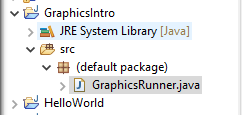
# Introduction to Graphics

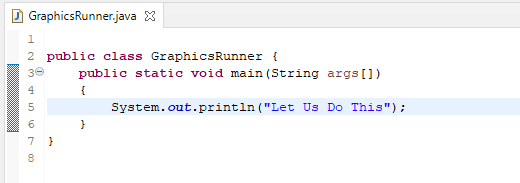
The programmer will be expected to use graphics in this year’s course. Graphics will be an integral part in producing the Graphical User Interface, or GUI, for many projects this year.

Create a Java Project called Graphics Intro. Inside of this Project created a class called GraphicsRunner and test it out by just printing a phrase out to see if it works.

Here is the Package Explorer portion:



And here is the Runner portion



It should output “Let Us Do This”. Now the programmer is ready to create a graphic window. The programmer has many choices on how to write graphics programs. One of the way the programmer could do this is to create an object that inherits the JFrame object in which to place the graphics. JFrame is an object class that can be found in the API. The programmer should familiarize themselves with the JFrame API.

The programmer then instantiates a soon to be created FirstGraphic object. This will replace the “Let’s Do this line of code

**public** **class** GraphicsRunner {

**public** **static** **void** main(String args[])

{

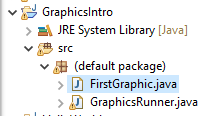
FirstGraphic graphicWindow =

**new** FirstGraphic("My First Graphic");

}

}

The programmer would then create a new class and place it into the GraphicsIntro Project called FirstGraphic.



This class should extend the JFrame class. In order to use JFrame, the programmer should import the following:

**import** java.awt.\*;

**import** javax.swing.\*;

**public** **class** FirstGraphic **extends** JFrame{

}

One of the ways a programmer could set a window is to create a constant WIDTH and HEIGHT, such that all mathematics on the width and height of the window can be proportional to whatever constant they set it to. Notice, the programmer set the WIDTH and HEIGHT to private so that no one outside this class could alter it.

**public** **class** FirstGraphic **extends** JFrame{

**private** **static** **final** **int** ***WIDTH*** = 1200;

**private** **static** **final** **int** ***HEIGHT*** = 480;

}

The programmer would then create the class’ constructor. Since the class inherited from JFrame, the programmer needs to use the super to instantiate the object. The API will show the programmer all the necessary constructors that JFrame has. They would then add in setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE) to the code. This will allow the program end when the window is closed. They would then add setSize(WIDTH,HEIGHT) to set the size of the window. Finally, they would add setVisible(true) such that the window can be seen. All of this is done so that a window will be shown when the program is run.

**public** FirstGraphic(String framename)

{

**super**(framename);

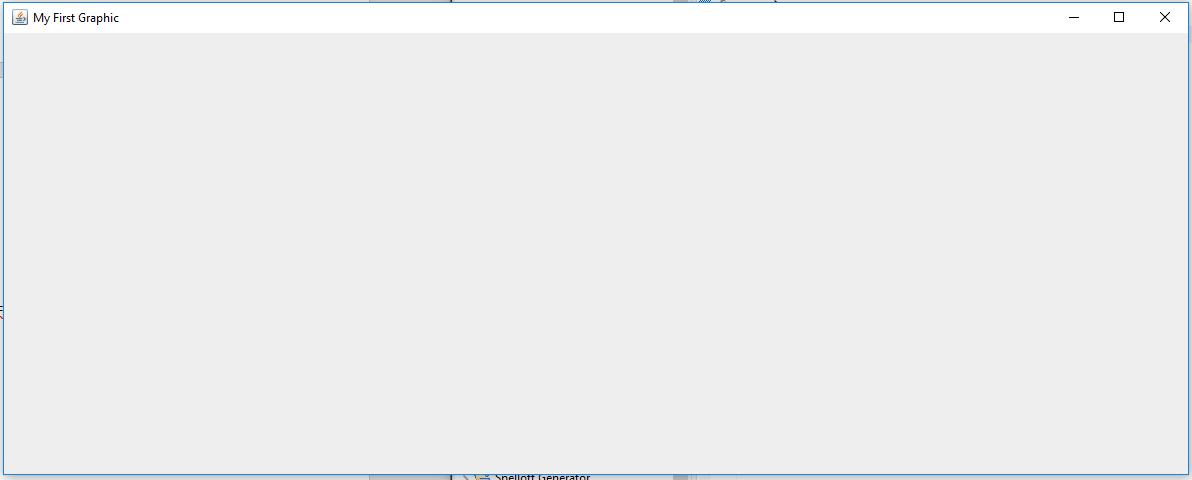
setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setSize(***WIDTH***,***HEIGHT***);

setVisible(**true**);

}

When the GraphicsRunner class is run, an empty window will show up.



