LBYCP12

Data Structures and Algorithm Analysis Laboratory

Laboratory Activity 1

Getting Started; Introduction to Java; Tool Setup

By Patrick Matthew J. Chan

Your Name, LBYCP12-EQ1/2

INTRODUCTION

This activity aims to introduce creating various console programs and graphical applications with Java programming using the `acm` package. The activity would include the installation and setup of an Integrated Development Environment (IDE), writing sample programs to solve simple problems, and documenting the results.

OBJECTIVES

- To learn about the necessary packages and softwares needed in writing Java programs, and their installation.
- To introduce the Java programming language to students
- To understand the use of variables, methods, control structures, and other features of Java, using the `acm` package.

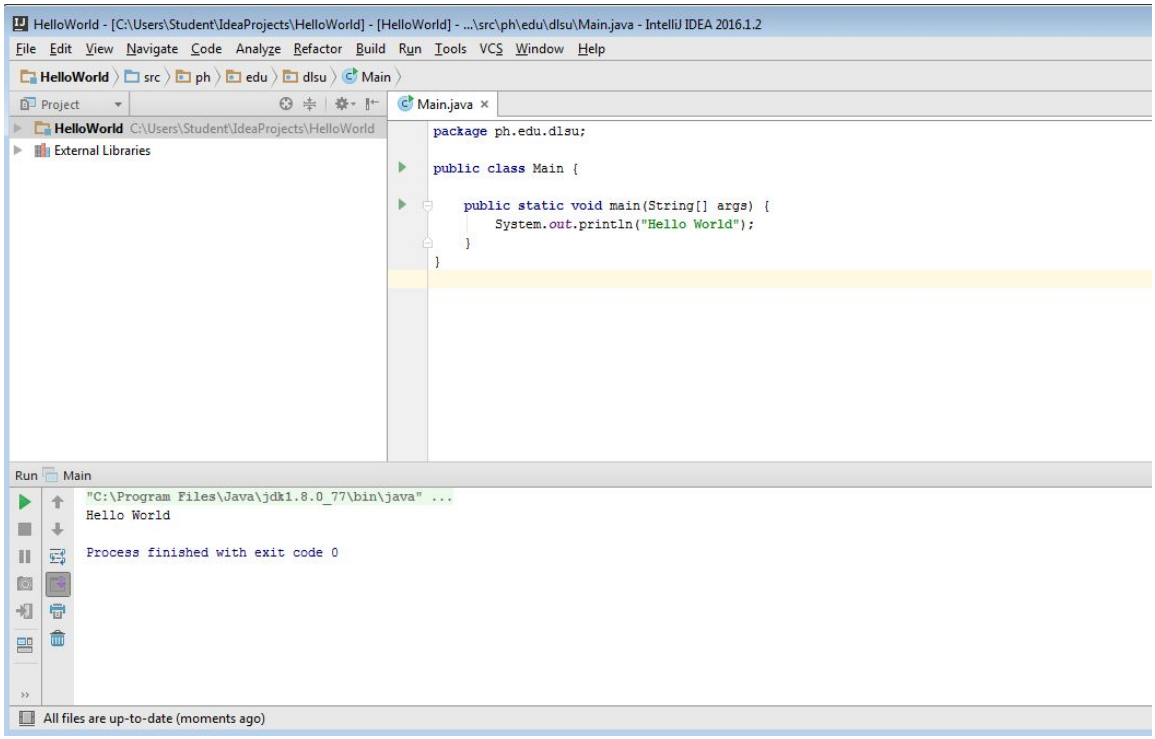
MATERIALS

1. Java SE Development Kit (JDK) 8
2. Integrated development environment (IDE)
    - IntelliJ IDEA or NetBeans IDE
3. `acm.jar` by the ACM Java Task Force

PROCEDURE

1. Download/install JDK 8 and IntelliJ IDEA or NetBeans IDE in your workstation.

2. Run a `Hello World!` Program (standard Java) in IntelliJ IDEA or NetBeans IDE.



3. Create a new project (entitled `HelloACM)`, include the `acm.jar` module to this project, and print `Hello ACM!` as follows:

```
/*
 * File: HelloProgram.java
 * -----------------------
 * This program displays "hello, world" on the screen.
 * It is inspired by the first program in Brian
 * Kernighan and Dennis Ritchie's classic book,
 * The C Programming Language.
 */

import acm.graphics.*;
import acm.program.*;

public class HelloProgram extends GraphicsProgram {
   public void run() {
      add(new GLabel("hello, world"), 100, 75);
   }
}
```

program comment

imports

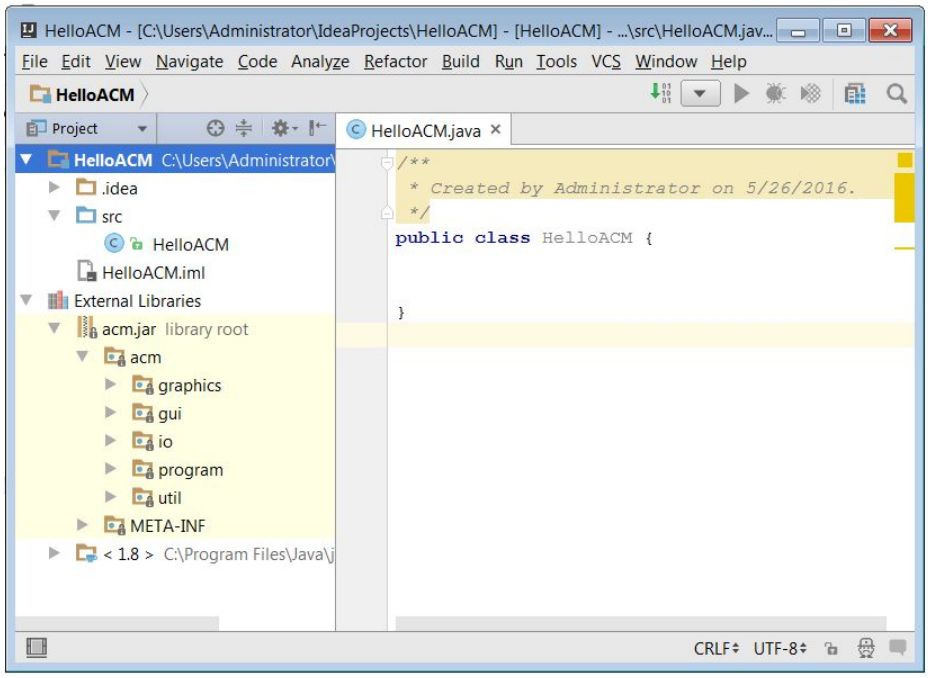
main class

Including the `acm.jar` to your project: (for IntelliJ IDEA)

- Go to **File >> Project Structure >> Modules >> Dependencies** and click **+** button, then choose **jars or directories...** .
- Look for your `acm.jar` file and choose it and click **ok**.
- You can check if it's successfully added by clicking **External Libraries** on

the left side of the window, and `acm.jar` is in the list.

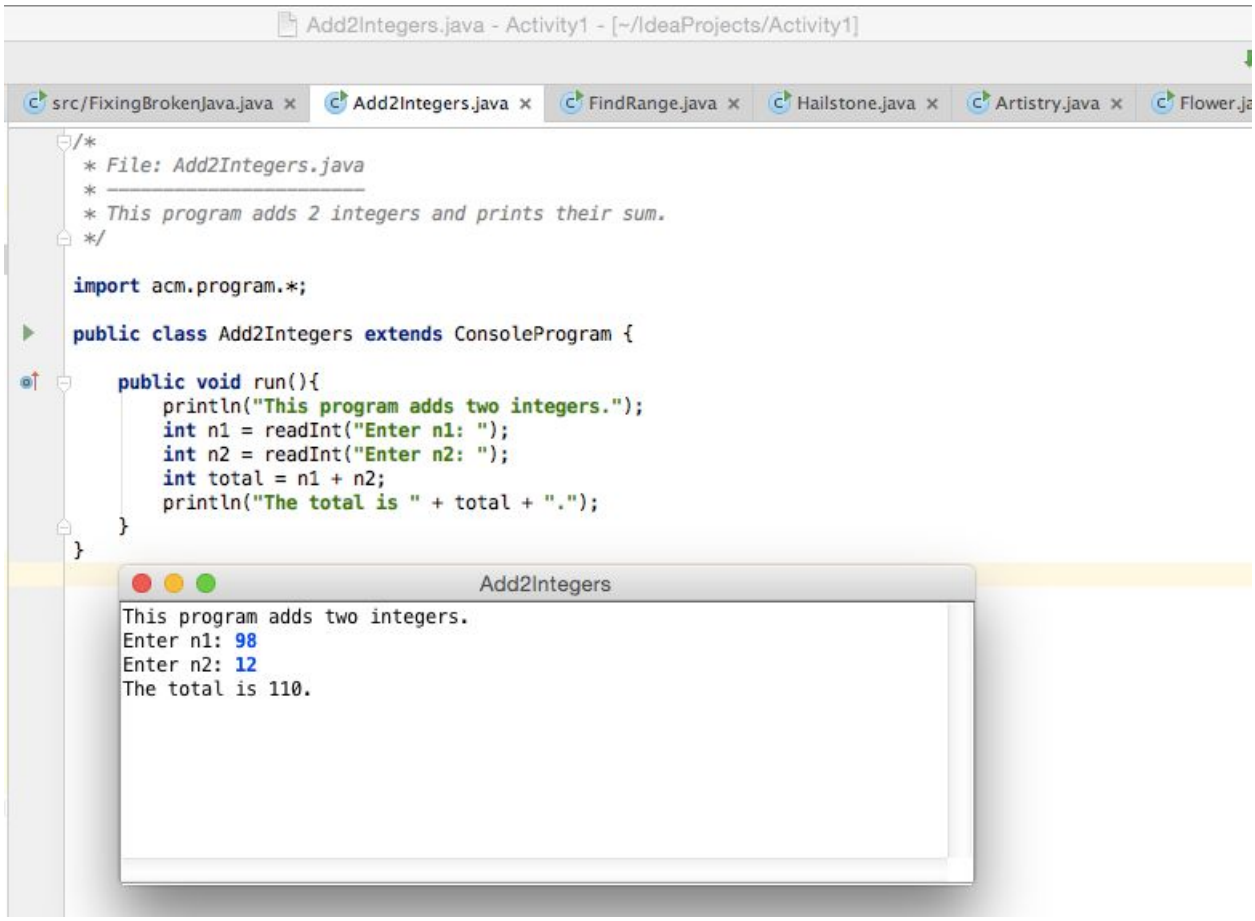
Ex.



