6CCS3SMT & 7CCSMASE Software Measurement and Testing

Coursework Overview

The coursework accounts for 15% of the final assessment and is an individual project divided into two equally important components: the test code and the report. The primary objective is to evaluate your reasoning when making decisions on the testing of a given piece of software and your ability to use the taught technologies such as JUnit and Mockito (or equivalent frameworks in Python) to write test cases. The source code for the software system under test is provided in both Java and Python, so you can use the selected technologies for any of these two programming languages.

The Task

You are provided with the source code of the software and the list of the requirements for it. The provided software contains faults, the number and locations in code of which will not be disclosed. You are required to write a report that explains your approach to testing the provided software. In your report, describe the types of testing you implemented based on what you've learned in class, and specify which parts of the code you prioritised for testing along with your reasoning. Discuss any trade-offs you encountered, such as time constraints or resource limitations, and reflect on the challenges you faced during the testing process, including how you addressed these challenges. Document any faults you detect during testing and explain how they were identified through your test cases. The report should be **up to 2000 words**.

In addition to the report, you need to write test cases that effectively test the functionality of the software. Focus on the quality and readability of your test code rather than solely aiming for full coverage. Generate a coverage report using a coverage tool of your choice and include it in your report. Use the provided system requirements to guide the design of your test inputs and expected outputs, and be sure to incorporate unit tests with mocks where necessary to isolate components of the software.

Software System under Test

The Shopping Cart system is an online shopping platform designed to facilitate a seamless purchasing experience for customers. It enables users to browse and add products to their cart while providing robust functionality to calculate the total cost accurately. The system incorporates a variety of discount mechanisms, including tiered discounts based on cart value, customer-specific pricing based on classification (Regular, Premium, or VIP), and bundle discounts for specific product combinations. Additionally, it supports time-limited promotions and the use of coupon codes for further savings.

List of system requirements:

1. The system shall allow users to browse and select products to add to a shopping cart.
2. The system shall calculate the total price of items in the shopping cart before applying any discounts.
3. The system shall apply a bundle discount: 10% off the price of each mouse if at least one laptop is in the cart. This discount applies for all mouse-laptop pairs.
4. The system shall apply tiered discounts based on the cart subtotal after bundle discounts:
    - 15% off if the subtotal exceeds £2,000
    - 20% off if the subtotal exceeds £7,000
    - 25% off if the subtotal exceeds £15,000
5. The system shall categorize customers into three types: Regular, Premium, and VIP, with Premium customers receiving an additional 10% discount and VIP customers receiving an additional 15% discount on their total.
6. The system shall allow users to enter a single coupon code per transaction. Available coupons:
    - "DISCOUNT10" for 10% off
    - "SAVE50" for £50 off.
7. The system shall support time-limited promotions that can be activated or deactivated. During an active promotion, a flat discount of 25% shall be applied. This discount shall be applied in addition to any other applicable discounts (bundle, tiered, customer type, or coupon).
8. Discounts shall be applied in the following order. First, bundle discounts and fixed-amount coupon discounts are applied, reducing the overall total of the order. Next, all percentage-based discounts (including tiered cart value discounts, customer type discounts, percentage-based coupon codes, and time-limited promotions) are applied to the updated total. Note that percentage discounts can be applied in any sequence without changing the final result.
9. The system shall print a detailed receipt summarising the items in the cart, the total price before discounts, and the final price after all applicable discounts.

10. The system shall display an error message if the credit card number does not meet the 16-digit requirement or if the transaction amount is zero or negative, preventing the transaction from proceeding.

