CSC171 — Homework 6

The goal of this assignment is to give you additional experience with objects and object-oriented programming.

1. Suppose you're writing a blogging application. Define a class `Article` representing articles posted to the blog.

    - Articles have the following properties: author name, date posted, text of the article, and number of "likes". Your class should have instance variables of appropriate types with getters and setters for these properties. Use `java.util.Date` for the date (you can look it up in the javadoc).
    - Your class should have two constructors: one takes both the author's name and the posting date, the other takes only the author's name and uses the current date for the posting date.
    - Your class should have methods `like` and `unlike` that increment and decrement, respectively, the number of likes.
    - Your class should have an informative `toString` method.

    Include a separate class that tests the use of your `Article` class by creating a few instances, manipulating them, and printing informative messages about them.

2. Suppose you're working on payroll system. Define a class `Employee` representing employees of a company.

    - Employees have the following properties: first name, last name, id number, and salary. Your class should have instance variables of appropriate types with getters and setters for these properties.
    - Your class should have two constructors: one takes all the properties, one takes only first and last names and sets the id number and salary to some reasonable default.
    - Your class should have a method `raise` that increments the employee's salary by the given amount (decrements if the amount is negative).
    - Your class should have an informative `toString` method.

    Include a separate class that tests the use of your `Employee` class, including giving an employee a 10% raise.

3. Write a program that uses an instance of the `java.util.Random` class to create and print a random integer between 1 and 100. Then use the same instance to create two random `double` values and save them in variables named `mu` and `sigma`. Then use these two values to create a Gaussian ("normally") distributed `double` value and print all three `double`s. Hint: If you have a gaussian with mean 0.0 and standard deviation 1.0, you simply scale by the standard deviation σ (`sigma`) and offet by the mean μ (`mu`).

4. Write a program that reads two strings (which may include spaces) from the user. Your program should then test whether the first string is equal to, starts with, ends with, or otherwise contains the second string. Note: The `java.lang.String` class has all the methods you need.

5. Java's `System.out` is an instance of the `java.io.PrintStream` class. This class has a very useful method: `printf`. This method takes multiple arguments. The first is always a string, called the *format string*, which specifies how the remaining arguments are to be printed. Each occurrence of a `%` symbol in the format string gets replaced by something, usually one of the arguments, whose format depends on what follows the `%` in the format string. You do not need to know all the details of format strings, but you should understand how it works.

   Using the documentation of the method and the format string syntax (linked above if needed), and write a program that does the following:

   - Read an integer and a double value and print them separated by a space in one call to `System.out.printf` (see `%d` and `%f`)
   - Print the value of `Math.PI` to three decimal places (see "precision" specifier)
   - Read a double value and print it in the US style with commas in it (e.g., value 123456 prints as "123,456") (see "`,`" flag)

   End each printed line with a newline (see `%n`). The Java Tutorial has a good short overview of the basics, and the textbook has further examples of formatted output.

Grading Scheme

Equal weight for each part.

Doesn't compile or is trivial	$< 50\%$
Compiles and is non-trivial	$\geq 50\%$
Complete and correct with good style and comments	100%
Incomplete, incorrect, bad style, no comments	$< 100\%$

Submission Requirements

Your submission **MUST** include a file named "`README.txt`" with your name, your NetID, the assignment number, and your lab section. This file should explain anything we need to know about how to build and run your project. In particular, be sure to explain how to run what parts of your submission for each question in the assignment.

Submit your solution as a single ZIP archive to BlackBoard before the deadline.

Late homeworks will not be graded and will receive a grade of 0.

All assignments and activities associated with this course must be performed in accordance with the University of Rochester's Academic Honesty Policy.