4. To familiarize the user input for Java with `acm.jar`, run the following sample program for adding two numbers, i.e. integers or doubles, and print their sum.

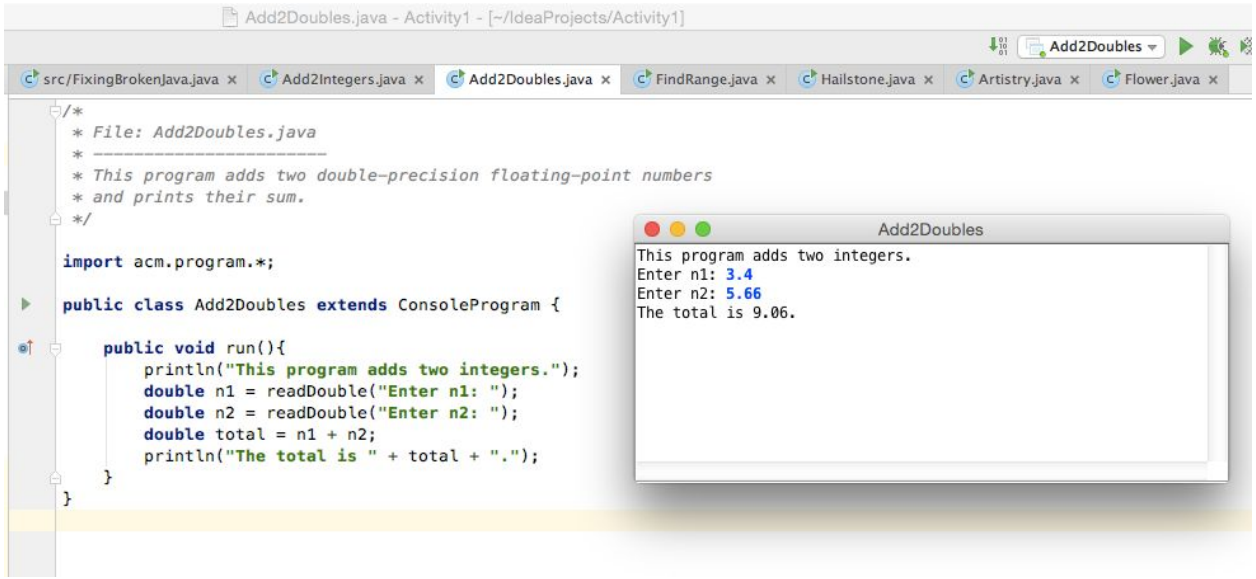
   a)

```
/*
 * File: Add2Integers.java
 * -----------------------
 * This program adds two integers and prints their sum.
 */

import acm.program.*;

public class Add2Integers extends ConsoleProgram {

   public void run() {
      println("This program adds two integers.");
      int n1 = readInt("Enter n1: ");
      int n2 = readInt("Enter n2: ");
      int total = n1 + n2;
      println("The total is " + total + ".");
   }

}
```



b)

```
/*
 * File: Add2Doubles.java
 * ----------------------
 * This program adds two double-precision floating-point numbers
 * and prints their sum.
 */

import acm.program.*;

public class Add2Doubles extends ConsoleProgram {

   public void run() {
      println("This program adds two numbers.");
      double n1 = readDouble("Enter n1: ");
      double n2 = readDouble("Enter n2: ");
      double total = n1 + n2;
      println("The total is " + total + ".");
   }

}
```



5. Solve the following programming exercises:

a.) **The Pythagorean Theorem:**

$$a^2 + b^2 = c^2$$

One of the oldest results in mathematics is the Pythagorean theorem, which relates the lengths of the three sides of a right triangle. Write a ConsoleProgram that accepts values for a and b as doubles (you can assume that a and b will be positive) and then calculates the solution of c as a double. Your program should be able to duplicate the following sample run, plus runs with other values:

b.) Find Range

Write a ConsoleProgram that reads in a list of integers, one per line, until the user enters a sentinel value of 0 (which you should be able to change easily to some other value). When the sentinel is read, your program should display the smallest and largest values in the list, as illustrated in this sample run:

Your program should handle the following special cases:

• If the user enters only one value before the sentinel, the program should report that value as both the largest and smallest.

• If the user enters the sentinel on the very first input line, then no values have been entered, and your program should display a message to that effect.

c.) Broken Java

As you begin writing larger and larger programs, you will invariably end up making mistakes in your code. Don't worry – this is a normal part of programming! To help you navigate the error messages you'll get in Eclipse when your code isn't working correctly, as well as to give you practice debugging an incorrect program, we have provided you a file called FixingBrokenJava.java that contains, unsurprisingly, a broken Java program. Your task in this part of the assignment is to fix the program so that it works correctly.

The program we've provided you reads in a positive integer from the user and then checks whether that number is *prime*; that is, whether it has any divisors other than 1 and itself. For example, 3 is prime and 31 is prime, but 54 is not (it's divisible by two) and 49 is not (it's divisible by seven). Unfortunately, our provided program has many problems: it doesn't compile due to syntax errors, and the logic itself is incorrect. Your task is to

- Fix the program so that it compiles, and
- Correct the logic errors so that it works correctly.

The broken code:

```
/*
 * File: FixingBrokenJava.java
 * Name:
 * Section Leader:
 *
 * This program does not work as intended.  It contains both
 * compile-time errors (errors that prevent the compiler from even
 * running the program) and run-time errors (errors where the
 * program does not function as intended).  Your job is to fix
 * this program so that it works correctly.  Note that it is *not*
 * sufficient to simply fix the compiler errors; you will need to
 * update the logic as well.
 *
```

```
 * This program attempts to read a positive integer from the user,
 * then check whether that integer is prime (whether its only
 * divisors are 1 and itself).  If so, it prints a message saying
 * that the number is prime; otherwise it says that the number is
 * composite.
 */
public class FixingBrokenJava extends ConsoleProgram {
	/* Reads a number from the user and reports whether or not it
	 * is prime.
	 */
	public void run() {
		/* Get the value from the user. */
		int value = readPositiveInt();

		/* Check whether or not it is prime. */
		if (isPrime(value)) {
			println(value + " is prime.")
		} else {
			println(value + " is composite.");
		}
	}

	/**
	 * Given a positive integer, returns whether that integer is
	 * prime.
	 *
	 * @param value The value to test.
	 * @return Whether or not it is prime.
	 */
	private boolean isPrime(int value) {
		/* Try all possible divisors of the number.  If any of them
		 * cleanly divide the number, we return that the number is
		 * composite.
		 */
		for (int divisor = 0; divisor <= value; divisor++) {
			if (value % divisor == 0) {
				return false;
			}
		}
	}

	/**
	 * Reads a positive integer from the user and returns it.
	 *
```

```
         * @return A positive integer entered by the user.
         */
        private int readPositiveInt() {
                /* Get an initial value. */
                int value = readInt("Enter a positive integer: ");

                /* If the value was nonpositive, reprompt the user. */
                while (value <= 0) {
                        println("Please enter a positive integer.");
                        int value = readInt("Enter a positive integer: ");
                }

                return value;
        }
}
```

d.) Hailstone Sequence

Douglas Hofstadter's Pulitzer-prize-winning book *Gödel, Escher, Bach* contains many interesting mathematical puzzles, many of which can be expressed in the form of computer programs. In Chapter XII, Hofstadter mentions a wonderful problem that is well within the scope of the control statements from Chapter 4. The problem can be expressed as follows:

Pick some positive integer and call it *n*.
If *n* is even, divide it by two.
If *n* is odd, multiply it by three and add one. Continue this process until *n* is equal to one.

On page 401 of the Vintage edition, Hofstadter illustrates this process with the following example, starting with the number 15:

15 is odd, so I make 3*n*+1: 46
46 is even, so I take half: 23
23 is odd, so I make 3*n*+1: 70
70 is even, so I take half: 35
35 is odd, so I make 3*n*+1: 106
106 is even, so I take half: 53

53 is odd, so I make 3*n*+1: 160
160 is even, so I take half: 80
80 is even, so I take half: 40
40 is even, so I take half: 20
20 is even, so I take half: 10
10 is even, so I take half: 5
5 is odd, so I make 3*n*+1: 16
16 is even, so I take half: 8
8 is even, so I take half: 4
4 is even, so I take half: 2
2 is even, so I take half: 1

As you can see from this example, the number goes up and down, but eventually—at least for all numbers that have ever been tried—comes down to end in 1. In some respects, this process is reminiscent of the formation of hailstones, which get carried upward by the winds over and over again before they finally descend to the ground. Because of this analogy, this sequence of numbers is sometimes called the Hailstone sequence, although it goes by many other names as well.

Write a ConsoleProgram that reads in a number from the user and then displays the Hailstone sequence for that number, just as in Hofstadter's book, followed by a line showing the number of steps taken to reach 1.

e.) Artistry!

Now that you have the acm.graphics package available to you, you have the ability to turn the computer into a digital canvas. To warm up with the graphics package, why not take a few minutes to draw a pretty picture?

In this part of the assignment, your job is to draw a picture of your choice using the acm.graphics package. The requirements for this part of the assignment are fairly lax (they're described shortly), so feel free to use this as an opportunity to play around with Java and see what you can make with the tools that are now available to you!

For this assignment, you should draw a picture with the following properties:

1. Your picture must use at least three different types of GObjects – for example, you could use GLine, GRect, and GOval.
2. Your picture must have at least one filled object.

3. Your picture must have at least least two different colors of objects.
4. Your picture must have at most twenty total GObjects in it. While we encourage you to have fun with this part of the assignment, we don't want you to go too crazy with it. :-)
5. You must sign your name in the bottom-right corner. To do this, create a GLabel with the phrase “Artistry by *name*,” where *name* is your name, and align it so that it is flush up against the bottom-right corner of the window. Be sure that all the text is visible and that none of the letters in the GLabel are cut off. (This GLabel doesn't count toward the the limits on the number of GObjects that you can have in the picture, nor does it count as one of the three different types of GObjects that you're required to have. If you want to count GLabel as one of the GObject types you're using, you'll need to have a second GLabel in your picture).

