North South University
Department of Electrical and Computer Engineering

CSE 215 (Both Theory and Lab): Assignment 03 (Problem Set)
Course Instructor: Dr. Mohammad Rezwanul Huq

Problem Solving on Class, Interface, Files and Exception

Important Instructions:

a. Class name of the solution of the problems should be like Problem1, Problem2 and so on.

b. Create a zip/rar file which contains the solution **(.java file only)** of the problems only. As an example, only include Problem1.java, Problem2.java and so on.

c. The file name must follow the format:
<section number>_<assignment number>_<student number>.
Example: 13_01_1710000042. Files with incorrect naming will not be accepted.

d. Use the link below to upload the zip/rar file.
http://bit.ly/cse215codesubmit

Submission Deadline:

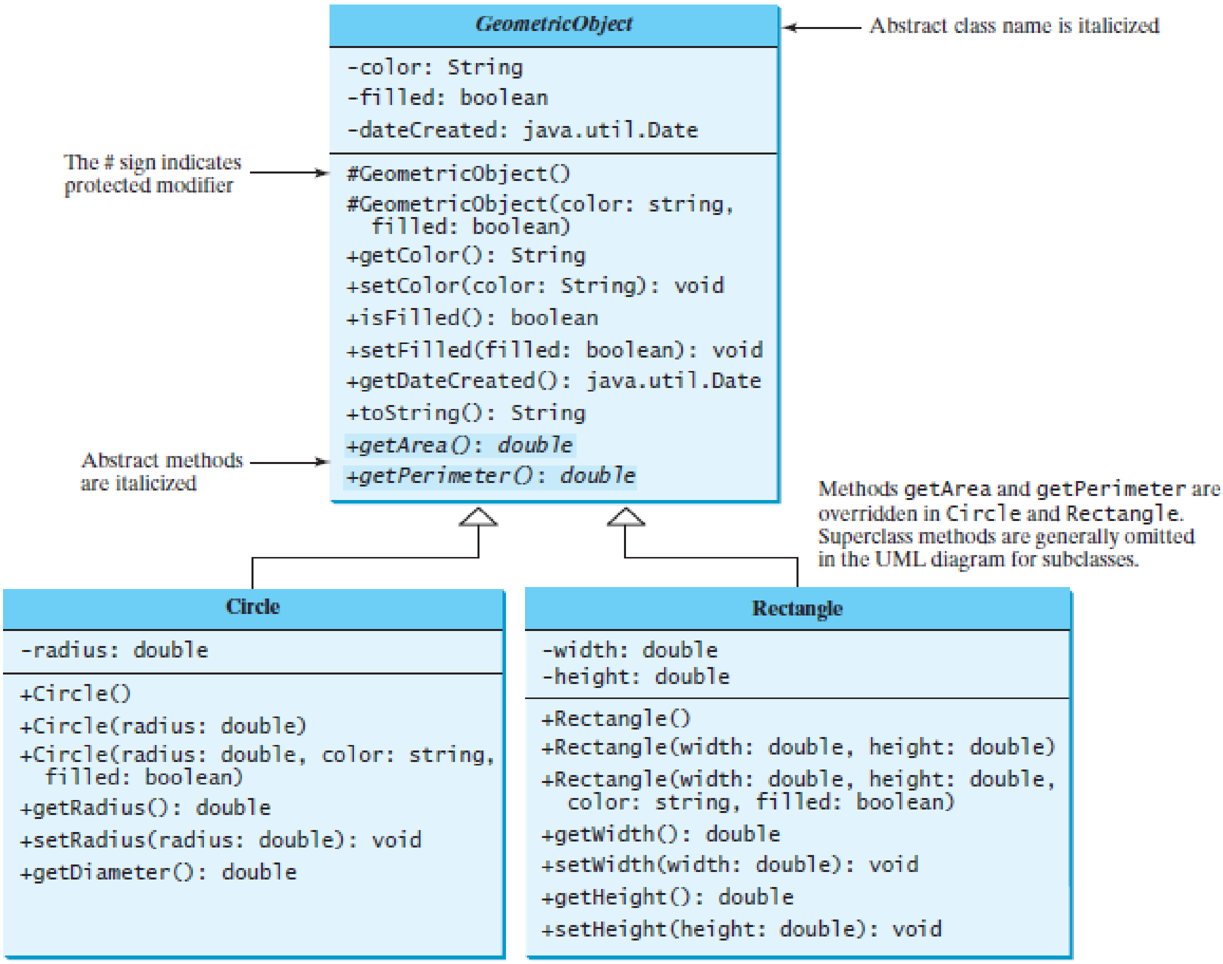
05 September 2019 11:59PM.

Problem Set

A. Class, Inheritance and Interface

1. (**Abstract Superclass and Concrete Subclasses**) Design the classes as per the following UML diagram.

