Coding Tests

**Generic Coding Instructions**

* **Choose any one** of the 2 problems below.
* Can be written in **.NET languages**
* The implementation **should adopt** **OOPS,** clean code practices, SOLID principles, and best practices.
* **Should be accompanied by unit tests (unit test is mandatory)** & preferably written using **TDD approach**
* Should commit code to a **public git repository (github)** under a public handle
* The codebase should be checked in properly in GitHub. **Please DON’T zip and upload**. Zip attachment will be straight away rejected.
* **Avoid high cyclometric complexity.**
* Use generic package names
* Commits should be incremental so that one can look at the commit log and make sense of how the code has progressed along with the test cases. (recommended at least up to 20 commits to show how the code progresses; larger number of commits isn’t a problem)
* The promotion rules are mutually exclusive, that implies only one promotion (individual SKU or combined) is applicable to a particular SKU. Rest depends on the imagination of the programmer for which scenarios they want to consider, for example (case 1 => 2A = 30 and A=A40%) or (case 2 => either 2A = 30 or A=A40%)

**Problem Statement 1: Promotion Engine**

We need you to implement a simple promotion engine for a checkout process. Our Cart contains a list of single character SKU ids (A, B, C. ) over which the promotion engine will need to run.

The promotion engine will need to calculate the total order value after applying the 2 promotion types

* buy 'n' items of a SKU for a fixed price (3 A's for 130)
* buy SKU 1 & SKU 2 for a fixed price ( C + D = 30 )

The promotion engine should be modular to allow for more promotion types to be added at a later date (e.g. a future promotion could be x% of a SKU unit price). For this coding exercise you can assume that the promotions will be mutually exclusive; in other words if one is applied the other promotions will not apply

Test Setup

Unit price for SKU IDs A 50

B 30

C 20

D 15

Active Promotions

3 of A's for 130

2 of B's for 45 C & D for 30

Scenario A

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | \* A | 50 | |
| 1 | \* B | 30 | |
| 1 | \* C | 20 | |
| Total | |  | 100 |
| Scenario | | B |  |
| 5 \* A | |  | 130 + 2\*50 |
| 5 \* B | |  | 45 + 45 + 30 |
| 1 \* C | |  | 28 |

Total 370

Scenario C

|  |  |  |
| --- | --- | --- |
| 3 | \* A | 130 |
| 5 | \* B | 45 + 45 + 1 \* 30 |
| 1 | \* C | - |
| 1 | \* D | 30 |

5/3012020

Total 280

Coding Tests - New SCM Platform - Confluence

**Problem Statement 2:** Business **Rules Engine**

Imagine you’re writing an order processing application for a large company. In the past, this company used a fairly random mixture of manual and ad-hoc automated business practices to handle orders; they now want to put all these various ways of hanadling orders together into one whole: your application. After a full day of workshops you have gathered the following set of rules which need to be managed by the new system.

* If the payment is for a physical product, generate a packing slip for shipping.
* If the payment is for a book, create a duplicate packing slip for the royalty department.
* If the payment is for a membership, activate that membership.
* If the payment is an upgrade to a membership, apply the upgrade.
* If the payment is for a membership or upgrade, e-mail the owner and inform them of the activation/upgrade.
* If the payment is for the video “Learning to Ski,” add a free “First Aid” video to the packing slip (the result of a court decision in 1997).
* If the payment is for a physical product or a book, generate a commission payment to the agent. Design a new system which can handle these rules and yet open to extension to new rules

References

* <https://hackernoon.com/introduction-to-test-driven-development-tdd-61a13bc92d92>

No labels