COMP1006/1406 – Fall 2023

Submit your `.java` files to Gradescope.

The assignment is out of 10.

The assignment has two parts. While the due date for the entire assignment is Friday, October 13th, the intention, or my *hope*, is that part one should be completed by Friday, October 6th, and that part two should be completed by Friday, October 13th.

Start early!
Submit often.

This assignment will **mostly** be graded for correctness.
9/10 of your grade will be based on correctness so be sure that you test your code!
The remaining mark will be determined by a manual inspection to ensure that your code did not violate any restrictions specified in this document (or in the skeleton files provided).

There is a 48-hour grace period in which submissions will be accepted without penalty. If need be, you can submit this assignment up to Sunday, October 15th, at 11:59pm without any penalty. However, there will not be any office hours and no guarantees that questions will be answered on discord over the weekend. Be sure to start early and submit often.

Do NOT change the input types, output types or modifiers of the methods you are completing. If you change any of these, it will not compile when submitted.

Part One

Temperature

Complete the provided `Temperature` class. Add any attributes and helper methods as needed but keep in mind that testing will involve only the methods you are asked to write/complete. You must complete the constructors and methods in the provided class (**without changing any signatures, return types, or modifiers**).

A *temperature* consists of a value (magnitude) and a scale. For example, if the temperature is $-17.4°C$, then its *value* is -17.4 and its *scale* is Celsius. The valid scales that we will consider for our `Temperature` objects will be Celsius, Fahrenheit or Kelvin. Once a scale has been set, a `Temperature` object will always display its temperature in that scale until the scale is changed. The default scale is Celsius if not specified.

In this problem you will need to be able to convert temperatures between Celsius, Fahrenheit and Kelvin. For help, see `https://en.wikipedia.org/wiki/Conversion_of_units_of_temperature`

The three scales are represented by Strings in the provided `Scale` class (class attributes). For this assignment, the purpose of the `Scale` class is to provide a consistent naming scheme for the different scales. Essentially, we assign fixed names for the three scales and use these everywhere in the code (so that a simple spelling mistake in your code does not result in failing every test case or crashing code that used your class).

Some examples of using a Temperature object:

```
Temperature t = new Temperature(10.1);
System.out.println(t.getScale());     // displays "CELSIUS"
System.out.println(t.toString());     // displays "10.1C"
t.setScale(Scale.FAHRENHEIT);         // change scale
System.out.println(t.toString());     // displays "50.18F"  (notice it converted the value!)
System.out.println(t.getScale());     // displays "FAHRENHEIT"
t = new Temperature(12.25, "KELVIN"); // scale input is not from Scale!
System.out.println(t.getScale());     // displays "NONE" because of previous bad setScale()
System.out.println(t.toString());     // displays "0.0N" because of previous bad setScale()
```

Note: You must provide your own **state** (instance attributes) for the `Temperature` class. You must decide what state to store for this problem. But, you must only use instance attributes. You should have no static attributes in your class.

Note: Temperature values are floating point numbers. If an object's value is expected to be `0.1` and your output (from `getValue()` is `0.09999999999998`), that is OK. You are **not** asked to perform any rounding in the `getValue()` method.

Note: The provided `toString()` method will always display your `Temperature` objects using three (3) decimal places. Rounding *will* happen automatically here in the String output.

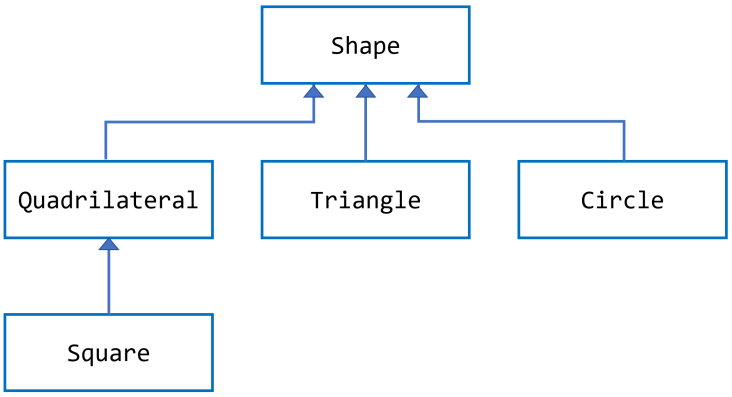
Note: A program called `SimpleTemperatureProgram` is provided with the code shown above that you can use as a starting point for your own testing if you wish.

Include your `Temperature.java` file when you submit.