In order to draw on the window, the programmer would have to create a panel to draw on. The panel is where most of the heavy listing is done. The panel will be a class the inherits the JPanel class and has a method called paint in it. Create a new class called FirstGraphicPanel and extend the JPanel class. In order for JPanel to work, the programmer needs to do the following imports:

**import** java.awt.\*;

**import** javax.swing.\*;

**public** **class** FirstGraphicPanel **extends** JPanel

{

}

This class must have a paint method that receives a Graphics object that is sent to it. The programmer could also write other methods and constructors as needed. In this case, a constructor is not needed and the programmer could choose to not write the constructor. The constructor is shown here just to let the programmer know that a constructor can appear in the class. Inside of paint, the programmer could call **super.paint()**, this is very useful when writing event driven programs. **super.paint()** allows the Panel to reset to a blank state, which is very useful when you need to redraw on the panel and do not want the old information to been seen. **super.paint()** must be written in the first line. The programmer can draw a blue line from the upper left corner of the window (0,0) to the lower right corner of the window (WIDTH, HEIGHT). In order for the programmer to know the width and the height, they call the JPanel methods getWidth() and getHeight() which will return the correct dimensions.

**public** **class** FirstGraphicPanel **extends** JPanel

{

**public** FirstGraphicPanel()

{}

**public** **void** paint(Graphics g)

{

super.paint(g);

g.setColor(Color.***BLUE***);

g.drawLine(0, 0, getWidth(), getHeight());

}

}

The FirstGraphicPanel must be added to the FirstGraphic Frame in order for it to work so the programmer adds a Panel to the frame before the setVisible command.

**public** **class** FirstGraphic **extends** JFrame{

**private** **static** **final** **int** ***WIDTH*** = 1200;

**private** **static** **final** **int** ***HEIGHT*** = 480;

**public** FirstGraphic(String framename)

{

**super**(framename);

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setSize(***WIDTH***,***HEIGHT***);

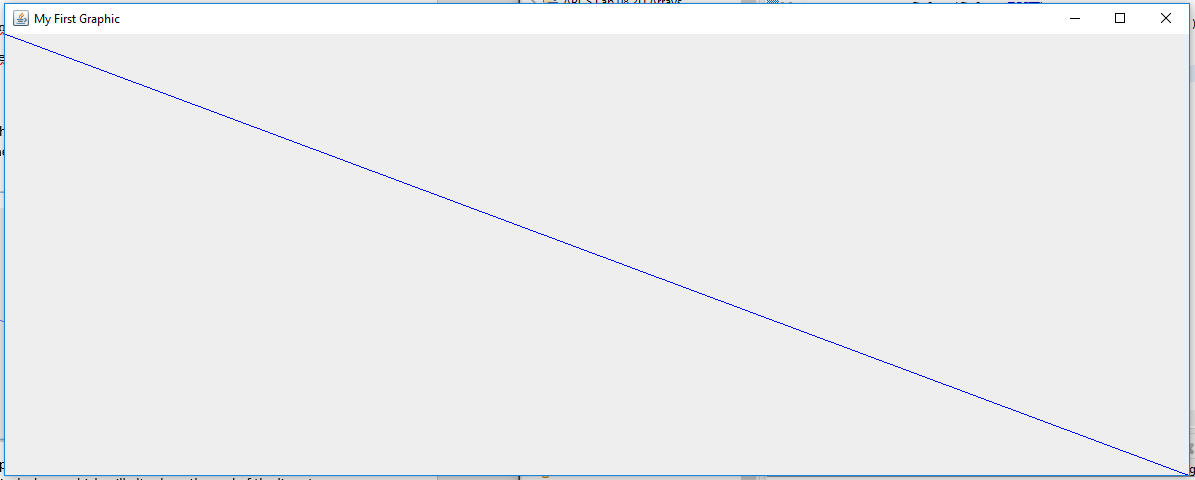
add(**new** FirstGraphicPanel());

setVisible(**true**);

}

}

It is important to note that the x coordinate is the width and it represents the columns of pixels, while the y coordinate is the height and represents the rows of pixels. It does not behave like a Cartesian Plane. The window will show the following



The programmer has now successfully created a graphic program. The next set of notes will go over how to JAR up the Project so that the program can run without using an IDE.