Java Graphical User Interface guide

Developing GUI's from 0

Java GUI exercises

In this series of exercises I hope cover the topic regardless to the input and output messages so please follow me in this series.

The exercises are took from the java how to program by Deitel and Pearson editorial house

**JOptionPane**, this is the class that we're going to use for the first exercises. This class is a static class which means that we don't need to create an instance of this class, just by the class name we can access to the resources that this class brings us.

Exercise 001 Message box:

```
import javax.swing.JOptionPane;
public class Dialog1 {

	public static void main(String[] args) {
		JOptionPane.showMessageDialog(null,"this is the JOptionPane");
	}

}
```

Exercise 002 I/O:

Get an input from the inputDialog and format it and finally display the data

```
 import javax.swing.JOptionPane;
public class DialogBox002 {
	public static void main(String[] args) {
		String name = null;
		byte charNum=0;
		//asigning the data from the inputdialog to the name variable
		name = JOptionPane.showInputDialog("type youre name:");
		//we can apply some logic in here or just display the value that
		//we just got
		charNum= (byte)name.length();
		name=String.format("welcome %S to the java world\nby the way %s have %d
characters", name,name,charNum);
		JOptionPane.showMessageDialog(null, name);
	}
}
```

Exercise 003 working with the values that we get from the user

This exercise shows how to work and parse data type once the user type something in the input dialog

The program comments tell what's going on any doubt feel free to ask

Here's the code:

```
import javax.swing.JOptionPane;
public class DialogBox003 {

	public static void main(String[] args) {
		String num1=null,num2=null;
		int result=0;
		//just a welcome message
		JOptionPane.showMessageDialog(null, "Welcome to this app\nwere going to
do a simple addition");
		num1=JOptionPane.showInputDialog("Type the firs number:");
		num2=JOptionPane.showInputDialog("Type the seccond number:");
		//in the next line we're changingin the string data type
		//and doing the addition also we're asigning the result to the result
variable
		result=Integer.parseInt(num1)+Integer.parseInt(num2);
		// as you can see in the console the num1 and num2 variables still being
Strings
		System.out.println(num1+num2);
		JOptionPane.showMessageDialog(null, "result: "+result);
		//this is another aproach
		//JOptionPane.showMessageDialog(null, "result:
"+(Integer.parseInt(num1)+Integer.parseInt(num2)));
	}
}
```

For the moment I think its clear how to display messages and how to manipulate de data that we're getting from the user through the inputDialog

The next topic that I'm planning covers it about the Graphic class which allows us to draw figures into a java window

Exercise 004 graphics

Regarding to today's topic just mention that we're going to use two classes from two different packages. And those are:

Java.awt.Graphics:  tool to draw

Javax.swing.JPane: area where we can draw

Javax.swing.JFrame: window frame

The first one has the methods that allow us to draw figures and a bunch of other interesting things.

If you to do any 2D draw it's important that you have to know or be familiarized with the Cartesian plane coordinates system where you need to know the (x,y) coordinate.

I'm not going to focus in this Cartesian plane coordinates system just I'm going to say that

X is for horizontal  positive or negative

Y is for vertical positive or negative

Also keep in mind that if you want to draw a line you need at least 2 dots which mean that you need ($x_1$, $y_1$), ($x_2$, $y_2$)

Ok now I'm going to show you how to draw a line in java:

```
import java.awt.Graphics;
import javax.swing.JPanel;

public class Line extends JPanel{
	public void paint(Graphics g){
		super.paint(g);
		int width= getWidth();//the total width
		int height= getHeight();//the total height

		//this is the line that makes the draw
		g.drawLine(0, 0, width, height);
	}
}
```

To execute this class we just have to call it from the main which it's in other class

Bout classes must be in the same package

```
import javax.swing.JFrame;
public class Main {

	public static void main(String[] args) {
		Line line = new Line();
		JFrame frame= new JFrame();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.add(line);
		frame.setSize(300, 300);
		frame.setVisible(true);
	}
}
```

I think the code regarding to the main class its comprehensive but if you have any daub feel free to ask and as soon as a could I'll answer.

Once you run the application you may be are thinking what about if you need to draw more than a single line.

Well in the next code I'll show you how to do it.

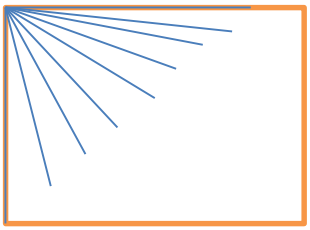
```
import java.awt.Graphics;
import javax.swing.JPanel;
public class CrossingLines extends JPanel{
	public void paint(Graphics g){
		super.paint(g);
		int height = getHeight();
		int width = getWidth();
		//the follow line of code draws a vertical line from left to right
		g.drawLine(0, 0, height, width);
		//the follow line of code draws a vertical line from right to left
		g.drawLine(width, 0, 0, height);
	}
}
```

```
import javax.swing.JFrame;
public class Main {

	public static void main(String[] args) {
		CrossingLines lines = new CrossingLines();
		JFrame drawFrame = new JFrame(); //this its going to be our window
		//so we have to prepare it to support a close operation
		//the follow line do it
		drawFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		//now we have to specify the window size (w and h)
		drawFrame.setSize(300, 300);
		//now we should add the drow that we already have
		drawFrame.add(lines);
		// now we have to make it visible otherwise were not going to be able to
see anything
		drawFrame.setVisible(true);
	}
}
```

As you can see when we want to draw more than one single line we just to put as many g.drawLine as many lines we want to draw.