Although you're free to draw whatever you'd like, be sure to write clean and elegant code as you would in the other parts of this assignment. Have a nice decomposition, comment your code, use helper methods where appropriate, and prefer constants to magic numbers.

There are many other graphics objects besides GRect, GOval, GLabel, and GLine and you are free to use them in your program. Chapter 9 of the book gives a good overview of what else is out there. You may find the GArc and GPolygon classes particularly useful. Chapter 9 also describes the Color class in more detail, so if you're interested in colors beyond the standard set I would suggest giving it a read.

f.) Target

Suppose that you’ve been hired to produce a program that draws an image of an archery target—or, if you prefer commercial applications, a logo for a national department store chain—that looks like this:

This figure is simply three GOval objects, two red and one white, drawn in the correct order. The outer circle should have a radius of one inch (72 pixels), the white circle has a radius of 0.65 inches, and the inner red circle has a radius of 0.3 inches. The figure should be centered in the window of a GraphicsProgram subclass.

g.) Pyramid

Write a GraphicsProgram subclass that draws a pyramid consisting of bricks arranged in horizontal rows, so that the number of bricks in each row decreases by one as you move up the pyramid, as shown in the following sample run:

The pyramid should be centered at the bottom of the window and should use constants for the following parameters:

BRICK_WIDTH
BRICK_HEIGHT
BRICKS_IN_BASE

The width of each brick (30 pixels) The height of each brick (12 pixels) The number of bricks in the base (14)

The numbers in parentheses show the values for this diagram, but you must be able to change those values in your program.

RESULTS AND DISCUSSION

1. Possible Error with a `ConsoleProgram`: "`NoClassDefFoundError`"

```
Exception in thread "main" java.lang.NoClassDefFoundError: Could not initialize class FormPreviewFrame
    at java.lang.Class.forName0(Native Method)
    at java.lang.Class.forName(Class.java:264)
    at acm.util.JTFTools.readMainClassFromClassPath(JTFTools.java:1092)
    at acm.util.JTFTools.getMainClass(JTFTools.java:461)
    at acm.program.Program.main(Program.java:1320) <5 internal calls>
```

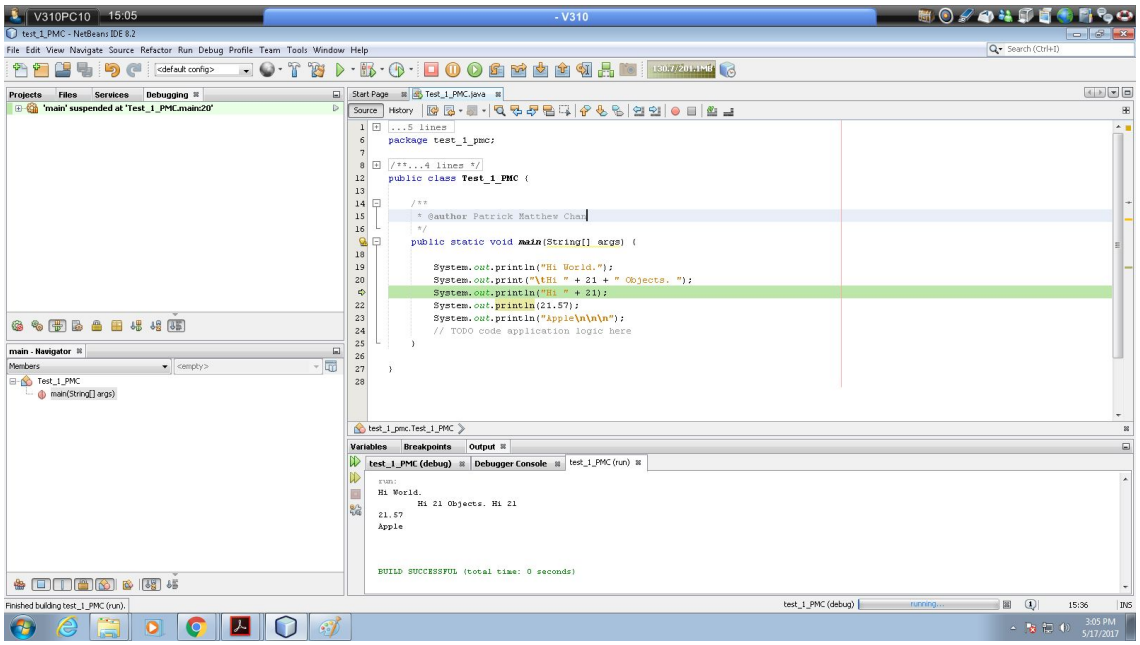
Solution:

Add `main()` function:

Ex.

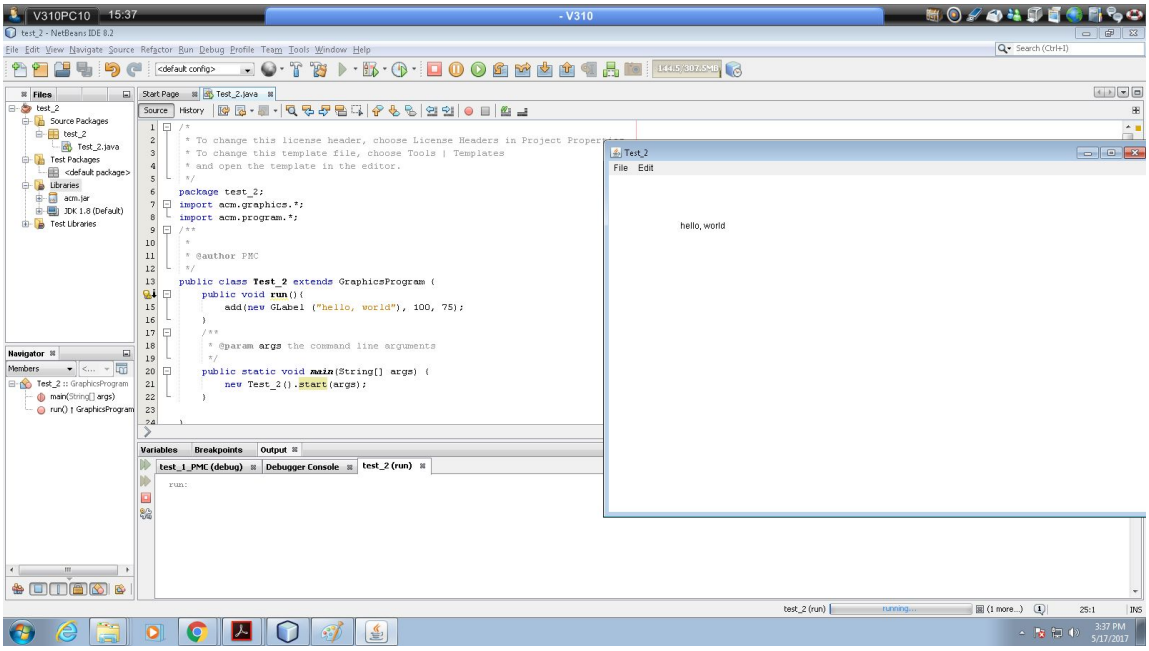
```
public static void main(String[] args) {

   new AddIntegers().start(args);    // AddIntegers is the class name

}
```

2. Hello World Program



My First Java Program

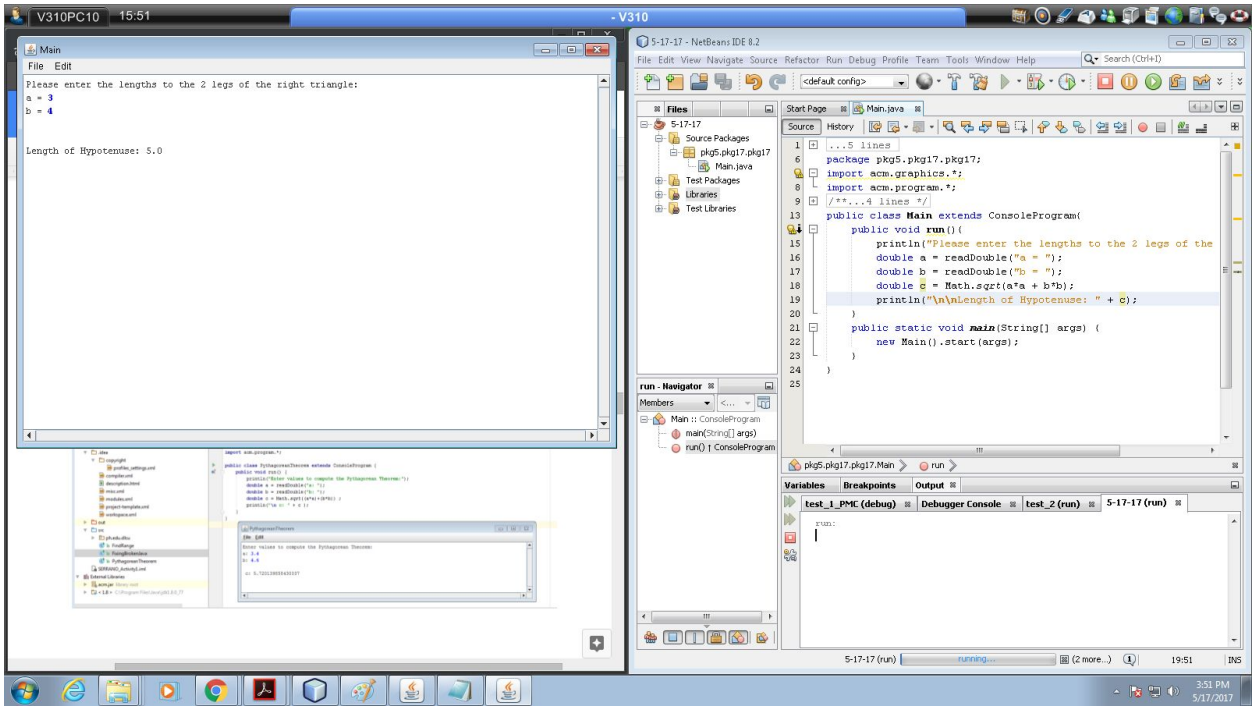
(Exploring the Usage of print and println in Java)



