvas = getContentPane().getGraphics();

int width = this.getContentPane().getWidth();

int height = this.getContentPane().getHeight();

canvas.drawLine(width/2, 0, width/2, height); // draws y axis.

canvas.drawLine(0, height/2, width, height/2);// draws x axis.

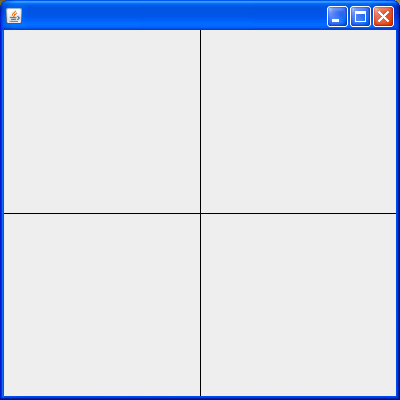
}

Add the red lines of code to the end of the paint method:

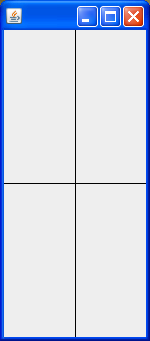
The first two lines get the width and height of the content pane so that we can draw the x and y axes correctly.

The last two lines draw the axes themselves.

Once these lines have been added, the program will look like this at the end:



Notice that if you change the frame’s size, the axes stay the same.



Finally, we will add the circle and the red arc.

@Override

public void paint(Graphics g) {

super.paint(g);

Graphics canvas = getContentPane().getGraphics();

int width = this.getContentPane().getWidth();

int height = this.getContentPane().getHeight();

canvas.drawLine(width/2, 0, width/2, height); // draws y axis.

canvas.drawLine(0, height/2, width, height/2);// draws x axis.

canvas.drawOval(width/2-radius, height/2-radius,

radius \* 2, radius \* 2);

canvas.setColor(Color.RED);

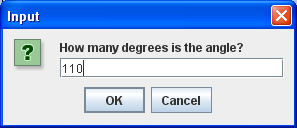
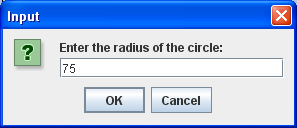
canvas.fillArc(width/2-radius, height/2-radius,

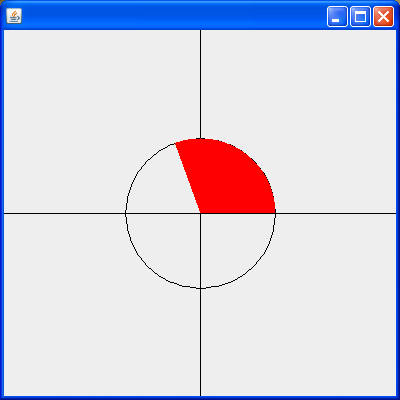
radius \* 2, radius \* 2, 0, degrees);

canvas.setColor(Color.BLACK);

}

Now the program should draw the circle with the given radius, and the red arc with the given degrees like this:





**Epilogue:**

If you want to have some fun drawing even more shapes, visit <http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/Graphics.html> to see a list of all the shapes you can draw.