Freshers Assignment 2020 - Mobile

Instructions:

1. Follow the Java coding standard, take reference from the link here
2. Create your own GitHub public repository, push your assignment and share your repository link via email (Reply on the same email where we have given the assignment).
3. Since, all assignments involve unit test cases, please use Junit Test Suite for unit test case development.
4. Use Java version 11 for development.
5. For any doubts and queries, write your queries in the whatsapp group only.

Part 1 - Java assignments

ASSIGNMENT #1

Topics Covered: Java I/O, String.

Write a java program that accepts details (item name, item type, item prize) of different items from

Command line and outputs the item name, item prize, sales tax liability per item, final prize (sales tax + item prize) to the console. The input should be accepted with following command line options:

-name <first item name>

-price <price of first item>

-quantity <quantity of first item>

-type <type of first item>

The following functionalities/behavior is expected:

- All options other than item name can come in any order i.e. after -name you can have -price, -type option. Item type is a mandatory option.

- The system should provide functionality to add more than one items i.e. Once the details of the first item is entered it should display a message saying:

  Do you want to enter details of any other item (y/n):

  Appropriate behavior is required to be implemented as per user input for this question.

- Make use of java's object oriented capabilities for implementing this business logic.

- Item type can have 3 possible values raw, manufactured and imported.

- Tax rules for the 3 types are as follows:

  raw: 12.5% of the item cost

  manufactured: 12.5% of the item cost + 2% of (item cost + 12.5% of the item cost)

imported: 10% import duty on item cost + a surcharge (surcharge is: Rs. 5 if the final cost after applying tax & import duty is up to Rs. 100, Rs. 10 if the cost exceeds 100 and up to 200 and 5% of the final cost if it exceeds 200).

Real Life Scenario:

Inventory management system.

Key Points:

1. Use Java's I/O capabilities to accept input from users.
2. Use Java's String functionalities to parse input strings.
3. Coding conventions should be followed.
4. Proper validation / info messages should be thrown on console.
5. Do appropriate exception handling wherever required.
6. Where ever required please write comments in the code to make it more understandable.
7. TDD methodology should be used

ASSIGNMENT #2

(1 days)

Topics Covered: Java Serialization, Sorting.

Write a menu driven command line java program that provides the following menu options:

1. Add User details.
2. Display User details.
3. Delete User details
4. Save User details.
5. Exit

The option (1) should ask for the following user details. All the following details are mandatory and the program should perform the required validations (like blank data, integer value for age, roll number etc). Roll Number is a key to identify the uniqueness among the students.

1. Full Name
2. Age
3. Address
4. Roll Number
5. Set of courses he is interested to enroll. There are a total of 6 courses (Course A, B, C, D, E and F). It is mandatory for each student to choose 4 out of 6 courses.

Once the validations are passed the user details should be added to an in memory data structure. The data structure should always keep the records sorted in ascending order. By default the records should be sorted on full name. If name is same for two students then sorting should be based on the roll number.

The option (2) should display the user details in the following format. Also the user should be provided with an option to sort the results (either in ascending or descending order) based on name, roll number, age, address.

Name	Roll Number	Age	Address	Courses

A	43	1	22 A, GGn	A, C, D, E

The option (3) should ask for roll number and delete the student details corresponding to that roll number. Throw a proper user friendly message in case the roll number entered by the user does not exist.

The option (4) should save the in memory details of all the users to a disk. Use java's serialization capabilities to serialize the in memory data to disk. If the user terminates the program after choosing this option the user's data should be saved to disk and next time the user runs the program the in memory collection should be pre populated with data already stored on the disk.

The option (5) should terminate the program but before termination it should ask the user if he wants to save his latest changes (additions, deletions of users) to disk.

Key Points:

1. Use Java's serialization mechanism to save user details to disk.
2. Use Java's comparable and comparator interfaces for sorting.
3. Coding conventions should be followed.
4. Proper validation / info messages should be thrown on console.
5. Student Info, course info, serialization code and command line menu code should be encapsulated in separate independent java classes.
6. Where ever required please write comments in the code to make it more understandable.
7. TDD methodology should be used

ASSIGNMENT #3

Topics Covered: Java Collections.

Design a Data Structure using Java's Collection Framework that represents a dependency graph.

Dependency Graph is an acyclic multi root directional graph with the exception of a root node, which has no parents.

Real Life Scenario:

Family Tree

Terminology used:

Parent: For edge A->B, A is a parent of B. There may be multiple parents for a child.

Child: For edge A->B, B is a child of A. There may be multiple children of a parent.

Ancestor: parent or grand-parent or grand-grand-parent and so on

Descendant: child or grand-child or grand-grand-child and so on

Basically the data structure should allow you to store the parent child relationship and this can go to the nth level.

Design:

The node information, which we will store, is:

Node Id --- This has to be unique.

Node Name. Need not be distinct.

Additional Information --- In the form of a key value pairs and this can be different for each node.

Operations:

Get the immediate parents of a node, passing the node id as input parameter.

Get the immediate children of a node, passing the node id as input parameter.

Get the ancestors of a node, passing the node id as input parameter.

Get the descendants of a node, passing the node id as input parameter.

Delete dependency from a tree, passing parent node id and child node id.

Delete a node from a tree, passing node id as input parameter. This should delete all the dependencies of the node.

Add a new dependency to a tree, passing parent node id and child node id. This should check for cyclic dependencies.

Add a new node to tree. This node will have no parents and children. Dependency will be established by calling the 7 number API.

Key Points:

Use Java's collection framework to implement Family Tree.

Proper validation / info messages should be thrown on console.

Do appropriate exception handling wherever required.

Where ever required please write comments in the code to make it more understandable.

TDD methodology should be used

ASSIGNMENT #4

Topics Covered: Java Multithreading.

Develop a multi-threaded java program where one thread reads the data from the database say, details of an Item from a mysql table. This thread builds an in-memory object, stores it in a collection. Simultaneously another thread should fetch already created Item objects from this collection and calculate the tax as per rules detailed in assignment#1 update the tax value in appropriate Item attribute and store it in a different collection. Finally print out the item details to console as detailed in assignment #1.

Implement such that the performance is optimal and thread race/deadlock is avoided.

Real Life Scenario:

Producer consumer mechanism.

Key Points:

Please make sure your database is setup and you are able to access it before starting with implementation of this assignment.

Use Java's multithreading support for implementation.

Proper validation / info messages should be thrown on console wherever required.

Do appropriate exception handling wherever required.

Where ever required please write comments in the code to make it more understandable.

TDD methodology should be used

Part 2 - Building an Application

ASSIGNMENT #5

Contact Application

Create a contact application (Android) with following features:

1. User can add a new contact with following details
    a. Full Name
    b. Contact number
    c. Email
    d. Company information
    e. Image
2. User can view a list of all contacts in the device
3. User can update existing contacts
4. User can delete a contact
5. UI can be customer as you like.

Use the following technologies/ Libraries in Android

- RxJava2
- Dagger2
- Conductor