Hello World Program using `acm` package

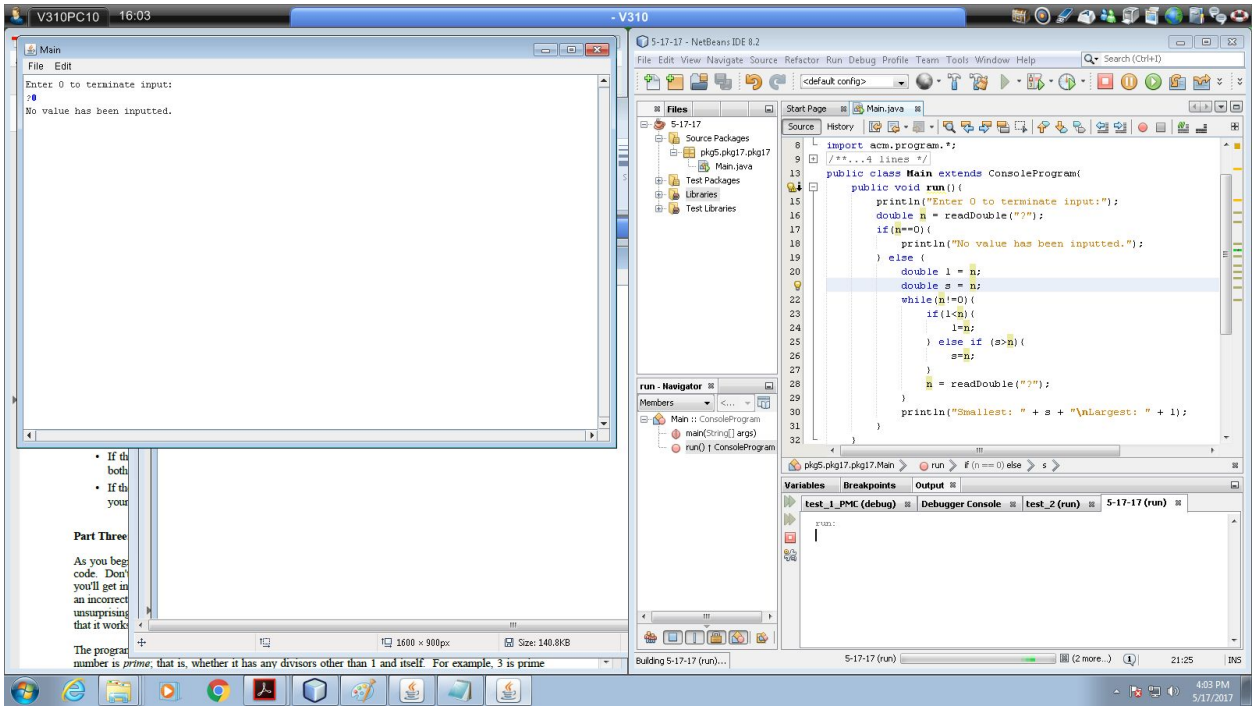
3. Results of Programming Exercises:

a. The Pythagorean Theorem

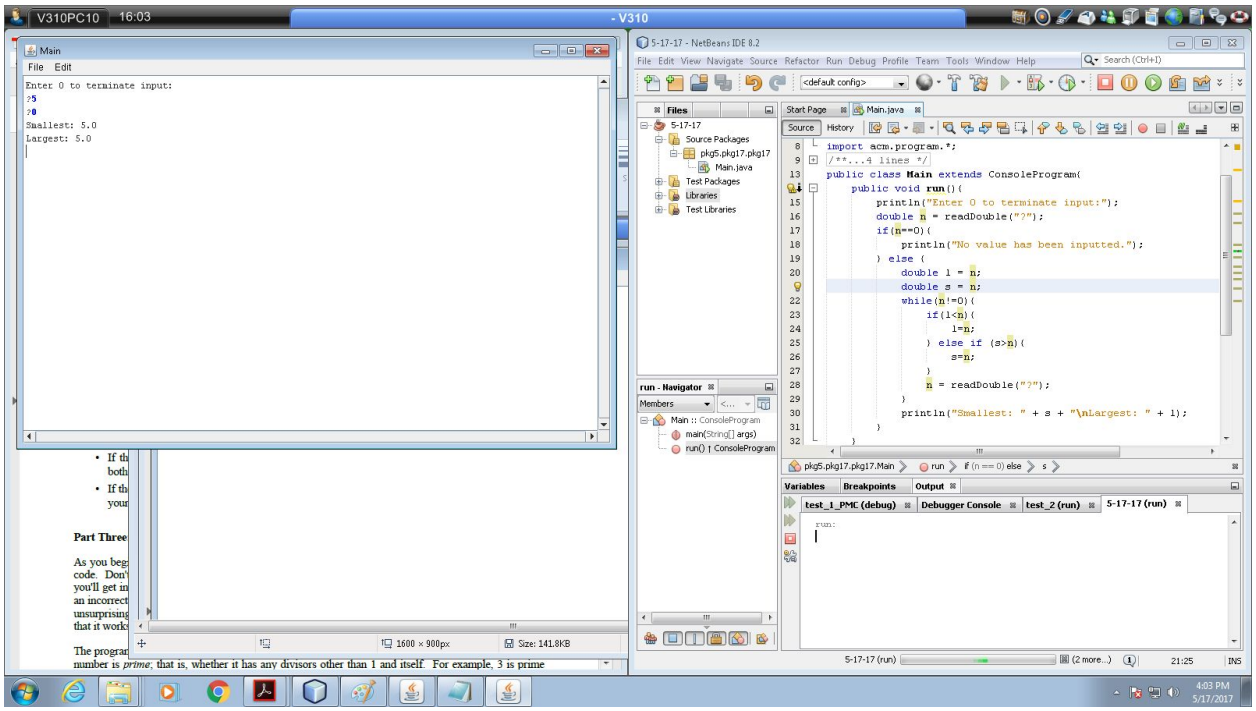


Program Computes for the Length of the Hypotenuse After Reading Two Inputs

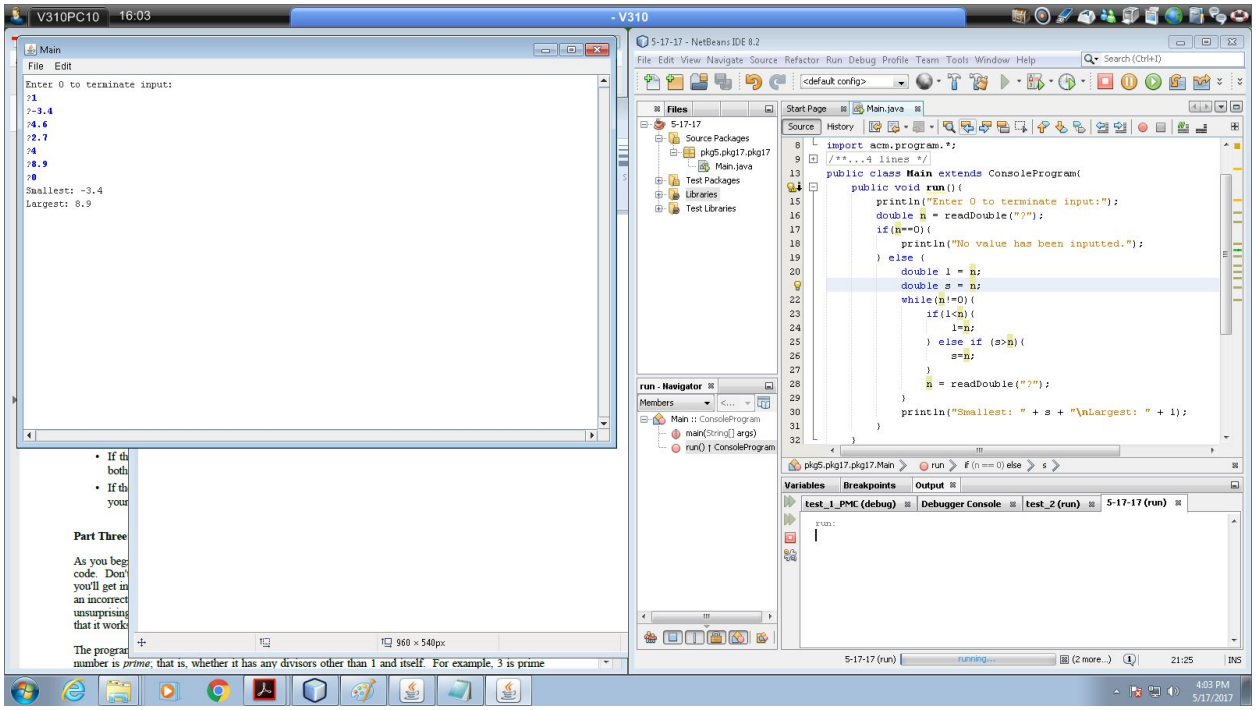
b. Find Range



With conditional statements in the code, if the first input is 0, the program would indicate that there were no input values instead.