   Write a test program that creates a Circle and another Rectangle object. Define the following method so that it determines which one of these two objects has larger area. The following method returns 1 if the first object is larger than the second, 0 if two are equal and -1 otherwise.
   public static int compareGeometricObject (GeometricObject ob1, GeometricObject ob2)


GeometricObject
-color: String
-filled: boolean
-dateCreated: java.util.Date
#GeometricObject()
#GeometricObject(color: string, filled: boolean)
+getColor(): String
+setColor(color: String): void
+isFilled(): boolean
+setFilled(filled: boolean): void
+getDateCreated(): java.util.Date
+toString(): String
+getArea(): double
+getPerimeter(): double
Abstract class name is italicized
The # sign indicates protected modifier
Abstract methods are italicized
Methods getArea and getPerimeter are overridden in Circle and Rectangle. Superclass methods are generally omitted in the UML diagram for subclasses.
Circle
-radius: double
+Circle()
+Circle(radius: double)
+Circle(radius: double, color: string, filled: boolean)
+getRadius(): double
+setRadius(radius: double): void
+getDiameter(): double
Rectangle
-width: double
-height: double
+Rectangle()
+Rectangle(width: double, height: double)
+Rectangle(width: double, height: double, color: string, filled: boolean)
+getWidth(): double
+setWidth(width: double): void
+getHeight(): double
+setHeight(height: double): void

2. (**Triangle class**) Design a new **Triangle** class that extends the abstract **GeometricObject** class as shown in Problem 1.

   Write a test program that prompts the user to enter three sides of the triangle, a color, and a Boolean value to indicate whether the triangle is filled. The program should create a **Triangle** object with these sides and set the color and filled properties using the input. The program should display the area, perimeter, color, and true or false to indicate whether it is filled or not.

3. (Enable **GeometricObject** comparable) Modify the **GeometricObject** class to implement the **Comparable** interface, and define a static **max** method in the **GeometricObject** class for finding the larger of two **GeometricObject** objects.

   Write a test program that uses the **max** method to find the larger of two circles and the larger of two rectangles.

4. (The **ComparableCircle** class) Define a class named **ComparableCircle** that extends **Circle** and implements **Comparable**. Implement the **compareTo** method to compare the circles on the basis of area.

   Write a test class to find the larger of two instances of **ComparableCircle** objects.

5. (The **Colorable** interface) Design an interface named **Colorable** with a **void** method named **howToColor()**. Every class of a colorable object must implement the **Colorable** interface. Design a class named **Square** that extends **GeometricObject** and implements **Colorable**. Implement **howToColor** to display the message **Color all four sides**.

Write a test program that creates an array of five **GeometricObjects**. For each object in the array, display its area and invoke its **howToColor** method if it is colorable.

B. Exception Handling

6. (**ArrayIndexOutOfBoundsException**) Write a program that meets the following requirements:
   ■ Creates an array with **100** randomly chosen integers.
   ■ Prompts the user to enter the index of the array, then displays the corresponding element value. If the specified index is out of bounds, display the message **Out of Bounds**.

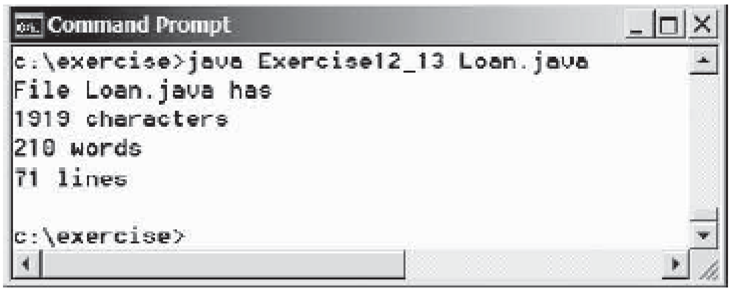
7. (**NumberFormatException**) Write a program, Calculator.java, which is a simple command line calculator. Note that the program terminates if any operand is nonnumeric. Write a program with an exception handler that deals with nonnumeric operands; then write another program without using an exception handler to achieve the same objective. Your program should display a message that informs the user of the wrong operand type before exiting.

8. (**Custom Exception Class**) Define a custom (user-defined) exception class named **InvalidRadiusException** that is raised when an instance of a Circle (see problem 1) has negative radius.

   Write a test program that creates a few instances (objects) of Circle type and throws **InvalidRadiusException** by incorporating appropriate try-catch-finally block.

C. File Handling

9. (**Count characters, words, and lines in a file**) Write a program that will count the number of characters, words, and lines in a file. Words are separated by whitespace characters. The file name should be passed as a command-line argument, as shown in the following figure.



```
c:\exercise>java Exercise12_13 Loan.java
File Loan.java has
1919 characters
210 words
71 lines

c:\exercise>
```

10. (**Process large dataset**) Consider that a file named **salary.txt** contains employees' information and salary of a university.

    Each line in the file consists of a faculty member's first name, last name, rank, and salary separated by a space (see Programming Exercise 12.24 in the textbook). Write a program to display the total salary for assistant professors, associate professors, full professors, and all faculty, respectively, and display the average salary for assistant professors, associate professors, full professors, and all faculty, respectively. You must define **Faculty** class and **appropriate array list** to store the data from the file.