EECS1022 -Winter 2023- Lab05

Due Date: Friday, March 10, 2023, before 23:59

Contents

EECS1022 -Winter 2023- Lab05

Due Date: Friday, March 10, 2023, before 23:59

Instructions:

Important Reminders

- You should attend your lab session (the one you are enrolled in). If you need to change your lab enrollment, you should contact the Undergraduate Office in the department. ***Instructors or TAs cannot change your enrollment***.
- You can submit your lab work in eClass any time before 23:59 on Friday (**March 10, 2023**) of the week the lab is due. Your last submission will overwrite the previous ones, **and only the last submission will be graded**.
- The deadline is strict, with no excuses: **you receive 0 for not making your electronic submission in time**. ***Emailing your solutions to the instructors or TAs will not be acceptable.***
- To submit your work, you need to use the York eClass.
- **Your submission will be graded by JUnit tests given to you and additional JUnit tests covering some other input values. *This is to encourage you to take more responsibility for the correctness of your code by writing more JUnit tests.***
- Developing and submitting a correct solution for this lab without compilation errors is essential. Hence, you must take a reasonable amount of time to test your code in different ways. If you submitted a solution with a small mistake in terms of syntax or do not comply with lab instructions, then you may receive 0 as a grade for the implementation of this lab
- There will be **a 25% penalty** on your lab final grade if your submitted code does not compile due to ***minor compilation errors***, given that TAs can fix these minor compilation errors. **You will receive a zero if your code contains major compilation errors that TAs can not fix**.

Academic Honesty

- Students are expected to read the Senate Policy on Academic Honesty. See also the EECS Department Academic Honesty Guidelines.
- **All labs are to be completed individually: no group work is allowed. Do not discuss solutions with anyone other than the instructor or the TAs. Do not copy or look at specific solutions from the net. *If you are repeating the course, you are not allowed to submit your own solution developed in previous terms or for other purposes. You should start from scratch and follow the instructions.***

EECS1022 -Winter 2023- Lab05

Due Date: Friday, March 10, 2023, before 23:59

Learning Outcomes and Objectives

The lab aims to familiarize students with creating new data types using a class. Your task is to write the Java code that meets these classes' specifications. Also, test your Java code using JUnit testing.

Learning Outcomes

- **CLO1**: *Create new data types using the notion of a class, declared with primitive and/or reference attributes, constructors, and methods (accessors and/or mutators).*
- **CLO2**: *Illustrate the difference between a class and its instances (objects) by writing a program which instantiates objects from classes and calls methods on those objects.*
- **CLO4**: *Write unit tests to check correctness of classes and use an IDE debugger to correct errors.*
- **CLO5**: *Use primitive arrays*, linked lists, and library collections (e.g., lists, tables) to implement iterative and recursive algorithms including searching (e.g., linear vs. binary) and sorting (e.g., selection sort, insertion sort, merge sort).

Lab Learning Objective

- To create a class with some attributes.
- To create different types of constructors; default, copy, ...
- To use a chain of constructors
- To create an object and call setter or getter methods
- To use JUnit Tests to verify your work
- To create methods based on a given API

Important pre-lab work you need to do before going to the lab

a. http://wiki.eclipse.org/The_Official_Eclipse_FAQs
b. Testing using Eclipse IDE
   a. Video: Java Unit Testing with JUnit - Tutorial - How to Create And Use Unit Tests
c. Read about debugging using Eclipse IDE
   a. https://www.eclipse.org/community/eclipse_newsletter/2017/june/article1.php
   b. Video: How to set breakpoints to debug code in Java using Eclipse Debug mode
   c. Video: How To Debug Java Code The Right Way - Eclipse Debugger Full Tutorial

Feel free to find online resources on these and share them with your classmate on the discussion forum

Getting Started

1. Start eclipse.
2. **Download the starter code "Lab5.zip" from the eClass course site**
3. Import the test project by doing the following:
   1. Under the **File** menu, choose **Import...**
   2. Under **General,** choose **Existing Projects into Workspace** and press **Next**
   3. Click the **Select archive file** radio button, and click the **Browse...** button. You may have to wait about 10 seconds before the file browser appears.
   4. In the file browser that appears, navigate to your home directory.
   5. Select the file **Lab5.zip** and click **OK**
   6. Click **Finish**.
4. All files you need for this lab should now appear in eclipse.