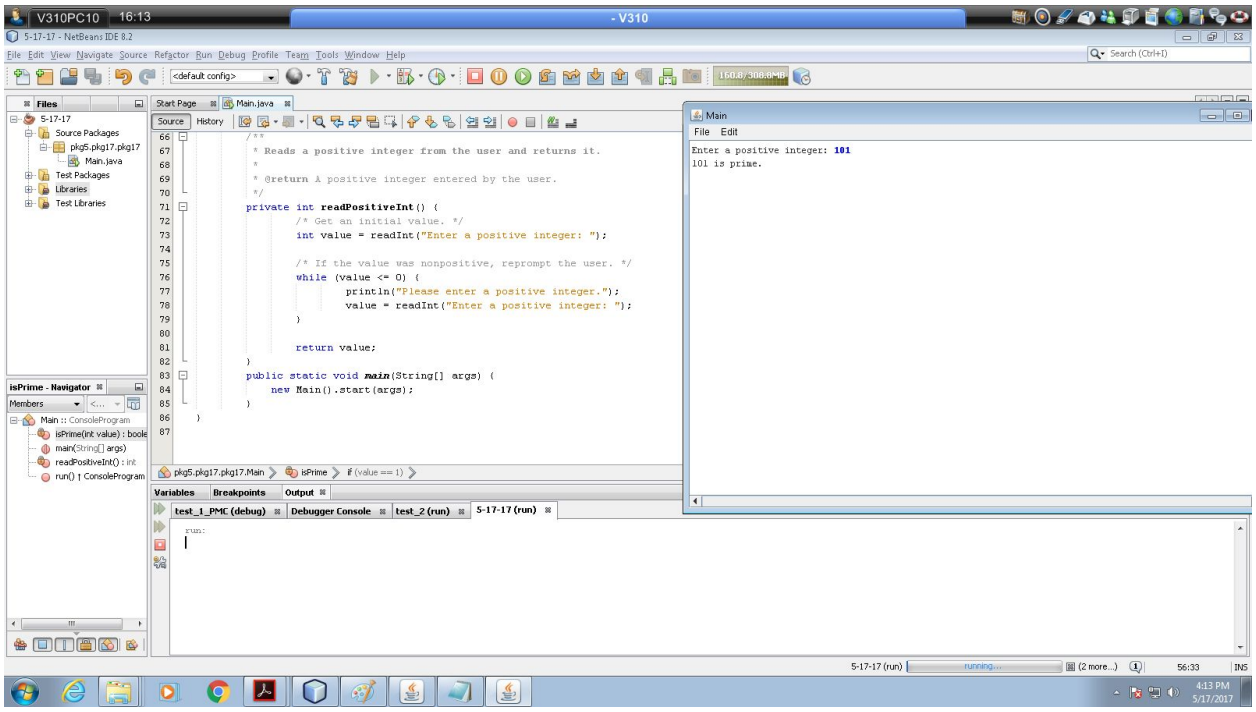


Having only one input results in it being simultaneously the largest and the smallest value.



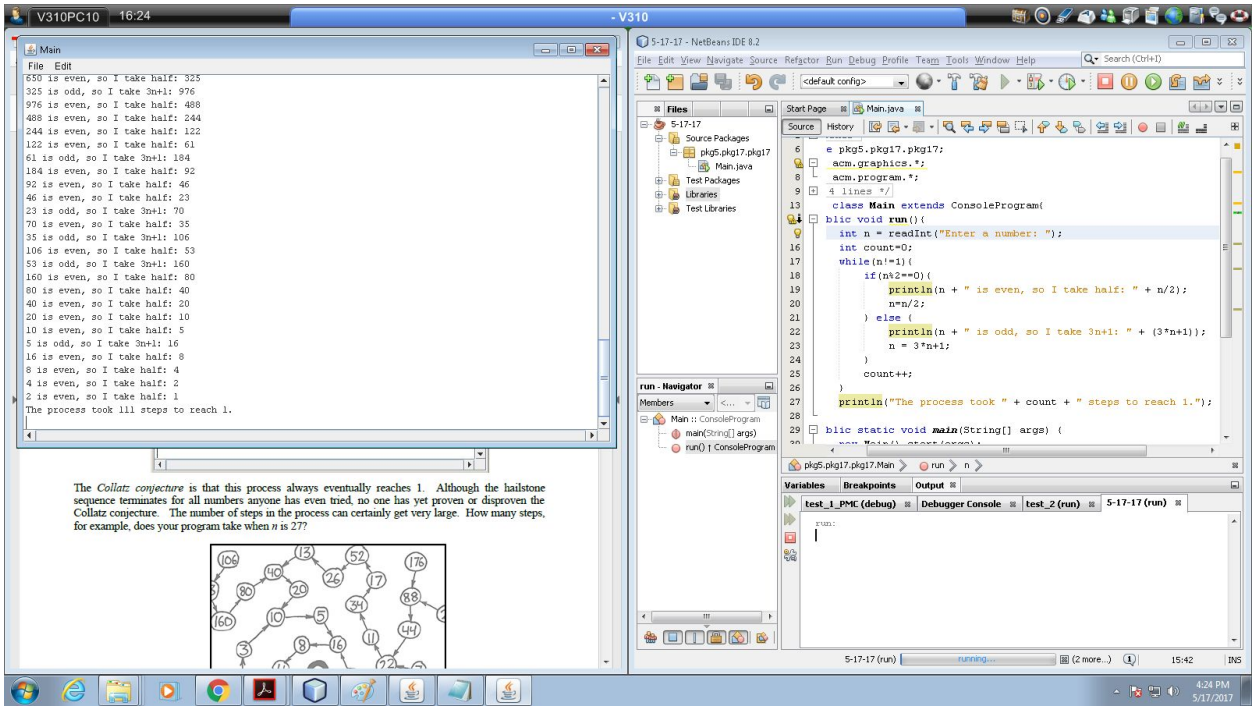
With iterative statements in the code, the program would keep accepting inputs until the input value is 0, before printing the smallest and largest values.

c. Broken Java



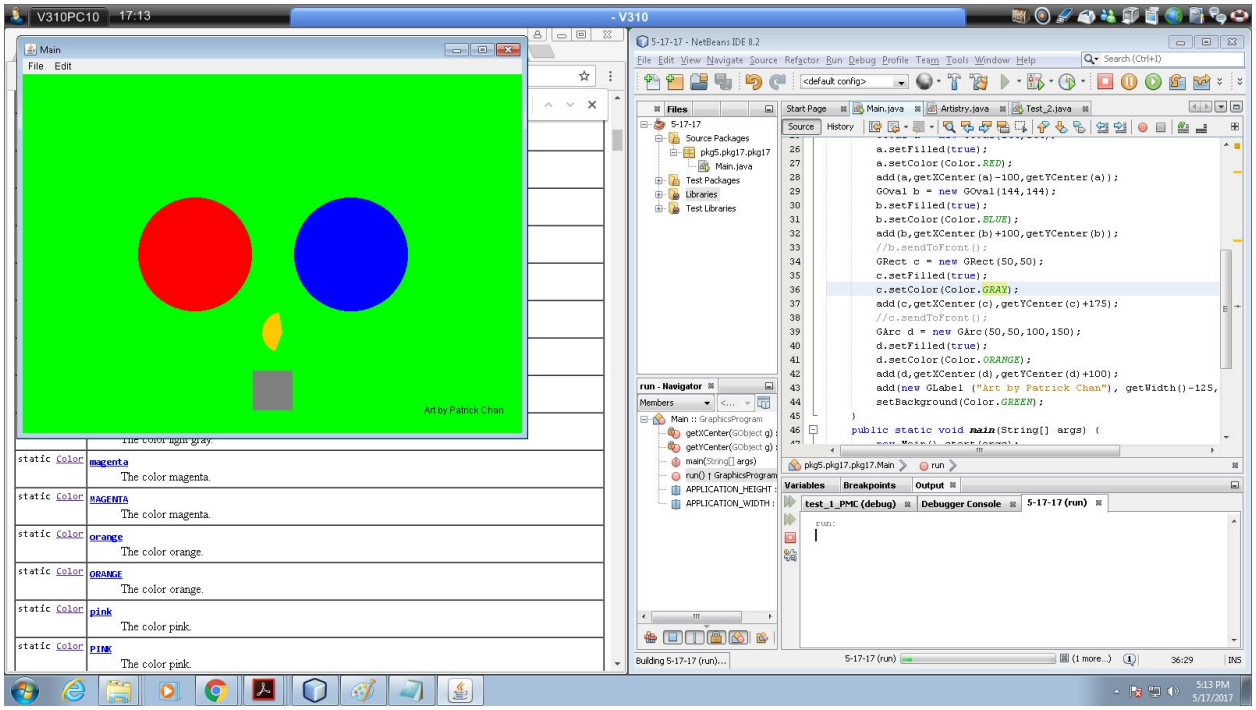
After fixing various syntax and logical errors, I finally ran the program and it functions as intended with correct results.

d. Hailstone Sequence



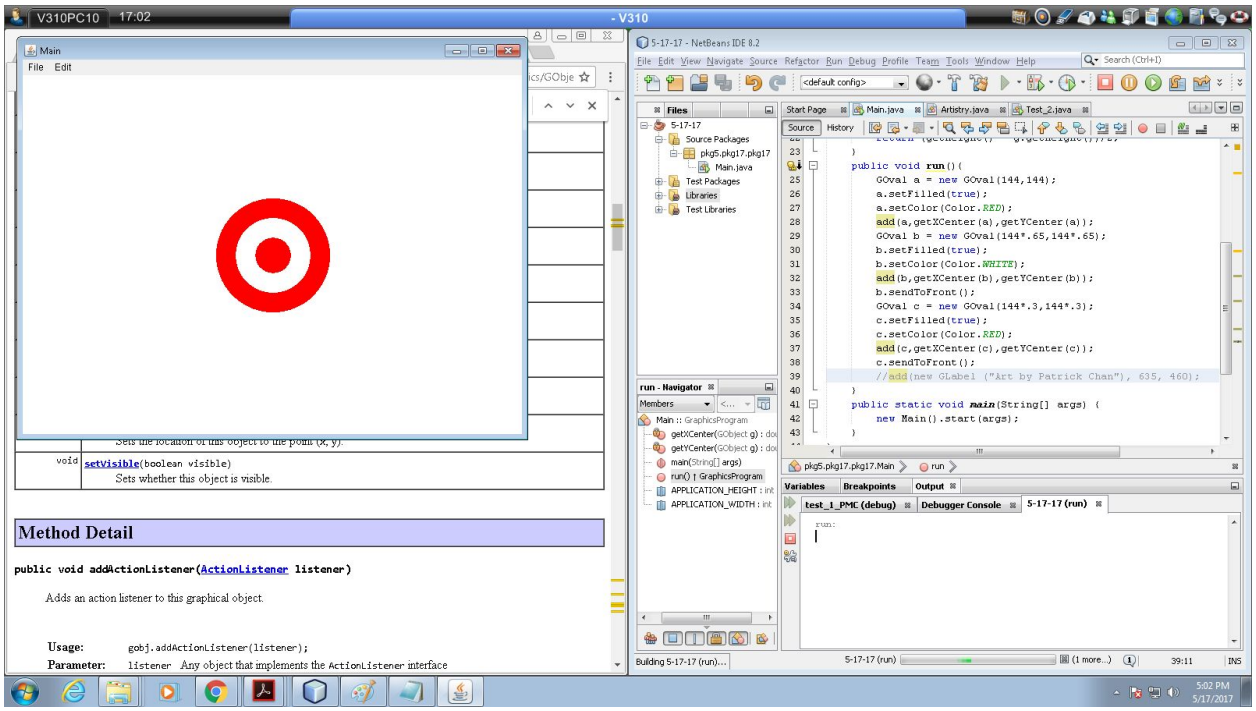
Inputting 27, as suggested by the procedures, takes 111 steps to reach 1.
(The program uses a counter that increments by one every time an operation is done in the manner of the Hailstone sequence)

e. Artistry!