Now it's time to apply something little bit more complex.

The idea its try to do something like the following figure:



As we can see in the image below, we can appreciate that de p1 its always in the same position and the p2 its equals to the half of the height and the width is from 0 to the half of the width. Knowing that, we know is able to use a loop to draw many lines.

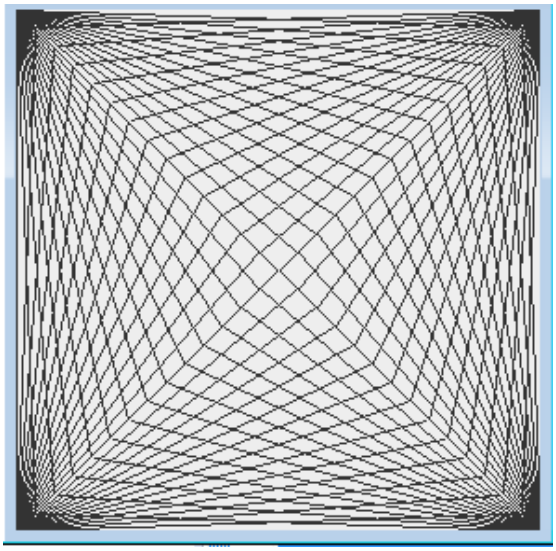
Let's try to do this

Everyone could have their own approach but  here I show you mains.

```
import java.awt.Graphics;
import javax.swing.JPanel;
public class Lines extends JPanel{

	public void paint(Graphics g){
		//setting the initial p2 values
		int x2=0;
		int y2 = getHeight();

		//setting our limits
		int xlimit = getWidth()/2;
		int ylimit = getHeight();

		super.paint(g);
		//now we have to calculate the values for the p2(x2,y2)
		while (x2 <= xlimit && y2 >= 0){
			g.drawLine(0, 0, x2, y2);
			//simple math to find the p2
			x2+=5;
			y2-=5;
		}
	}
}
```

```
import javax.swing.JFrame;
public class Main {

	public static void main(String[] args) {
		Lines lines = new Lines();
		JFrame drawFrame = new JFrame();

		drawFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		drawFrame.add(lines);
		drawFrame.setSize(400, 200);
		drawFrame.setVisible(true);
	}
}
```

Taking as base the example above let's try to do something like the next image:



This is my approach to get this image as result

```
import java.awt.Graphics;
import javax.swing.JPanel;
public class Lines extends JPanel{
	//-----------------------------
	//defining our set of points
	//to make our draw
	//-----------------------------
	private int [] p1 = new int[2];
	private int [] p2 = new int[2];
	private int [] p3 = new int[2];
	private int [] p4 = new int[2];
	public void paint(Graphics g){

		//setting the initials values
		int maxWidth = getWidth();
		int maxHeight = getHeight();
		p1[0] = 0;
		p1[1] = maxHeight;
		p2[0] = maxWidth;
		p2[1] = maxHeight;
		p3[0] = 0;
		p3[1] = 0;
		p4[0] = maxWidth;
		p4[1] = 0;
		//-----------------------------------------

		while ( p1[0] <= maxWidth && p1[1] >= 0 && p2[0] >= 0 && p2[1]>=0 &&
				p3[0] <= maxWidth && p3[1] <= maxHeight && p4[0] >= 0 &&
				p4[1] <= maxHeight){
			g.drawLine(0, 0, p1[0],p1[1] );
			p1[0]+=10;
			p1[1]-=10;
```

```
                        g.drawLine(maxWidth, 0, p2[0],p2[1]);
                        p2[0]-=10;
                        p2[1]-=10;

                        g.drawLine(0, maxHeight, p3[0], p3[1]);
                        p3[0]+=10;
                        p3[1]+=10;

                        g.drawLine(maxWidth, maxHeight, p4[0], p4[1]);
                        p4[0]-=10;
                        p4[1]+=10;
                }

        }
}

import javax.swing.JFrame;
public class Main {
        public static void main(String args[]){
                Lines lines = new Lines();
                JFrame drawFrame = new JFrame();

                drawFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                drawFrame.setSize(400, 400);
                drawFrame.add(lines);
                drawFrame.setVisible(true);
        }
}
```

Here it's another example let's say of lineal art using loops