CSE12- Winter 2016 Project 1: Warm up.

Due: 10 January, 2016, at 23:59:59

Points: 70

Files to submit

Project 1 is designed just to make sure you are able to use the UNIX machines at UCSD, TritonEd and Vocareum.

Logistics:
In EACH AND EVERY FILE that you turn in, we need the following in comments at the top of each file. These are essential so that we can more easily process your submissions and ensure that you receive proper credit. Remember that if the file is a source code file, to put these in comments.

NAME: <your name>
ID: <your student ID>
LOGIN: <your class login>

Turn in:
See the submission instructions *at the end* of this document. You will need a *Vocareum* account for submission, so if you have not received your login information, post on the appropriate post on Piazza.

Getting Started
We strongly recommend that you work on the lab machines, but if you choose to work on your own machine, you can. Just make sure you know what you are doing. We'll be doing all of our testing from the lab machines. Lab accounts should be set up sometime during week 1. Instructions below assume you are using the lab machines.

Create a subdirectory call "HW1" in your class account. All of your files should be placed in that subdirectory. If you cannot remember how to create directories, refer to a unix tutorial

Problem #0 (3 points)
1. First read and sign the Integrity of Scholarship agreement for CSE 12
a) Go to TritonED: https://tritoned.ucsd.edu/webapps/portal/frameset.jsp
b) Content -> Homeworks->hw1
c) If the agreement is not signed that we will not grade your homework.
2. For another 2 points, fill out a short pre-test. Same location on TritonED. You will receive your points for just completing the quiz, although do your best. You have 3 attempts.

Problem #1 (7 points)

The purpose of this problem is to get your comfortable both on the command line as well as using the Eclipse IDE. For part A you may use any program you like to edit your java files (e.g., Dr. Java, vim, Notepad++, or even Eclipse), but we would like you to compile the program and generate Javadoc via the command line.

Download the following files from the hw1 directory (TritonEd) and save them to your HW1 directory:

CounterStat.java
CounterStat.pdf

Look at the file CounterStat.pdf. This is a PDF of documentation created using Javadocs. Using whatever editor you like (vim, Dr. Java, or even Eclipse), modify CounterStat.java with appropriate javadoc comments, so that it generates similar documentation. Replace the author field with your name.

Next generate the javadocs for this file **via the command line**, and place all of the documentation files in a subdirectory called **doc** in your HW1 directory. If you do not know how to do this, and don't know where to start, try Googling "javadoc command line" (without the quotes). I recommend skipping the StackOverflow link and going to the official Java page. The section on "options" will be particularly useful.

Look at the generated CounterStat.html file to be sure it was generated appropriately, and **matches** what is in CounterStat.pdf (with your name as the author). When you turn in CounterStat.java, we will run javadoc on your file to create the required documentation.

In addition, place the following information in your HW1-Answers.pdf file (also put it in doc directory)

- What command line is used to create the javadoc documentation in HW1/doc?
- What command-line flag(s) is/are used to create the author and version entries for the class?

Problem #2 (20)

Next you will write JUnit tests and run them from the **Eclipse**. Make sure you are done with problem 1 before continuing. Follow the Eclipse documentation or any other tutorial to create a new Java project with the package **hw1**. Create the class `CounterStat` in the `hw1` package. You can load in the existing source code, or you can create a new class and copy and paste in the code from `CounterStat.java`. Then, create a JUnit test class called CounterStatTest that contains the code from CounterStatTest.java. Again, a simple way to do this is to create a new Test and then copy and paste the code in, or you can import the tester class source file. Use Google, talk to the tutors and talk to your classmates if you are having trouble. ***It is completely OK to help each other out with this part (i.e. getting set up to work in Eclipse) and is not considered cheating.*** Compile your code and run the provided JUnit tests to make sure everything compiles properly.

Most of this file is already complete, and you should be able to compile and run it (see below). However, there are some 'TODO:' marked in comments where you are to complete the code. (both Java files).
These completions including adding comments at the top of the file, completing one method in CounterStat.java and completing the code to properly run some of the unit tests against the CounterStat class.

Take a screenshot of your code loaded into Eclipse after running the tests. Place that screenshot in your HW1-Answers.pdf file.

Here's what to do for this part:

1. First, complete the TODO items in CounterStat.java.
    a. Finish the method that calculates the average
    b. Create an INCORRECT resetWrong method()
2. Second, complete the TODO items in CounterStatTester.java
3. In particular, modify CounterStatTester.java so that your ResetWrong test fails. (This is what we expect, right? Wrong method should fail) The version of CounterStat.java that does not pass the ResetWrong test is the version you should turn in. To be clear. CounterStat.java must *compile* but it should fail a reasonable ResetWrong test. We will run your tests against an error-free version of CounterStat.java to insure that all tests pass. Then we will run your tests against your turned in version of Counter.java to see the failed Reset test.
4. What was the most difficult/confusing part about getting setup in Eclipse?

