Programming Languages
Homework 1

- Any modifications or clarifications will be in RED below. Submit your entire program in one .java file. Do not submit .class files. Do submit your report that shows your program works as expected. The purpose of this homework is to build a Java class hierarchy that uses polymorphism. We will hand-translate this into C in the next homework assignment. This will help you understand how virtual functions are implemented very efficiently in compiled languages like C++ and Java.
- Use Java to write the following program (I added some starts below for folks who have no experience with Java). Here is a link for how to compile and run a Java program on Linux (openlab.ics.uci.edu) using the commands **javac** (to compile) and **java** (to run). Define a class hierarchy for Shapes. Define a base class Shape with methods to draw(), which draws this shape, and area(), which computes and returns the area of this shape:

    - (20 points) Define a base class, called *Shape*, with two methods (double *area()* and void *draw()*). Be sure this class has a constructor which takes parameters to initialize the data members. draw() should print out the name of the specific Shape to help us see what is going on. Be sure area returns a double as integer would not be precise enough for practical use. Here is a start

```
class Shape
{
   String name;
   Shape(String newName)
   {
        name = newName;
   }
   double area()
   {
      return 0.0;
   }
   void draw()
   {
        System.out.println("Shape.draw() You should never see this.");
        // note if you copy&paste from the Internet, double quotes are Unicode, not ASCII
   }
}
```

    - (40 points) Derive a total of four classes from Shape: derive three classes (*Circle*, *Square*, and *Triangle*) directly from your base class and make them concrete by providing implementations for the virtual functions introduced in your base class *Shape*. The methods for each of these three classes must actually behave differently from one another (example: Circle area vs.

Square area vs. Triangle area).

```
class Triangle
   extends Shape
{
   int myHeight, myBase;
   Triangle(String name, int h, int b)
   {
      super(name);
      myHeight = h;
      myBase = b;
   }
}
```

Derive the fourth class, *Rectangle*, from *Square* and add a width. For *draw()* use character graphics. The shapes should resemble the shape defined by the class.

- (20 points) Write a class *Picture* that holds a list of Shapes. Write a method, called add(Shape sh) that adds the shape referred to by sh to this picture. Also define two polymorphic methods that operate on a Picture: void *drawAll()* and double *totalArea()*. Implement Picture as a LinkedList of Shapes.

```
class Picture
{
   void add(Shape sh)
   {
   }
   void drawAll()
   {
   }
   double totalArea()
   {
      return 0.0;
   }
}
```

- (20 points) Write a main program in a public class named mainClass, in a file called mainClass.java, that builds a Picture and fills it with two triangles, two circles, two squares, and two rectangles (Specified below). Then have it call drawAll(), then have it print out the totalArea() of the shapes on that picture. For the dimensions on each object, the autograder will pass you two numbers. For all objects that only require one dimension to build (ie: circle), just use the first argument. For the rest of the objects ensure that you use both dimensions (example: java MyProgram 2 3 -> FirstTriangle(2, 3)). Your first shape of any type should use the supplied dimension *n* while the second shape of the same object will use the dimension *n-1*. (eg: TriangleOne(4,5), TriangleTwo(3,4))

```
public class mainClass
{
   static void println(double d)
   {
        System.out.println(“Double d is “ + d);
   }
   public static void main(String[] args)
   {
      Picture p = new Picture();
      p.add(new Triangle(“FirstTriangle”, 5, 5));
      p.add(new Triangle(“SecondTriangle”, 4, 3));
      p.add(new Circle(“FirstCircle”, 5));
      ….
      p.printShapes(); // this could print out the first output shown below, for your benefit
      println(p.totalArea());
      p.drawAll();
   }
}
```

You can compile this program
javac mainClass.java
You can run the program with this command
java mainClass arg1 arg2
(where arg1 and agr2 should be any integer you wish)

Here is sample output for the program. We dont care what order you print the shapes and their areas in, but make sure you use the exact format for naming below to ensure the autograder grades you correctly:

```
FirstTriangle(5, 5) : 12.5
SecondTriangle(4, 4) : 8

FirstCircle(5) : 78.54
SecondCircle(4) : 50.27

FirstSquare(5) : 25
SecondSquare(4) : 16

FirstRectangle(5, 5) : 25
SecondRectangle(4, 4) : 16

Total : 231.31
```

/* various pictures of the shapes omitted because they large */

<u>HW1 Specifics:</u>

The output shapes may be simple and unimpressive. Additionally, the image can be the same for each square, each circle, etc. in that it need not scale to the size of the shape. Do not use any external libraries for the drawing graphics, simple character graphics is what we want. You can still draw almost standard shapes with character graphics, but circle will be the difficult shape. E.g, a rectangle

```
*********
*       *
*       *
*       *
*********
```

or

```
----------
|        |
|        |
|        |
----------
```

What to submit: You should submit one file:

1. To Gradescope, submit one file, mainClass.java, containing your entire program. The autograder will compile your file using Java 8 (same as the openlab version) so be aware of that when building your Java application.