Lab Structure

After successfully importing the starter code/project "Lab5.zip"

The lab folder/directory structure is as follows:

- **src/lab5**/: directory contains Java files: **Game.java**.
- **src/lab5**/: directory contains Java files (JUnit test cases): **JunitTest_GameTest.java.** These files contain several JUnit test cases that can help to test your code. *It should be noted that you need to run the JUnit tester* ***JunitTest_GameTest.java*** *after you complete the* ***Game.java*** *class to check your work. Nonetheless, passing all given tests does not guarantee full marks for this lab. Therefore, you are required to write additional tests to ensure the correctness of your implementations.*
- **doc/**: directory contains Java documentations for lab in HTML format.

EECS1022 -Winter 2023- Lab05

Due Date: Friday, March 10, 2023, before 23:59

Lab Restrictions:

- Any use of Java library classes or methods (e.g., **ArrayList, System.arraycopy**) is forbidden. That is, there must **not** be any import statement at the beginning of this class. Violation of this requirement will result in a **70% penalty** on your marks.
- For the JUnit test cases, the class **JunitTest_GameTest.java** given to you
  - Do not modify the test methods given to you.
  - You are allowed to add new test cases by creating new test methods.
- For each method which you are required to implement, **derived from the JUnit test methods**:
  - No **`System.out.println`** statements should appear in it.
  - No Scanner operations (e.g., **`input.nextInt()`**) should appear in it. Instead, declare the method's input parameters as indicated by the JUnit tests.
- **Hint: You may use `java.util.Arrays` class, class `Character` or Class `String`.**

Lab Exercise

Computational thinking for a software developer/computer programmer is a critical skill that is consistently applied. This lab requires you to develop a solution using Java object-oriented programming that simulates a basketball shootout game.

Two players agree to limit the number of **ball throw attempts to 3 throws each**. The first player will make all three throw attempts (keep track of the successful baskets made where the ball goes into the basket). After the first player completes all three shots, the second player will make all three throw attempts. The player who makes the most baskets (gets the ball in the hoop) will be declared the winner. In the case of a tie, the tie counter is incremented by one. Then, the game is repeated until a winner can be determined. Note that the game can be repeated many times.

The losing player of the shootout game will have to give the winning player a move ticket(s). The number of movie tickets is determined by the total number of baskets made by the winner, less the total number of baskets made by the losing player. The losing player gives half of a movie ticket for every tied game (if there were any tied games). If the final calculated number of movie tickets has a decimal value, it should be rounded to the nearest whole number since you can't purchase half a ticket!

Example: If the player-1 made a total of 3 baskets, and player-2 made a total of 2, and they had three tied games, the number of movie tickets would initially be **3-2=1**, but increased by **3 X 0.5=1.5**, making the owed number of tickets 2.5 which must be rounded up to **3 movie tickets**.

In this lab, you need to write Java code to implement the **Game.java** class described by the given Java API inside the **doc folder** where the documentation for this lab is stored. You'll see there is a file called **index.html**. Clicking on this file shows the lab/project documentation in your browser. You do not have to include JavaDoc comments.

Infer Class and Method Specifications from JUnit Tests

You may have noticed that the above "overview" descriptions are not as precise as the class specifications and method implementations. In fact, unlike the previous labs, we will not provide detailed specifications in this handout. To obtain the precise specification, you need to carefully ***analyze the test cases*** in the provided JUnit tests to understand the expected behaviours of each method.

In professional software development, test cases often play a vital role in specifying software requirements. This is called **Test-Driven Development** (TDD). Read more about it here: https://en.wikipedia.org/wiki/Test-driven_development.

Submit your work by using the course eClass

Check List:

Before submitting your files for this lab, you need to make sure you completed the following

	There is No compilation error generated from your implementation
	The **Game.java** file contain the implementation for this lab.

Submit The Following File:

1) You need to submit **one** file, **Game.java.**