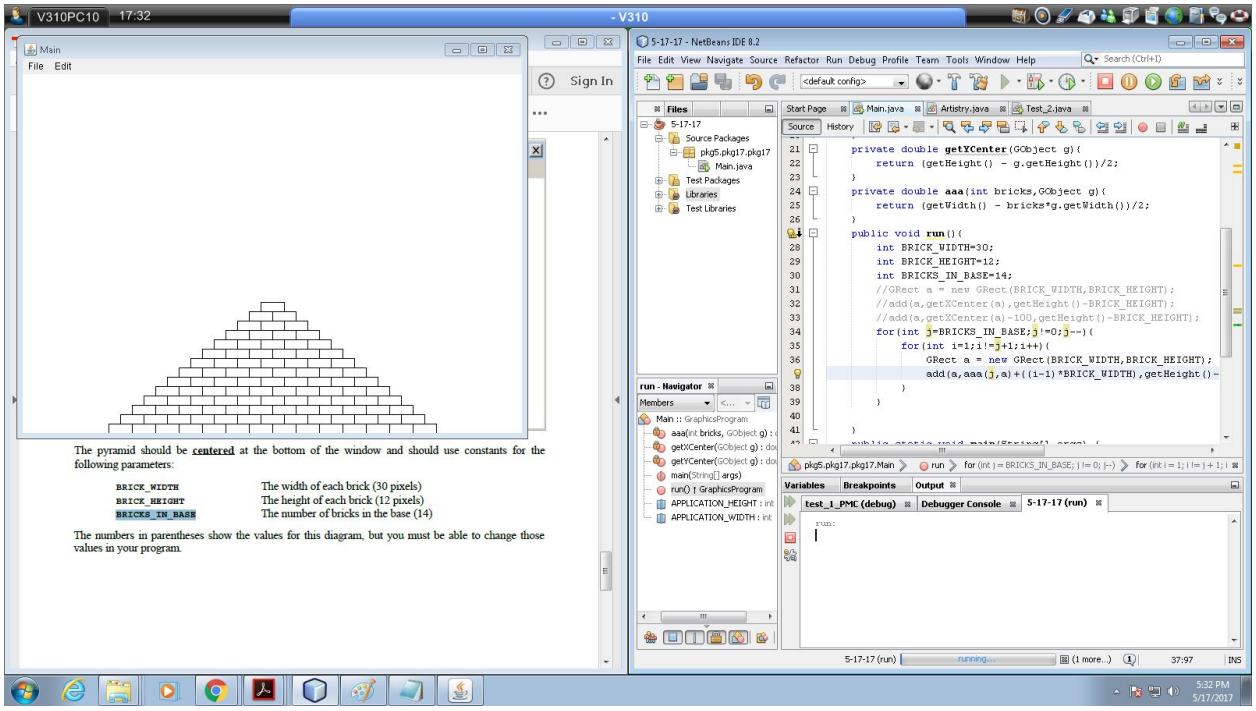
My artistry, using various GObjects, such as GOval, GArc, ...

f. Target



The target is centered, and the circles follow the sizes as instructed, by using ratios between inches and pixels.

g. Pyramid



Using two “for loops”, the program proceeds to draw a pyramid row by row. (The values for BRICK_WIDTH, BRICK_HEIGHT, BRICKS_BY_BASE are utilized as variables in the code, so that they can be easily changed.)

SUMMARY

In this activity, after learning about how to setup and install Java and an IDE to a computer, I learned about some basics in Java programming. This included some basic terminologies such as classes, with their states and behaviors, objects, methods, and etc. Here, I have also learned about importing packages, the main class to write my code, writing comments on the code, and the basic printing function *System.out.println()* I used to make my Hello World application.

Aside from this, I have seen the usefulness of the `acm` package. The first part of the programming exercises utilizes the `acm.program` package, which simplifies the program syntax, and runs the program on a separate window instead. Coming from having a background in C programming, I found it easy to adapt to this new language for making conditional and iterative statements, as the only differences I only need to remember so far are how the `acm` package uses *println()* to print values, and *readInt(),*

readDouble(),... to read specific types of inputs from the user instead. One thing that I have noticed and liked about these methods of reading user inputs, is that the `acm` package automatically checks the user input if it is in the correct format, reprompts the user if the format is "illegal", and only accepts the input if the format is correct.

On the other hand, the second part of the programming exercises utilizes the `acm.graphics` package, which instead creates a drawing space, or canvas, to draw graphics on the screen. In this activity, I have learned about the class GObjects, which consists of subclasses GRect, GOval, GLine, GLabel, GPolygon, GImage and GCurve, and allows various shapes and text labels to placed on the screen. Furthermore, these classes have various characteristics that can be used to customize its appearance on the canvas, such as its size, color, being simply an outline or being filled, visibility, z-dimension placement, etc. Then, various GObjects can be placed in a canvas to create an illustration onscreen.

Aside from learning the basics of Java, this activity has introduced me to the concept of object oriented programming, in which the creation of classes with its states and behaviors allows programmers to simplify the codes and processes for creating complex operations onscreen, such as drawing shapes with various dimensions on a new window. This feature allows programmers to simplify the creation of a graphical user interface and games, in a sense that he simply needs to tell the computer what needs to be drawn on screen, and where they are supposed to go, instead of having to draw the shapes one by one.

APPENDIX

The code:

a. **The Pythagorean Theorem**

```
package ph.edu.dlsu.chan.pythagoreantheorem;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
import java.awt.*;
```

```
import java.awt.event.*;
import acm.graphics.*;
import acm.program.*;
/**
 *
 * @author Patrick Matthew Chan
 */
public class PythagoreanTheorem extends ConsoleProgram{
   public void run(){
          println("Please enter the length to the 2 legs of the right triangle");
          double a = readDouble("a = ");
          double b = readDouble("b = ");
          double c = Math.sqrt(a*a + b*b);
          println("\n\nlength of hypotenuse: " + c);
      readLine("\n\nPress any key to repeat program");
      getConsole().clear();
      run();
   }
   public static void main(String[] args) {
      new PythagoreanTheorem().start(args);
   }
}
```

b. Find Range

```
package ph.edu.dlsu.chan.findrange;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
import acm.graphics.*;
import acm.program.*;
/**
 *
 * @author Patrick Matthew Chan
 */
public class FindRange extends ConsoleProgram{
   public void run(){
      println("Enter 0 to terminate input:");
```

```
        double n = readDouble("?");
        if(n==0){
           println("No value has been inputted.");
        } else {
           double l = n;
           double s = n;
           while(n!=0){
              if(l<n){
                 l=n;
              } else if (s>n){
                 s=n;
              }
              n = readDouble("?");
           }
           println("Smallest: " + s + "\nLargest: " + l);
        }


        readLine("\n\nPress any key to repeat program");
        getConsole().clear();
        run();
     }
     public static void main(String[] args) {
        new FindRange().start(args);
     }
}
```

c. **Broken Java**

```
package ph.edu.dlsu.chan.fixingbrokenjava;

/*
 * File: FixingBrokenJava.java
 * Name:
 * Section Leader:
 *
 * This program does not work as intended.  It contains both
 * compile-time errors (errors that prevent the compiler from even
 * running the program) and run-time errors (errors where the
 * program does not function as intended).  Your job is to fix
 * this program so that it works correctly.  Note that it is *not*
```

```
 * sufficient to simply fix the compiler errors; you will need to
 * update the logic as well.
 *
 * This program attempts to read a positive integer from the user,
 * then check whether that integer is prime (whether its only
 * divisors are 1 and itself).  If so, it prints a message saying
 * that the number is prime; otherwise it says that the number is
 * composite.
 */
import acm.graphics.*;
import acm.program.*;
/**
 *
 * @author Patrick Matthew Chan
 */
public class FixingBrokenJava extends ConsoleProgram{
	/* Reads a number from the user and reports whether or not it
	 * is prime.
	 */
	public void run() {
	/*System.out.println("getWidth() = " + getWidth());
	System.out.println("getHeight() = " + getHeight());
	System.out.println("getConsole().getWidth() = " +
getConsole().getWidth());
	System.out.println("getConsole().getHeight() = " +
getConsole().getHeight());*/


		/* Get the value from the user. */
		int value = readPositiveInt();

		/* Check whether or not it is prime. */
		if (isPrime(value)) {
			println(value + " is prime.");
		} else {
			println(value + " is composite.");
		}


	    readLine("\n\nPress any key to repeat program");
	    getConsole().clear();
	    run();
	}
```

```
        /**
         * Given a positive integer, returns whether that integer is
         * prime.
         *
         * @param value The value to test.
         * @return Whether or not it is prime.
         */
        private boolean isPrime(int value) {
                /* Try all possible divisors of the number.  If any of them
                 * cleanly divide the number, we return that the number is
                 * composite.
                 */
           if(value==1){
              return true;
           }
                for (int divisor = 2; divisor < value; divisor++) {
                        if (value % divisor == 0) {
                                return false;
                        }
                }
           return true;
        }

        /**
         * Reads a positive integer from the user and returns it.
         *
         * @return A positive integer entered by the user.
         */
        private int readPositiveInt() {
                /* Get an initial value. */
                int value = readInt("Enter a positive integer: ");

                /* If the value was nonpositive, reprompt the user. */
                while (value <= 0) {
                        println("Please enter a positive integer.");
                        value = readInt("Enter a positive integer: ");
                }

                return value;
        }
      public static void main(String[] args) {
         new FixingBrokenJava().start(args);
      }
}
```