Part Two

Shapes

In this problem, you are provided with a `Shape` class and will need to create the other four classes shown in the following class hierarchy.



Each shape object has one (x,y) coordinate that *anchors* it in the x-y plane. This anchor point is fixed and cannot change once an object is created. The other parameters can change though. The provided `XYCoord` class is used to store (x,y) coordinates. All shapes must be able to compute their own area and perimeter[1] in addition to specific methods (outlined below). You are free to add any *state* to your classes as needed but you should not be storing any information in a given object twice.

Each of the classes will have a constructor that is appropriate for the particular shape detailed as follows:

```
public Quadrilateral(XYCoord a, XYCoord b, XYCoord c, XYCoord d)
  // a quadrilateral has four straight sides and four corner points (vertices)
  // the corner points are specified by the inputs a,b,c,d
  // a is the anchor coordinate for this shape.

public Square(XYCoord anchor, double length)
  // a square is a special quadrilateral (all sides have the same
  // length and the angle at each corner is 90 degrees).
  // the anchor is the bottom-left corner of the square and length >= 0

public Triangle(XYCoord a, XYCoord b, XYCoord c)
  // a,b,c are the three coordinates of the corners (vertices) of the triangle
  // a is the anchor coordinate for this shape

public Circle(XYCoord centre, double radius)
  // centre is the anchor coordinate and radius >= 0
```

[1]The word circumference is usually used circle and shapes with curves but we will use perimeter for everything here to be consistent.

You will also need to create the following *setter* methods in the specified classes. Note that there is no setter for the `Quadrilateral` class.:

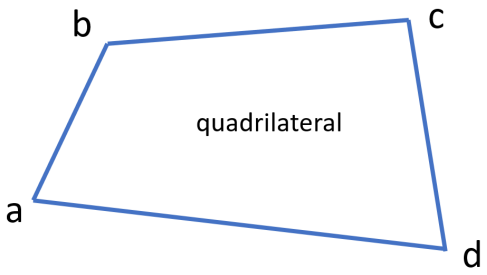
```
Circle Class
public void setRadius(double newRadius)
  // changes the object's radius to be newRadius

Triangle Class
public void setBC(XYCoord newB, XYCoord newC)
  // changes the object's b and c points to be newB and newC

Square Class
public void setLength(double newLength)
  // changes the object's length to be newLength
```

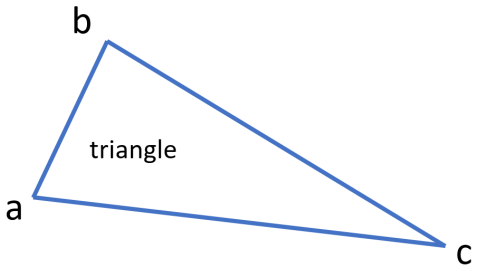
You can use the provided `ShapeExampleApp` to help test your code. Do NOT change or submit the provided `Shape.java` file. Do NOT change or submit the provided `XYCoord.java` file. When testing, we will use the files as provided (and delete any version you submit).

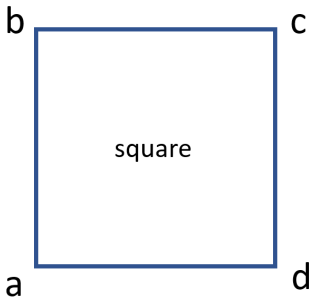
The area for the square and circle should be straightforward to compute. For the quadrilateral, note that you can think of this shape as being composed of two triangles next to each other. For a triangle, you might find the following resource helpful: `https://byjus.com/maths/area-of-a-triangle/`

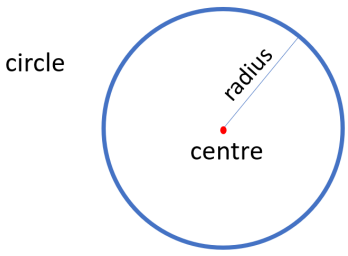
Include your `Quadrilateral.java`, `Square.java`, `Triangle.java` and `Circle.java` files when you submit.

Note that the shapes will always be created with coordinates as shown below. That is, the ordering of the coordinates will correspond to the labelling on the diagrams.









1) You should have no static attributes in any of your classes.
2) Read the specifications in the skeleton files carefully if provided.
3) Be sure to use **INFORMATION HIDING.**

Part One: Do **not** use Strings that *look* like the attributes from `Scale`. Use the actual static constants provided.

Part Two: The classes must use inheritance that follows the UML diagram provided!