EECS1022 -Summer 2022- Lab04

Due Date: Friday, June 17, 2022, before 22:00

Contents

EECS1022 -Summer 2022- Lab04

Due Date: Friday, June 17, 2022, before 22:00

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 22:00 on Friday (**June 17, 2022**) 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, it's important you 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 -Summer 2022- Lab04

Due Date: Friday, June 17, 2022, before 22:00

Learning Objectives

- Given a computational problem, develop a Java solution composed of:
  - Numerical Literals and operators
  - String Literals and operators
  - Variables and assignments
  - (Nested) Selections/Conditionals/If-Statements
  - OOP Basics: Classes, Attributes, Constructors, Accessor and Mutator Methods, Method Invocations, Context Objects, Dot Notation
  - Inferring Java Classes from JUnit Tests
- Use the given JUnit tests to guide the development.

Style Rules

Please refer to the Lab2 handout for style rules.

Getting Started

1. Start eclipse.
2. **Download the starter code "Lab4.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 **Lab4.zip** and click **OK**
   6. Click **Finish**.
4. All files you need for this lab should now appear in eclipse.

EECS1022 -Summer 2022- Lab04

Due Date: Friday, June 17, 2022, before 22:00

Lab Structure

After successfully importing the starter code/project "Lab4.zip"

The lab folder/directory structure is as follows:

- **src/lab4**/: directory contains a Java file named **YorkVendingMachine.java**.
- **src/lab4**/: directory contains a Java file (JUnit test cases) named **JunitTest_YorkVendingMachine.java**. This file contains several JUnit test cases that can help to test your code.
  It should be noted that you need to tun the JUnit tester ***JunitTest_YorkVendingMachine.java*** *after you complete the* **YorkVendingMachine***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.*

EECS1022 -Summer 2022- Lab04

Due Date: Friday, June 17, 2022, before 22:00

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.

EECS1022 -Summer 2022- Lab04

Due Date: Friday, June 17, 2022, before 22:00

Lab Exercise

In this lab, you need to write Java code to implement the YorkVendingMachine class.

The York Vending Machine Problem

You are required to develop an object-oriented program solving a (simplified) York Vending Machine (YVM) problem:

- There are four products offered (attend to the exact spellings):
  - "Coke"
  - "Orange Juice"
  - "Kitkat Chunky Bar"
  - "Lay's Classic Chips"
- The VM's LED screen always displays a message:
  - When first created, the message is "Empty YVM Started"
  - The message only changes when the add or dispense operation (described below) is invoked.
- Given a YVM instance, say yvm, there are five operations supported:
  - Get the current message displayed on the YVM: **yvm.getMessage()**.
  - Check the overall stock of the current YVM: **yvm.checkStock().**
  - Given a product name, check its current stock, e.g., **yvm.checkStock("Coke")**
  - Given a product name, add units to its stock, e.g., **yvm.addtoStock("Coke", 5)**
  - Given a product name, dispense units from its stock, e.g., **yvm.dispense("Coke", 2)**
  - Clear the overall stock of the current YVM: **yvm.clear()**
  - Add the same number of units for each product of the current YVM: **yvm.addBatch(5).**
- Input names to the addtoStock and dispense operations may be invalid, in which case you need to return error messages accordingly.
- Input units to the addtoStock and dispense operations are always valid: you need not worry about non-positive or too-large values.

EECS1022 -Summer 2022- Lab04

Due Date: Friday, June 17, 2022, before 22:00

Infer Class and Method Specifications from JUnit Tests

You may have noticed that the above "overview" descriptions are not precise enough as the specifications of the class and method implementations. In fact, unlike the previous labs, we will not provide the 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 **YorkVendingMachine.java** file contains the implementation for this lab.

Submit The Following File:

1) You need to submit the **YorkVendingMachine.java** file.