# CS 171: Introduction to Computer Science II
# Assignment #1: OOP and Shape Representations

**Due: Monday Sept 14 at 11:59 PM on Canvas [Late Submission: See Syllabus for policy details.]**

**Submission Instructions:** This programming assignment should be submitted electronically on Canvas. Make sure that all Java files you submit can be compiled and run against Java Development Kit 8 (or higher).

**Goals:** This assignment is designed to familiarize you with Object-Oriented Programming (OOP) concepts using the topic of *shapes*. You will become more familiar with writing Java classes and implementing instance methods that update or access instance variables. You will also apply method overloading, method overriding, and class inheritance. These OOP concepts are extremely important, practical, and fun to play around with. Enjoy!

**!! Important !!** Your method headers (i.e., return type, method name, and parameter list) should match the **exact** description in the handout and starter code, including the **case sensitivity** of any method's name; otherwise, our testing scripts will fail and you risk getting 0 points.

## Part 1: Complete these simple Java classes

Your first task is to complete the implementation of the following classes that represent different shapes, by filling in the missing methods in each class. For some methods you will need to either complete or define the signature of the method as well. Pay careful attention to the expected *name* of the method, the order and type of *parameters*, and the method's *return type*, since an incorrect method signature will cause our testing code to fail when running against your submission.

Note that the missing methods in the code are marked by a `TODO` prefix. Make sure to remove the `TODO` tags once you're done coding and testing your implementation. This is common practice when you are coding a problem in the real world. Complete all missing code in the following classes:

- Class **Circle** is defined inside Circle.java and is meant to represent a circle shape.
- Class **Rectangle** is defined inside Rectangle.java and is meant to represent a rectangle shape.
- Class **ShapeTester** is defined inside ShapeTester.java and is meant to compare different objects (i.e. instances) of shapes.

## Part 2: Implement object inheritance in Java

It is often the case that the classes you write to represent real-world data types will inherit properties (data and methods) from other classes. In this assignment, your second task is to write a new class called **Cylinder** and save it in a file named Cylinder.java. Class Cylinder must `extend` the class Circle and satisfy the following requirements:

- The center of the cylinder should have an `x` coordinate, a `y` coordinate, and the `z` coordinate should be 0. It should also have a `height`, which is how tall the cylinder is in the z direction. All coordinates and dimensions should be of type `double`. Think carefully about which variables you will inherit from class Circle and, therefore, which new ones will you need to define in class Cylinder.

- There should be two constructors in class Cylinder. The first one takes no arguments and sets the center of the base at (0.0,0.0,0.0), the radius of the base at 1, and the height at 1. The second constructor takes as input the x and y coordinates, the radius of the cylinder, and the height of the cylinder.
- Your class should define the methods `getHeight` (which returns the cylinder height), and `setHeight` (which updates the cylinder height).
- Your class should override the `getArea` method from class Circle to make it return the area of the Cylinder object instead. Your method implementation here must utilize the getArea computation already defined in the super class Circle. (You may invoke other methods from the super class Circle as well, if needed.)
- Your class should define the method `getVolume` which returns the volume of the Cylinder object (as a `double` value).

### Testing and submitting your work

You can define a main method in each class you complete or write, in order to test the different methods you have implemented in that class. You should try different test cases and troubleshoot your code using inputs you know the answers to. Your code will be assessed by a main method that you do not have access to. Remember that your methods must have signatures (headers) that **match exactly the provided descriptions**, otherwise our testing code will not be able to call your methods correctly, resulting in a zero grade.

The Circle class has the most information filled in, followed by the Rectangle class. The ShapeTester class only has method descriptions, and it is up to you to write the Cylinder class. To submit this assignment, you must upload four files on Canvas: **Circle.java**, **Rectangle.java**, **ShapeTester.java**, and **Cylinder.java**. Make sure all files compile and run successfully before you submit them.

**Discussion and asking for clarifications:** Use our course Piazza page to post questions about the assignment if you need a clarification about any of the methods. Do **not** post solutions!

**Grading:** If a program does not compile, you will get 0 points for it. Programs that do compile successfully will be executed with different test cases. The breakdown of marks is as follows (detailed breakdown of the marks is available inside the starter code files):

- Class Circle correctness [20 points]
- Class Rectangle correctness [20 points]
- Class ShapeTester correctness [20 points]
- Class Cylinder correctness [35 points: 8 points for proper class and instance variables declaration, 6 points for constructors, 12 points for getArea, 9 points for remaining methods]
- Code clarity and style (add meaningful comments when appropriate!) [5 points]

**Honor Code:** Solve this assignment individually; do not collaborate with classmates or look for online solutions. We check for code plagiarism. The assignment is governed by the College Honor Code and Departmental Policy. Remember, any code you submit must be your own; otherwise you risk being investigated by the Honor Council and facing the consequences of that. Finally, please remember to have the following comment included at the top of the file:

```
/*
THIS CODE WAS MY OWN WORK, IT WAS WRITTEN WITHOUT CONSULTING
CODE WRITTEN BY OTHER STUDENTS OR COPIED FROM ONLINE RESOURCES. _Your_Name_Here_
*/
```