Example Scenario:

To explain further how the discounts are stacked let's consider an example scenario. We have a VIP user with a shopping cart of one laptop (costs £3000) and one mouse (costs £100). The customer also has a coupon code of "SAVE50" and it is a time-limited promotion period. First, the 10% bundle discount will be applied to the customer's shopping cart resulting in the overall amount of £3100 - £10 (10% of the mouse cost) = £3090. Then, the customer will have £50 off (due to coupon code of SAVE50) before any discounts and the amount will become £3040. Lastly, an overall discount of 55% (15% for shopping cart over 2000 + 15% for being VIP + 25% for promotion time) will be applied altogether and the final amount the customer pays will be £3040 * (1 – 0.55) = £1368.

Submission Guidelines

You need to submit your work in two different submission areas on Keats: "Coursework Report Submission" and "Coursework Artefact Submission". As the name suggest, in the "Coursework Report Submission" you need to submit a pdf file with your report.

In the "Coursework Artefact Submission" you should submit a single zip file containing:

1. Your test cases in the folder named "Tests"
2. Your coverage report in the folder named "Coverage"
3. A text file listing any dependencies for your test cases (if any).

Marking Rubric

The 15 points for the coursework will be awarded based on the following marking rubric:

The rationale for testing	The rationale for input/output selection is weak or poorly explained. **0 points**	The rationale for input/output selection is mostly logical and well-articulated. **1 point**	The rationale for input/output selection is highly logical, clearly articulated, and demonstrates deep understanding. **2 points**	
Inputs and outputs correspond to system requirements	The input and output selection did not consider system requirements at all. **0 points**	They inputs and outputs cover only part of the system requirements. **1 point**	The inputs and outputs cover all system requirements. **2 points**	
Edge cases	The selected inputs do not consider any edge cases **0 points**	The selected inputs consider edge cases **1 point**		
Coverage report & rate	There is no coverage report provided. **0 points**	The implemented test cases achieve low coverage. **1 point**	The implemented unit test cases achieve high coverage. **2 points**	The implemented test cases do not achieve high coverage but the author of the report provides a good justification for the lack of high coverage. **2 points**
Mocking	Mocks have not been used in the implemented test cases at all, and there is no justification on why their use was inappropriate for the provided project. **0 points**	Mocks have been used, but their usage is not appropriate. **1 point**	Mocks have been used appropriately in the implemented test cases. **2 points**	Mocks have not been used in the implemented test cases, but there is a good justification why mocking is not appropriate for the testing of the provided project. **2 points**
Percentage of detected faults	No faults detected **0 points**	Small number of known faults in the code have been detected. **1 point**	Medium number of know faults in the code have been detected. **2 points**	All known faults were detected. **3 points**
Explanation on how faults were detected	Faults are undocumented and/or poorly described. No meaningful explanation of how faults were identified through testing. **0 points**	Faults are partially documented with basic descriptions. Explanations of how faults were identified are included but may lack clarity, detail, or	Faults are thoroughly documented with clear, detailed descriptions. Explanations of how each fault was identified are	

		explicit linkage to test cases. **1 point**	precise and directly linked to the relevant test cases. **2 points**
Test code	The test code is not well-organised and/or is not readable **0 points**	The test code is well-organised and is readable. **1 points**	

Frequently Asked Questions

Can the report exceed 2000 words?
No. The 2000-word limit is a strict requirement and must not be exceeded.

Are there specific Python/Java/Mockito (or other library) versions we should use?
No, there are no strict version requirements. However, it is recommended that you use the latest stable versions of all tools and libraries.

Is there a required structure for the report?
There is no mandatory structure, but you should include a table summarising the detected faults. The table should include columns such as:

- **Class name** where the fault is located
- **Line numbers** of the fault
- **Description** of the fault
- **Test case(s)** that expose the fault

Should we fix the faults or modify the source code in any way?
No. You should not alter or fix the source code. Your test cases should execute the provided code as is and demonstrate the faults through failing tests.

What format should the coverage report be submitted in?
The format depends on the tool you use. Most tools generate reports in HTML or similar formats. You should include all files related to the coverage report in a folder named "Coverage" within your submission. If your coverage tool does not produce exportable files, you may include a screenshot of the coverage report instead.