# Data Structures and Object Oriented Programming

Programming assignment #1

**Homework 1 is due April 10th, 2017, at 23:59:59**

Instructor: Marina Langlois

**Grading: 100 points**

Submit your files using Vocareum. Make sure that you follow directions on how to submit files precisely. Failing to do so in a future will result in 20-90% penalty of you grade. First homework will be exempt from this rule.

## Contents

## 1 Purpose of the project

Homework assignment #1 is designed to make sure you are able to use the UNIX machines at UCSD labs, TritonEd platform, Vocareum to submit your files and get you started to think and code in Java again.

## 2 Getting Started

You are allowed to work on your own machine, but we strongly recommend that you work on the lab machines. Just make sure you know what you are doing. We will 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 http://alvinalexander.com/unix/edu/examples/mkdir.shtml

## 3 Part 1 (6 points in total)

1. First download, read and sign the Integrity of Scholarship agreement for CSE 12 (2 points)
    - Go to TritonED (Ted): https://tritoned.ucsd.edu/webapps/portal/frameset.jsp
    - Content → Homeworks → HW1
    - Download, read and sign the agreement in the end. (Insert your full name and date)
    - Do not forget to submit the agreement file along with other homework submission files (read below)
    - **If the agreement is not signed then we will not grade your homework.**
2. For another 2 points, please fill out a short quiz. You can find it on Ted, same place as the agreement. You will receive your points for just completing the quiz but do your best.
3. For another 2 points, please watch the tutorial about Plagiarism Tutorial that should help you identify different scenarios and what to if you found yourself in one of them: http://libraries.ucsd.edu/assets/elearning/cse/cseplagiarismlink/story.html

## 4 Part 2 (10 points in total)

The purpose of this part is to get you comfortable both with the command line as well as with using the Eclipse IDE. For this part you may use any program you like to edit your java files (e.g., DrJava, vim, Notepad++, or even Eclipse), but **you must compile the**

**program and generate Javadoc via the command line.**

1. Download the following files from the HW1 folder (TritonEd/Content/Homeworks/HW1) and save them to your HW1 directory:
   Stats.java
   Stats.pdf

2. Open the file **Stats.pdf**. This is a **sample** PDF documentation created using javadoc, and the javadoc documentation that you generate must look like this sample. Using whatever editor you like (vim, DrJava, or even Eclipse), modify **Stats.java** with appropriate javadoc comments, so that it generates similar documentation. Replace the author field with **your** name and the date with the current date.

3. Next generate the javadocs for this file **via the command line**, and place all of the documentation files in your HW1 directory. If you do not know how to do this, and dont 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. Yes, Googling! That might seems surprising that I asked you to Google but knowing how to search for the information is very important skill in this course.

4. Check the generated **Stats.html** file to make sure that it was generated appropriately, and matches the content of**Stats.pdf** (with your name as the author and current date as the date). Do not worry about a style when you submit it. When you turn in **Stats.java**, we will also run javadoc on your file to create the required documentation.

5. In addition, create and place the following information in your **HW1-Answers.pdf** file

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

## 5 Part 3 (14 points in total)

Next you will write JUnit tests and run them from the **Eclipse**. Make sure you are done with Part 2 before continuing. Follow the Eclipse documentation in the Course Website/Appendix or any other tutorial to create a new Java project with the **package hw1**.

- Create the class *Stats* 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 *Stats.java.*

- Then, create a JUnit test class called *StatsTest* that contains the code from *StatsTest.java*. Again, a simple way to do this is to create a new class 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 JUnit 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 include: adding comments at the top of the file, completing one method in Stats.java and completing the code to properly run some of the unit tests against the Stats class. Refer to the following instructions.

Here is what to do for this part:

1. First, complete the TODO items in Stats.java.
   • Complete the method that calculates the average
   • Create a resetWrong() method that resets the statistics incorrectly (i.e: not resetting the numbers correctly)"
2. Second, complete the TODO items in StatsTest.java In particular, modify StatsTest.java so that your ResetWrong test fails. (This is what we expect, right? Wrong method should fail). The version of Stats.java that does not pass the ResetWrong test is the version you should turn in. To be clear: Stats.java must compile but it should fail a reasonable ResetWrong test. We will run your tests against an error-free version of Stats.java to insure that all tests pass. Then we will run your tests against your turned in version of Stats.java to see the failed Reset test.
3. Take a screenshot of your code loaded into Eclipse after running the tests. Place that screenshot in your HW1-Answers.pdf file.

## 6 Part 4 (45 points in total)

Use Eclipse to complete this programming assignment. I spent my spring break in Las Vegas but did not get a chance to gamble. So I decided to create a little game of my own and I need your help to finish it. The name of the game is *Pig*. Here is a link that describes the rules: https://en.wikipedia.org/wiki/Pig_(dice_game). You will implement a Two-Dice Pig version (https://en.wikipedia.org/wiki/Pig_(dice_game)#Two-Dice_Pig). Also, to help you understand the logistics, you could play a few rounds of this game online: http://www.playonlinedicegames.com/pig. (Note: It is a single-die game).

The following part explains the game flow, followed by the instructions for the program (to be completed in the file Pig.java under the HW1 folder).

**Game Flow**:

In this game we asssume that the human always starts before the computer. Following should be the game flow:

- In the human's turn, roll two dies (each with random numbers form 1-6, Hint: Use random number generator), prompt him/her for the next action: 1 to continue rolling, 2 to stop/hold (pass turn to the opponent). Act accordingly. Do not forget to check if either output is 1, or both the outputs are 1 or they are a double. Handle all these cases wisely. At the each of every roll, print the score till that point for that particular turn. Print the total score when the turn is over.
- Computer's turn is very similar. The difference is that option to continue or stop/hold is generated randomly.
- Alternate between the human and computer turns until either one of them wins. Set the limit to something small, like 20 to play the game faster.