For the rest of this assignment (and the rest of this course) you can use any editing environment you choose.

Problem #3 (40)

Use Eclipse to complete this programming assignment. You job is to create a Java program that simulates a simplified war game:

- Computer generates a random card
- A user generates a card
- The larger card wins the game
- Same card (same kind) results in a tie.

We have provided some starter code in the SimpleWar.java which you can download from TritonEd. You will write your game in the main method, use helper methods if needed. Good style is encouraged since you will be graded it on (comments, wise use of helper methods, readability).

Game Flow:

When the game starts, computer chooses a card randomly and displays it, then there is a prompt for a user's input. User should input a number from 2-14: 2 is for 2,….14 is for ace. The program then generates a suit and outputs a card for the user. Based on the kind of the card, the winner is determined and the game continues. To exit the current game, the user should input -1. After -1 is pressed, you should show a statistics of the game, list of all the cards (history) and prompt a user for another game.

Here is a sample run (User input is shown in blue):

My card is: Two of Clubs
What is your card (kind)? (2-14, -1 to finish the game): 5
Five of Clubs
You won

My card is: Three of Clubs
What is your card (kind)? (2-14, -1 to finish the game): 2
Two of Hearts
I won

My card is: Six of Clubs
What is your card (kind)? (2-14, -1 to finish the game): 6
Six of Hearts
A tie

My card is: Five of Spades
What is your card (kind)? (2-14, -1 to finish the game): -1

I won: 33% You won: 33% We tied: 33%

My Moves: Two of Clubs Your Moves: Five of Clubs
My Moves: Three of Clubs Your Moves: Two of Hearts
My Moves: Six of Clubs Your Moves: Six of Hearts
Play again? n
Bye, see you later!

See more running examples in another file (on Ted, under hw1)

Your exact formatting doesn't have to match ours, but the game play and the info shown should match.

Here are some detailed requirements of the game play and specifics about the program:

- You will write your code in the main method, but you should use good style and helper functions as needed.
- There are no restrictions on what can be imported for this assignment.
- **Assume an intelligent player** (for this assignment).
    - Only valid input will be given. We are not trying to break your code, just checking that it works properly
    - For example, if you instruct the player to put (2-14), we won't put 'z'
- The game should repeat until the user enters 'n' or '-1' (-1 to exit current game, n to stop the program).
- The game should track the full guess history for the game in a LinkedList of Strings. These variables are already set up in the starter code. You just need to use them.
- You can't generate more than 4 cards of the same kind. If a user chooses, say, a king 5 times in a row, then say: "All cards of this type have been played. Pick another one." And let the user pick another card.
- You can't generate the same card twice during one game.
- Do NOT use a lot of if statements/switch..case to determine a winner. If you look at the starter code, you will see two string arrays: Kind and Suit. Use its indices to generate a random card and determine a winner/tie.

How to submit your homework

Files to submit: (Create doc directory in Vocareum)

CounterStat.java
CounterStatTester.java
SimpleWar.java
doc/CounterStat.html

doc/HW1-Answers.pdf

Below are instructions for submitting all of your homeworks for CSE12.
You should have received an email from support@vocareum.com indicating your userid and password for this website. This is our submission site (at least for the first homework)

1. Follow the link https://labs.vocareum.com/home/login.php and log in.

2. You should be able to see the course info and click on “details”

3. You should be able to see HW1: SimpleWar. Click on “Upload Assignment”.

4. You will see a pop-up page. Click on the ”Upload files” on the upper left corner. You can simply drag the files to window (but also READ STEP 5 to create the necessary directories).

5. If you want to add a new directory, you need to create a folder in Vocareum first, and then drag those files that belongs to that directory into the folder. **You should preserve the directory structure you used when testing your homework. Do not upload a zipped file.** If you are not sure if the directory structure is correct or not, go to step 7.

6. We are setting the number of submission to be unlimited. So feel free to submit the assignment multiple times.

7. Once you submit the assignment, you should be able to see a submission report:

```
Your submission passed!

stdout:
-------
Passed [File exists ( doc/HW1-Answers.pdf)]
Passed [File exists ( doc/CounterStat.html)]
Passed [File exists ( src/hw1/CounterStat.java)]
Passed [File exists ( src/hw1/CounterStatTester.java)]
Passed [File exists ( src/hw1/SimpleWar.java)]
Submission compiles!
```

Make sure your submission passes the submission step: it checks if the files are named and compile properly.