d. Hailstone Sequence

```
package ph.edu.dlsu.chan.hailstone;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
import acm.graphics.*;
import acm.program.*;
/**
 *
 * @author Patrick Matthew Chan
 */
public class Hailstone extends ConsoleProgram{
   public void run(){
      int n = readInt("Enter a number: ");
      int count=0;
      while(n!=1){
         if(n%2==0){
            println(n + " is even, so I take half: " + n/2);
            n=n/2;
         } else {
            println(n + " is odd, so I take 3n+1: " + (3*n+1));
            n = 3*n+1;
         }
         count++;
      }
      println("The process took " + count + " steps to reach 1.");
      readLine("\n\nPress any key to repeat program");
      getConsole().clear();
      run();
   }
   public static void main(String[] args) {
      new Hailstone().start(args);
   }
}
```

e. Artistry!

```
/* © 2017 by Patrick Matthew Chan */
package ph.edu.dlsu.chan.artistry;
import acm.graphics.*;
import acm.io.*;
import acm.program.*;
import acm.util.*;
import java.io.*;
import java.applet.*;
import java.awt.event.*;
import java.awt.*;
import java.util.*;
import javax.swing.*;
/* @author Patrick Matthew J. Chan [LBYCP12-EQ1]*/
public class Artistry extends GraphicsProgram{
   //~~~~~~~~~~~~~~~~~~~~~~~~~~ Main Classes
~~~~~~~~~~~~~~~~~~~~~~~~~~//
   //main classes
   public static void main(String[] args) {
      new Artistry().start(args);
   }
   public void init(){
      ;
   }
   /*private double getXCenter(GObject g){
   return (getWidth() - g.getWidth())/2;
   }
   private double getYCenter(GObject g){
   return (getHeight() - g.getHeight())/2;
   }*/
   public void run(){
      setTitle("Artistry (Resizable)");
      setSize(500,500);
      removeAll();
      drawArt(getGCanvas());
      this.addComponentListener(new ComponentAdapter() {
         @Override
         public void componentResized(ComponentEvent e) {
            removeAll();
            drawArt(getGCanvas());
         }
```

```
    });
}


public void drawArt(GCanvas gc){
    myArt art=new myArt();
    art.scale(gc.getWidth()/art.getWidth(),
            gc.getHeight()/art.getHeight());
    gc.add(art);
    //setBackground(Color.GREEN);
    GLabel e=new GLabel ("Art by Patrick Chan");
    add(e, getWidth()-e.getWidth()-10, getHeight()-20);
}

public class myArt extends GCompound{
    public myArt(){
        GRect bnd=new GRect(1200,1200);
        bnd.setVisible(true);
        bnd.setFilled(true);
        bnd.setFillColor(Color.GREEN);
        bnd.setColor(Color.GREEN);
        add(bnd);
        double aThird=bnd.getWidth()/3;
        GRect o=new GRect(bnd.getWidth(),aThird/3);
        o.setFilled(true);
        o.setFillColor(Color.yellow);
        o.setColor(Color.yellow);
        add(o);
        GOval a = new GOval(5*aThird/6,5*aThird/6);
        a.setFilled(true);
        a.setColor(Color.RED);
        add(a,aThird/2,bnd.getHeight()/3-aThird/2);
        GOval b = new GOval(5*aThird/6,5*aThird/6);
        b.setFilled(true);
        b.setColor(Color.BLUE);
        add(b,bnd.getWidth()-b.getWidth()-aThird/2,
              bnd.getHeight()/3-aThird/2);
        GRect c = new GRect(aThird/2.5,aThird/2.5);
        c.setFilled(true);
        c.setColor(Color.GRAY);
        add(c,(bnd.getWidth()-c.getWidth())/2,
              (bnd.getHeight()-c.getHeight())/2+c.getHeight());
        GArc d = new GArc(aThird/2,aThird/2,100,150);
        d.setFilled(true);
```

```
            d.setColor(Color.ORANGE);
            add(d,(bnd.getWidth()-d.getWidth())/2,
                  c.getY()+c.getHeight()+aThird/3);
        }
    }

    //~~~~~~~~~~~~~~~~~~~~~~~~~ Debugging & Misc
~~~~~~~~~~~~~~~~~~~~~~~~//
    // <editor-fold defaultstate="collapsed" desc="p(),pl(),pel()">
    public static void p(Object a){//debug
        System.out.print(a+"");
    }
    public static void pl(Object a){//debug
        System.out.println(a+"");
    }
    public static void pl(){//debug
        System.out.println();
    }
    public static void pel(){//debug
        System.err.println();
    }
    // </editor-fold>
    //application size  //alternative method is setSize(x,y)
    public static final int APPLICATION_WIDTH = 500;
    public static final int APPLICATION_HEIGHT = 500;



    //~~~~~~~~~~~~~~~~~~~~~~~~~ Global Variables
~~~~~~~~~~~~~~~~~~~~~~~~//

    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Methods
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//

}
```

f. Target

```
package ph.edu.dlsu.chan.target;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
import java.awt.*;
import java.awt.event.*;
import acm.graphics.*;
import acm.program.*;
/**
 *
 * @author Patrick Matthew Chan
 */
public class Target extends GraphicsProgram{
   public static final int APPLICATION_WIDTH = 640;
   public static final int APPLICATION_HEIGHT = 480;
   private double getXCenter(GObject g){
      return (getWidth() - g.getWidth())/2;
   }
   private double getYCenter(GObject g){
      return (getHeight() - g.getHeight())/2;
   }
   public void run(){
      GOval a = new GOval(144,144);
      a.setFilled(true);
      a.setColor(Color.RED);
      add(a,getXCenter(a),getYCenter(a));
      GOval b = new GOval(144*.65,144*.65);
      b.setFilled(true);
      b.setColor(Color.WHITE);
      add(b,getXCenter(b),getYCenter(b));
      b.sendToFront();
      GOval c = new GOval(144*.3,144*.3);
      c.setFilled(true);
      c.setColor(Color.RED);
      add(c,getXCenter(c),getYCenter(c));
      c.sendToFront();
      //add(new GLabel ("Art by Patrick Chan"), 635, 460);
   }
   public static void main(String[] args) {
      new Target().start(args);
   }
}
```

g. Pyramid

```
package ph.edu.dlsu.chan.pyramid;

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
import java.awt.*;
import java.awt.event.*;
import acm.graphics.*;
import acm.program.*;
/**
 *
 * @author Patrick Matthew Chan
 */
public class Pyramid extends GraphicsProgram{
   public static final int APPLICATION_WIDTH = 640;
   public static final int APPLICATION_HEIGHT = 480;
   private double getXCenter(GObject g){
      return (getWidth() - g.getWidth())/2;
   }
   private double getYCenter(GObject g){
      return (getHeight() - g.getHeight())/2;
   }
   private double aaa(int bricks,GObject g){
      return (getWidth() - bricks*g.getWidth())/2;
   }
   public void run(){
      int BRICK_WIDTH=30;
      int BRICK_HEIGHT=12;
      int BRICKS_IN_BASE=14;
      //GRect a = new GRect(BRICK_WIDTH,BRICK_HEIGHT);
      //add(a,getXCenter(a),getHeight()-BRICK_HEIGHT);
      //add(a,getXCenter(a)-100,getHeight()-BRICK_HEIGHT);
      for(int j=BRICKS_IN_BASE;j!=0;j--){
         for(int i=1;i!=j+1;i++){
            GRect a = new GRect(BRICK_WIDTH,BRICK_HEIGHT);
```

```
add(a,aaa(j,a)+((i-1)*BRICK_WIDTH),getHeight()-(BRICK_HEIGHT*(BRICKS
_IN_BASE-j+1)));
          }
       }

   }
   public static void main(String[] args) {
      new Pyramid().start(args);
   }
}
```

h. Menu

i. MainFrame.java