Once a game finishes, your code needs to:

- Show the statistics of the game(s).
- Ask if a user wants to quit the game (in which case the game should be terminated with a 'Thank You for playing!' message) or if he/she wants to play again. If the user wants to play again, ask if he/she wants to clear the statistics before starting a new game.

Below is a sample run. Note that you can find more sample runs under HW1 (TritonEd) in a folder named SampleRuns.

```
----------------------------------
Scores. You: 0, Computer: 0
----------------------------------
**** Your turn starts ****

Your roll - Dice1: 6, Dice2: 4

Your score in this turn: 10

Enter a number: 1 to continue, 2 to stop/hold
2
Your total score in this turn: 10

Overall total score for you: 10

----------------------------------
```

```
Scores. You: 10, Computer: 0
-----------------------------------
**** Computer's turn starts ****

Computer's roll - Dice1: 3, Dice2: 4

Computer's score in this turn: 7

Computer stops

Computer's total score in this turn: 7

Overall total score for computer: 7

-----------------------------------
Scores. You: 10, Computer: 7
-----------------------------------
**** Your turn starts ****

Your roll - Dice1: 5, Dice2: 2

Your score in this turn: 7

Enter a number: 1 to continue, 2 to stop/hold
1
Your roll - Dice1: 6, Dice2: 1

Your total score in this turn: 0

Overall total score for you: 10

-----------------------------------
Scores. You: 10, Computer: 7
-----------------------------------
**** Computer's turn starts ****

Computer's roll - Dice1: 4, Dice2: 6

Computer's score in this turn: 10

Computer continues

Computer's roll - Dice1: 5, Dice2: 2
```

```
Computer's score in this turn: 17

Computer stops

Computer's total score in this turn: 17

Overall total score for computer: 24

Computer won game this game!
--- Statistics ---
  Computer won: 100.0%, You won: 0.0%

Do you want to play again? Enter 1 for Yes, 2 for No
2
Thank You for playing!
```

**Code Rules**

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

- We have provided some starter code in the *Pig.java* and *Stats.java* which you can download from TritonEd. You will write your game in the main method, use helper methods if needed. If the same code repeated more than once, use a helper method. **Good style is a must since you will be strictly graded on it.** Refer to the Style guide in the Appendix for more information.
- You **must** use a *Stats* object to keep track of statistics.
- The game should repeat until the user enters n.
- You can assume a semi-intelligent player: The input numbers are only 1, 2 for the game options. And the user will only enter either 'y' or 'n' when prompted for another game.
- For HW1, your exact formatting does not have to match ours, but the game play and the info displayed should match the samples given.

## 7 Java questions (25 points in total)

Create a text file called *JavaTrueFalse.txt* and write down the answers to the questions in HW1/HW1_TF.pdf (TED), with the number of the question followed by either the word **True** or **False**. One answer/line. eg.
1. True
2. False

and so on. No name on the top please. Just your answers. This allows us to auto-grade this part. **Make sure you follow the above format.**

This part is open book and open notes. You may use Google to help you determine the answers to these questions, and you may run any Java code to help you determine the answers. However, you may not ask your classmates for the answers nor may you give the answers to any of your classmates. The point is to understand the answers, as we assume that you have this knowledge from CSE 11 or CSE 8B and we will build on it.

## 8 Extra Credit (5 points in total)

You could implement another version of Pig (Big Pig) and become a Pig expert. The rules are also in Wiki. You need to:

1. Change your code according to the algorithm you found.
2. Describe the major changes that you've made and put your answers to HW1-Answers.pdf.
3. Name your file *BigPig.java*
4. Put the code in a package hw1 (just like the rest of the java files).
5. submit it with other files.

## 9 Turning files in

In each file that you turn in (except JavaTrueFalse.txt), we need the following comments at the top of each file. These are essential so that we can more easily process your submissions and insure that you receive proper credit. Also, follow the coding style guidelines on the course website.

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

**Homework Submission**

Below are instructions for submitting all of your projects for CSE12. You should have received an email from *supportvocareum.com* indicating your *userid* and *password* for this website. This is our submission site. If you have not received your login information, request one in the appropriate post on Piazza forum. If you were not added to Piazza, then you can add yourself by using this link: (add the link).

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 PA1. 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.
5. We are setting the number of submission to be unlimited. So feel free to submit the assignment multiple times. **However, keep in mind that only the latest submission will be graded.**
6. Once you submit the assignment, you should be able to see a new yellow button called check submission. We have created a check submission script for you to check if your submission has the correct directories, files and their structures.
7. Once you submit and receive a message (see below) that everything is fine, you are done with submitting.

If something is wrong (name of the file, number of the files etc) there will be an error message. Make sure you correct your errors and resubmit. **We are going to grade the latest version you submit**.

**Files and directories to submit**

1. Stats.java
2. Stats.html
3. StatsTest.java
4. Pig.java
5. HW1-Answers.pdf
6. Agreement.txt
7. JavaTrueFalse.txt
8. BigPig.java (optional)

## 10 Appendix

### 10.1 Eclipse Tutorial

Eclipse_Tutorial_Link

## 10.2 Style guide

Style_Guide

## 10.3 Converting different files to .pdf

It is not enough just to change extension from, say, .docx to .pdf. Try it, and then open your new "pdf" file. Did not work, right? It means you need a few extra steps to make a conversion.

Conversion