CS 235 - Homework 1

Deadline

11:59 pm on **Friday, September 3, 2021**.

Purpose

To make sure you have read over the course syllabus, to check your understanding of the basics of Java file names, compiling, and execution, and to start practicing some simple Java programming,

How to submit

You complete **Problems 1 and 2** on the course Canvas site (syllabus confirmation and reading questions, and some short-answer questions related to Java file name basics, compiling Java from the command-line, and executing Java from the command-line).

For **Problems 3 onward**, you will create the specified `.java` and `.txt` files, and then submit those to the course Canvas site. (You'll be creating `.class` files, also, but you do not submit those.)

Important notes:

- You will need to use the CS50 IDE in completing a few of the problems. See the handout "CS50 IDE Setup and Getting Started", on the public course web site, if you have not used this before.
- Here is the **initial** set of this class's **Java style standards** (ASK ME if any are unclear to you!):
  - Follow the Java naming standards that have been discussed in class.
  - Attempt "javadoc-style" comments for each Java class and method, in the same style as you see in posted in-class examples.
  - Everything inside a set of { } must be indented by AT LEAST 3 SPACES -- and the beginnings of statements that are sequential should be indented the SAME NUMBER of spaces. (That is, sequential statements should line up.)
  - { and } should each go on their OWN line, with { lined up evenly with the preceding line, with the { }'s contents indented by at least 3 spaces, and with } lined up with the opening {. That is, handle the curly braces as you see in all posted class examples!

Problem 1 - 20 points

Problem 1 is correctly answering the "HW 1 - Problem 1 - required syllabus confirmation and reading questions" on the course Canvas site.

Problem 2 - 10 points

Problem 2 is correctly answering the "HW 1 - Problem 2 - short-answer questions on Java file names, compiling, and execution" on the course Canvas site.

Problem 3 - 30 points

- You will find the source code for the Week 1 Lab versions of the classes `GameDie.java` and `GameDieTest.java` on the CS 235 public course web page, both posted along with this homework handout, and under "In-class Examples", in the Week 1 Lab section. Create copies of these classes that you will modify for this homework.
  - Include your name in an additional `@author` line in each of their opening comments, and
  - change the `@version` comment to have today's date.
- Compile and correct as necessary until your classes successfully compile, and `GameDieTest` successfully runs.
- In your `GameDieTest.java`, add at least the following:
  - declare a third `GameDie` instance with a different number of sides from those already declared
  - print to the screen the result of rolling that new `GameDie` instance at least once
  - print to the screen the result of displaying the current top of the new `GameDie` instance after that roll
  - (you may add additional statements using `GameDie` objects if you would like!)
- Again compile and correct as necessary until your classes successfully compile, and `GameDieTest` successfully runs.
- At this point, demonstrate that your `GameDieTest` was working by running the following in the CS50 terminal. (It would also work in a Mac OS X or Linux terminal, and it worked on the `vlab.humboldt.edu` command prompt when I tried it, also...!)

  ```
  java GameDieTest > hw1-3-demo.txt
  ```

  - (the > above is called output redirection -- it causes whatever would be printed to the screen to instead be printed in a file whose name is given after the >)
  - (would you like to see the contents of that file? The `more` command lets you view the contents of a file one screen at a time: `more hw1-3-demo.txt`)
- Submit your resulting file `hw1-3-demo.txt`. (You'll be submitting `GameDie.java`, and `GameDieTest.java` as part of Problem 4.)

Problem 4 - 40 points

- Consider how the methods and data fields are declared in the `GameDie` example.
- We decide we'd like a `numRolls` data field, storing the number of times a `GameDie` object has been rolled. This should initially be `0`, and should be increased by 1 each time that `GameDie` object is rolled.
  - So -- in `GameDie.java`, declare this private data field appropriately,
  - initialize it properly in `GameDie`'s constructors,
  - modify the `roll` method so it appropriately modifies `numRolls` each time a roll occurs, and
  - add an accessor method `getNumRolls` that returns the number of times that the calling `GameDie` has been rolled.

- Compile and correct as necessary until your modified class successfully compiles.
- In your `GameDieTest.java`:
  - print to the screen the result of calling your new `getNumRolls` method at least twice:

    ...once before one of the `GameDie` instances has been rolled,

    ...and once immediately after one of the `GameDie` instances has been rolled.
  - (you hope to see it return `0` for the not-yet-rolled die, and `1` for the just-rolled die)
  - (do you want to do this for more of your `GameDie` instances? That's fine, if you would like! And you may add additional statements using `GameDie` objects if you would like.)
- Again compile and correct as necessary until your classes successfully compile, and `GameDieTest` successfully runs.
- At this point, demonstrate that your `GameDieTest` was working by running the following in the CS50 terminal (or in a Mac OS X or Linux terminal, or in a `vlab.humboldt.edu` command prompt);

  ```
  java GameDieTest > hw1-4-demo.txt
  ```
- Submit your resulting files `hw1-4-demo.txt`, `GameDie.java`, and `GameDieTest.java`.

NOTE!

Would you like to make more additions to class `GameDie`? Feel free to do so! And also demonstrate them in your `GameDieTest` application class.