```
/* © 2017 by Patrick Matthew Chan */
package ph.edu.dlsu.chan.menu;
import static acm.util.JTFTools.pause;
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import ph.edu.dlsu.chan.artistry.Artistry;
import ph.edu.dlsu.chan.findrange.FindRange;
import ph.edu.dlsu.chan.fixingbrokenjava.FixingBrokenJava;
import ph.edu.dlsu.chan.hailstone.Hailstone;
import ph.edu.dlsu.chan.pyramid.Pyramid;
import
ph.edu.dlsu.chan.pythagoreantheorem.PythagoreanTheorem;
import ph.edu.dlsu.chan.target.Target;
/* @author Patrick Matthew J. Chan [LBYCP12-EQ1]*/
public class MainFrame extends JFrame{
   //~~~~~~~~~~~~~~~~~~~~~~~~~~ Main Classes
~~~~~~~~~~~~~~~~~~~~~~~~~~//
   //main classes
   public void ini() {
      //set layout
      setLayout(new GridLayout(8,1,10,5));
      //set properties of child components
      B1.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            SubProgramCalled=1;
```

```
    }
});
B2.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        SubProgramCalled=2;
    }
});
B3.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        SubProgramCalled=3;
    }
});
B4.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        SubProgramCalled=4;
    }
});
B5.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        SubProgramCalled=5;
    }
});
B6.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        SubProgramCalled=6;
    }
});
B7.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        SubProgramCalled=7;
    }
});

//add components
add(L);
add(B1);
add(B2);
add(B3);
```

```
        add(B4);
        add(B5);
        add(B6);
        add(B7);
        //set properties of container
        setSize(400,700);
        setTitle("Menu");
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                isWindowClosed=true;
                dispose();
                System.exit(0);
            }
        });
        //set visibility to true
        setVisible(true);
        //etc
        setResizable(false);

getRootPane().setBorder(BorderFactory.createMatteBorder(0,40,20,
40,getBackground()));

        //main program
        while(!isWindowClosed){
            pause(0);
            switch(SubProgramCalled){
                case 1:
                    //objects
                    PythagoreanTheorem pyt=new PythagoreanTheorem();
                    JFrame f1=new JFrame(pyt.getTitle());
                    //app thread
                    Thread t1=new Thread(new Runnable() {
                        @Override
                        public void run() {
                            pyt.start(new String[0]);
                        }
                    });
                    t1.setPriority(Thread.MIN_PRIORITY+1);
                    //setup the frame
                    f1.setVisible(true);
                    f1.setSize(new
Dimension(ACM_FRAME_CONSOLE_WIDTH,ACM_FRAME_CONSOL
E_HEIGHT));
```

```
                f1.add(pyt);

f1.setMenuBar(pyt.getMenuBar().createOldStyleMenuBar());
                f1.addWindowListener(new WindowAdapter() {
                    @Override
                    public void windowClosing(WindowEvent e) {
                        super.windowClosing(e);
                        pyt.stop();
                        f1.dispose();
                    }
                });
//f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                f1.validate();
                //start thread
                t1.start();
                SubProgramCalled=-1;
                break;
            case 2:
                //objects
                FindRange fin=new FindRange();
                JFrame f2=new JFrame(fin.getTitle());
                //app thread
                Thread t2=new Thread(new Runnable() {
                    @Override
                    public void run() {
                        fin.start(new String[0]);
                    }
                });
                t2.setPriority(Thread.MIN_PRIORITY+1);
                //setup the frame
                f2.setVisible(true);
                f2.setSize(new
Dimension(ACM_FRAME_CONSOLE_WIDTH,ACM_FRAME_CONSOL
E_HEIGHT));
                f2.add(fin);

f2.setMenuBar(fin.getMenuBar().createOldStyleMenuBar());
                f2.addWindowListener(new WindowAdapter() {
                    @Override
                    public void windowClosing(WindowEvent e) {
                        super.windowClosing(e);
                        fin.stop();
                        f2.dispose();
                    }
```

```
                });
//f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                f2.validate();
                //start thread
                t2.start();
                SubProgramCalled=-1;
                break;
            case 3:
                //objects
                FixingBrokenJava fix=new FixingBrokenJava();
                JFrame f3=new JFrame(fix.getTitle());
                //app thread
                Thread t3=new Thread(new Runnable() {
                    @Override
                    public void run() {
                        fix.start(new String[0]);
                    }
                });
                t3.setPriority(Thread.MIN_PRIORITY+1);
                //setup the frame
                f3.setVisible(true);
                f3.setSize(new
Dimension(ACM_FRAME_CONSOLE_WIDTH,ACM_FRAME_CONSOL
E_HEIGHT));
                f3.add(fix);

f3.setMenuBar(fix.getMenuBar().createOldStyleMenuBar());
                f3.addWindowListener(new WindowAdapter() {
                    @Override
                    public void windowClosing(WindowEvent e) {
                        super.windowClosing(e);
                        fix.stop();
                        f3.dispose();
                    }
                });
//f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                f3.validate();
                //start thread
                t3.start();
                SubProgramCalled=-1;
                break;
            case 4:
                //objects
                Hailstone ha=new Hailstone();
```

```
                JFrame f4=new JFrame(ha.getTitle());
                //app thread
                Thread t4=new Thread(new Runnable() {
                    @Override
                    public void run() {
                        ha.start(new String[0]);
                    }
                });
                t4.setPriority(Thread.MIN_PRIORITY+1);
                //setup the frame
                f4.setVisible(true);
                f4.setSize(new
Dimension(ACM_FRAME_CONSOLE_WIDTH,ACM_FRAME_CONSOL
E_HEIGHT));
                f4.add(ha);

f4.setMenuBar(ha.getMenuBar().createOldStyleMenuBar());
                f4.addWindowListener(new WindowAdapter() {
                    @Override
                    public void windowClosing(WindowEvent e) {
                        super.windowClosing(e);
                        ha.stop();
                        f4.dispose();
                    }
                });
//f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                f4.validate();
                //start thread
                t4.start();
                SubProgramCalled=-1;
                break;
            case 5:
                //objects
                Artistry ar=new Artistry();
                JFrame f5=new JFrame("Artistry (Resizable)");
                //app thread
                Thread t5=new Thread(new Runnable() {
                    @Override
                    public void run() {
                        ar.start(new String[0]);
                    }
                });
                t5.setPriority(Thread.MIN_PRIORITY+1);
                //setup the frame
```

```
                f5.setVisible(true);
                f5.setSize(new Dimension(ar.APPLICATION_WIDTH +
ACM_FRAME_OFFSET_X, ar.APPLICATION_HEIGHT +
ACM_FRAME_OFFSET_Y));
                f5.add(ar);

f5.setMenuBar(ar.getMenuBar().createOldStyleMenuBar());
                f5.addWindowListener(new WindowAdapter() {
                   @Override
                   public void windowClosing(WindowEvent e) {
                      super.windowClosing(e);
                      ar.stop();
                      f5.dispose();
                   }
                });
//f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                f5.validate();
                //start thread
                t5.start();
                SubProgramCalled=-1;
                break;
             case 6:
                //objects
                Target ta=new Target();
                JFrame f6=new JFrame(ta.getTitle());
                //app thread
                Thread t6=new Thread(new Runnable() {
                   @Override
                   public void run() {
                      ta.start(new String[0]);
                   }
                });
                t6.setPriority(Thread.MIN_PRIORITY+1);
                //setup the frame
                f6.setVisible(true);
                f6.setSize(new Dimension(ta.APPLICATION_WIDTH +
ACM_FRAME_OFFSET_X, ta.APPLICATION_HEIGHT +
ACM_FRAME_OFFSET_Y));
                f6.add(ta);

f6.setMenuBar(ta.getMenuBar().createOldStyleMenuBar());
                f6.addWindowListener(new WindowAdapter() {
                   @Override
                   public void windowClosing(WindowEvent e) {
```

```
                    super.windowClosing(e);
                    ta.stop();
                    f6.dispose();
                }
            });
//f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            f6.validate();
            //start thread
            t6.start();
            SubProgramCalled=-1;
            break;
        case 7:
            //objects
            Pyramid pyr=new Pyramid();
            JFrame f7=new JFrame(pyr.getTitle());
            //app thread
            Thread t7=new Thread(new Runnable() {
                @Override
                public void run() {
                    pyr.start(new String[0]);
                }
            });
            t7.setPriority(Thread.MIN_PRIORITY+1);
            //setup the frame
            f7.setVisible(true);
            f7.setSize(new Dimension(pyr.APPLICATION_WIDTH +
ACM_FRAME_OFFSET_X, pyr.APPLICATION_HEIGHT +
ACM_FRAME_OFFSET_Y));
            f7.add(pyr);

f7.setMenuBar(pyr.getMenuBar().createOldStyleMenuBar());
            f7.addWindowListener(new WindowAdapter() {
                @Override
                public void windowClosing(WindowEvent e) {
                    super.windowClosing(e);
                    pyr.stop();
                    f7.dispose();
                }
            });
//f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            f7.validate();
            //start thread
            t7.start();
            SubProgramCalled=-1;
```

```
                break;
            default:
                if(SubProgramCalled!=-1){
                    System.out.println("(no such subprogram)
SubProgramCalled = " + SubProgramCalled);
                    SubProgramCalled=-1;
                }
                break;
          }
        }

    }
    //~~~~~~~~~~~~~~~~~~~~~~~~ Debugging & Misc
~~~~~~~~~~~~~~~~~~~~~~~~//
    // <editor-fold defaultstate="collapsed" desc="p(),pl(),pel()">
    public static void p(Object a){//debug
        System.out.print(a+"");
    }
    public static void pl(Object a){//debug
        System.out.println(a+"");
    }
    public static void pl(){//debug
        System.out.println();
    }
    public static void pel(){//debug
        System.err.println();
    }
    // </editor-fold>
    //application size  //alternative method is setSize(x,y)
    //public static final int APPLICATION_WIDTH = 400;
    //public static final int APPLICATION_HEIGHT = 650;



    //~~~~~~~~~~~~~~~~~~~~~~~~ Global Variables
~~~~~~~~~~~~~~~~~~~~~~~~//
    private static final long serialVersionUID = 1L;//la lang
    JLabel L = new JLabel("Sub-activities:");
    JButton B1 = new JButton("Pythagorean Theorem");
    JButton B2 = new JButton("Find Range");
    JButton B3 = new JButton("Fixing Broken Java");
    JButton B4 = new JButton("Hailstone");
    JButton B5 = new JButton("Artistry");
    JButton B6 = new JButton("Target");
```

```
   JButton B7 = new JButton("Pyramid");

   private int SubProgramCalled =-1;
   private boolean isWindowClosed=false;

   public static final int ACM_FRAME_OFFSET_X = 16;
   public static final int ACM_FRAME_OFFSET_Y = 52;
   public static final int ACM_FRAME_CONSOLE_OFFSET_Y = 59;
   public static final int
ACM_FRAME_CONSOLE_WIDTH=754+ACM_FRAME_OFFSET_X;
   public static final int
ACM_FRAME_CONSOLE_HEIGHT=472+ACM_FRAME_CONSOLE_OF
FSET_Y;
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Methods
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//

}
```

ii. Menu.java

```
/* © 2017 by Patrick Matthew Chan */
package ph.edu.dlsu.chan.menu;
/* @author Patrick Matthew J. Chan [LBYCP12-EQ1]*/
public class Menu{
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Main Classes
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//
   //main classes
   public static void main(String[] args) {
      MainFrame m=new MainFrame();
      m.ini();
   }
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Debugging & Misc
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//
   // <editor-fold defaultstate="collapsed" desc="p(),pl(),pel()">
   public static void p(Object a){//debug
      System.out.print(a+"");
   }
   public static void pl(Object a){//debug
      System.out.println(a+"");
   }
   public static void pl(){//debug
```

```
        System.out.println();
    }
    public static void pel(){//debug
        System.err.println();
    }
    // </editor-fold>
    //application size  //alternative method is setSize(x,y)
    //public static final int APPLICATION_WIDTH = 400;
    //public static final int APPLICATION_HEIGHT = 650;

    //~~~~~~~~~~~~~~~~~~~~~~~~~ Global Variables
~~~~~~~~~~~~~~~~~~~~~~~~~//

    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Methods
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~//

}
```

REFERENCES

1. E Roberts. *Art and Science of Java*. Pearson; 2013.
2. E Roberts, M Sahami, and M Stepp, *CS 106A: Programming Methodology (Java) Handouts*, Stanford University.