CS202, assignment 3, Total marks – 15.
Deadline – 21/04/2019 11.55 PM

Question 1

Amacon.com

You have been approached by a startup ecommerce company **Amacon** to create a software application that could be used to sell their products. Your software implementation must use object oriented programming approach. You must also use Java Collection Framework and exception handling techniques where you define and throw your own specialized exceptions. Amacon software application involves three entities: **database**, **administrator**, and **customer**.

The **database** would be updated only by the **administrator** and it should support the following operations:

1. The database should have the capability to **insert** a new category or a new product. Each product in the database should have a price and the number of units available. Each product is listed under exact one category. For example, in the path “electronics > smartphone>Oneplus", “Oneplus” is a product listed under subcategory “smartphone” and that in turn is under the top level category “electronics”. This method should be called with two parameters: first is the path to the subcategory and second parameter is the product to be added in that subcategory. For the same example mentioned above, call to insert will be as: **insert("electronics > smartphone", "Oneplus")**. If that subcategory does not exist in the database then this function should create the above subcategory along with the product mentioned. If the database already has that product under that category then program should raise a custom exception. While adding this product in the database, insert should take input from the administrator about the properties of that product.

2. The database should have the capability to **delete** a sub-category/ product. Example, this method could be called in either of following two ways: **delete("electronic > smartphone>Oneplus")** or **delete("electronic > smartphone").** The first one just delete the product Oneplus but the second one removes all the subcategory/products under

"smartphone" subcategory. If invalid path is specified in delete API then program should raise a custom exception.

3. The database should have the capability to **search** for a product and is called as **search("Oneplus")**. If the product does not exist, the program should raise a custom exception. If the product exists, then the search method will print the path to the product as well as details of the product. This method should also return the reference to the product.

4. The database should have the capability to **modify** the properties of a product, i.e change its price or the number of units in stock. It can internally call **search** API described above to get the reference to the product for updating the properties of that product. It can take input from administrator for the properties to be updated.

5. The database should be able to handle the **sale** of a product. This API would be called by the customer after he has added some products in his cart. This API accepts 3 parameters as follows: **sale(product, quantity, remaining_funds_with_customer)**. If that much of stock is unavailable or if funds are insufficient to buy this product the program should raise appropriate custom exceptions.

Customer

A customer has to explicitly add funds. He is provided with a shopping cart that can hold multiple products of his choice. This cart should support the following operations:

1. The cart should have the capability to **add a product** and store it. This should make use of **search** operation on the database. While adding a product to the cart customer should add the quantity to purchase as well.
2. The cart should have the capability to **check-out**, i.e buy **all** products in the cart. This should make use of **sale** operation on the database.

You have to implement a menu-driven system that supports the following queries. A user can act as an administrator and a customer (both only one per program invocation), which must be prompted as input before making the following queries:

1. Administrator
    a. Insert product/category
    b. Delete product/category
    c. Search product
    d. Modify product
    e. Exit as Administrator
2. Customer
    a. Add funds (assume all integer operations)
    b. Add product to the cart

c. Check-out cart
d. Exit as Customer

The program should loop with above two modes unless explicitly terminated. Before termination the program should print the total revenue generated by Amacon.

Question 2

Congratulations! Amacon.com is very pleased with your software Version 1.0 (Question 1 above) and now they have decided to give you another contract to implement the Version 2.0 of the Amacon software. Note that you must use the Amacon.com software you developed in Question 1 to implement this Version 2.0.

In this Version 2.0, you have to extend the Amacon.com platform to make it more user-friendly and to ensure that the system is also foolproof. These new set of requirements from Amacon.com are as following:

1. Version 2.0 platform should welcome recurring customers. It should be able to identify each customer by a unique username (case sensitive) and save the customer's cart state after each session. Platform should allow the returning customer to recall their cart the next time they log in Amacon.com.
2. Version 1.0 (Question 1) platform is extremely impractical for the admins as they have to re-initialise the database before every admin sessions. Version 2.0 platform should maintain the last modified state of its database across all admin and customer sessions.
3. Version 2.0 platform should be robust to corner cases and should never crash. JUnit based unit testing is a good way of testing your platform effectively and comprehensively. You have to design a **JUnit Test Suite** for the Amacon system that contains different JUnit test cases for testing different functionality of Amacon.com Version 2.0 as mentioned below:
    a. Test Cases to validate user defined Exceptions mentioned in Version 1.0 (Question 1):
        i. Inserting the same product at the same path.
        ii. Deleting a product which does not exist.
        iii. Searching for a product which does not exist.
        iv. Adding a product to the cart which is out of stock
        v. Checking out the cart when the customer is out of funds
    b. Test Cases related to database modifications in Version 2.0:
        i. Test the serialization of database
        ii. Test the serialization of customer cart

About Junit - https://www.codejava.net/testing/junit-test-suite-example-how-to-create-and-run-test-suite-in-command-line-and-eclipse

Question 3.

A new banking service has just started in India and the name of this bank is **Serious Bank of India** (SBI). They have an online portal where the customers can login and transfer their money from one SBI account to another SBI account. Lately customers have started complaining to SBI that they have to wait for several minutes for their online transfers to complete. SBI's IT team has investigated this complaint and found that the bottleneck is due to slow processing of transactions. At any given time there are millions of transactions waiting to happen but their software performs all the transactions sequentially. They have approached you to help them in parallelizing their software.

Their requirements are as following:

1) There are **10000** accounts in their bank. You should create all the accounts as soon as the banking software is launched. SBI don't want to alter their existing implementation of Account class. Assume that none of the methods inside Account class will be using any multithreading features.

2) A transaction is performed only between two different account holders. A transaction is done by first debiting the amount from the sender's account and then crediting that amount to the destination's account. Once the amount is credited to the destination account, only then the transaction is said to be completed.

3) At any given time there are exactly **10 million** pending transactions in SBI. You can simulate this by first randomly creating all pending transactions before processing those transactions.

4) Once all the transactions are created, the computation required to complete these transactions should be done in parallel. You **must use the best possible parallelization strategy** in your implementation. Your implementation must be able to process pending transactions faster by increasing the thread count.

5) After all the transactions are completed, you must verify if the above step complete with correctness with any number of threads you use (assume maximum number of threads are four). Report this result with appropriate messages ("Processing PASSED" or "Processing FAILED"). Execution time should only be considered for total time taken to complete step 4 above.

Deliverables:

1) Your implementation of SBI software by using an object oriented approach.
2) Brief explanation of why you chose that particular parallelization strategy.
3) Draw these three graphs for your banking software by using threads = 1, 2, 3, and 4: a) execution time graph, b) speedup graph, and c) parallel efficiency graph. You should upload the PDF of these graphs and upload along with your solution on backpack.

Source: Dr. Vivek