MILESTONE 3

Overview

In this milestone, you will be making your code more robust and resistant to errors. You will be handling exceptions, and perhaps using them, so that your application does not crash unexpectedly. A user might enter something unexpected, causing the application to crash. This would be disastrous for your store!

Furthermore, you will be implementing a complete JUnit test harness to test all the methods defined in your classes. This will ensure that everything is functioning and will function correctly across all possible cases.

Well, in this project, you will not be required to implement tests for all methods, just some of them…

*Keep in mind that most of the tasks you will be required to do are relatively **open-ended**. You must justify any decisions you made that were not obvious in your Change Log. Refer to the examples in Milestone 1.*

Adding Exception Handling

Your first task is to add exception handling to your code. To do so, make use of ***try-catch-finally*** blocks to ensure that at any point in your code, regardless of what the user enters, the application does not crash or throw an unhandled error.

Document the points of vulnerability that you identify in your code from Milestone 2 in your Change Log. You can even make use of errors in your code (as seen in the labs) to alter the control flow of the program. This decision is up to you.

Exception Handling must be added to all classes that require it.

JUnit Testing

Software should always be tested as completely as possible before being deployed for use. You will be using JUnit to perform Unit Testing on your classes and methods. Your JUnit test classes should exist in a separate package from your source code.

Note 1: You should not test your methods that take user input using JUnit.

Note 2: You may have to add the relevant JUnit libraries to your classpath. IntelliJ makes this easy. Make the necessary imports, and IntelliJ will detect the missing libraries and prompt you to add them.

You need to provide complete testing of:

- StoreManager
- An additional class. You can choose between Inventory or Shopping Cart.

Make sure that your change log includes which part of the codes you were not have been able to test using JUnit. It should also specify if changes were needed in the code due to testing.

Questions

1. What were the testing methods/strategies that you used for this milestone? Be detailed. Use the testing terminology presented in lectures.
2. Were there some things you were unable to test with JUnit? What were they, and why were you not able to? (*Think about the levels of testing.*)
3. With respect to Question 2, should these parts of the code be tested? Should every inch of code be tested in general?

Milestone 3 Deliverables

1. The following classes completed according to Milestone 3 specifications. This means, with exception handling.
    a. *StoreView.java*
    b. *ShoppingCart.java*
    c. *Inventory.java*
    d. *StoreManager.java*
    e. *Product.java*
2. A package named ***storetest*** with the JUnit test classes.
3. Everything applicable to Milestone 3 from General Submission Requirements:
    a. *storeUML_M3.pdf* with the updated version of your UML diagram
    b. *report.pdf* with the Change Log and the answers to the questions.

Note: remember to document all your classes.