**Case Study 1**

METRO SYSTEM

Design and develop an application for central metro system. The application needs to provide the swipe in and swipe out functionality.

Swipe in refers to boarding the station. Swipe out refers to coming out of the station. Every metro station has both swipe in and out facilities. There will be a metro card issued to every user with a minimum balance of Rs 100.

The metro line consists of 5 linear stations as mentioned below.

Instructions –

A. When a new user is created, accept user’s basic details along with the card balance to issue the card to the user.

B. For Swipe in functionality, the application should

1. Accept the user’s input as source station.

2. The station can be from the above list only. Create a custom exception (with a meaningful message to the user) to handle invalid station inputs.

3. Validate the minimum required balance in the card. The user should have minimum balance of Rs 20 in the card. If the balance is not there, throw custom exception with appropriate message to user and do not allow to swipe in.

4. On successful swipe in, which means if the minimum balance is present then print the message as “You have successfully swiped in at the station” + <Source Station Name>.

C. For Swipe out functionality, the application should

1. Accept the user’s input for destination station.

2. The station can be from the above list only. Create a custom exception (with a meaningful message to the user) to handle invalid station inputs.

3. Calculate the total fare based on source and destination stations and deduct the fare from the card balance. The fare calculation is based on the below rules.

a. Fare between any 2 adjacent stations is Rs 5. Example – Fare between L1 and L2 is Rs 5 and fare between L2 and L3 is Rs 5.

4. After deducting balance, the message needs to be printed “You have successfully swiped out with card balance as” + <actual card balance>

D. Show the output on the console.

E. Handle appropriate exceptions with appropriate message whenever required.

F. The design should be flexible enough so that in future, more stations and different fare calculation methods can be added.

=====================================================================

**Case Study 2:**

A company named XYZ Retail is in the business of selling:

· Books

· CDs

· Cosmetics

Books are sales tax exempt whereas CDs have a sales tax of 10% and Cosmetics have sales tax of 12%. Design a shopping basket application, which will calculate total price inclusive of taxes using the formula ((quantity \* unitPrice) + tax) where tax = (quantity \* unitPrice) \* (salestax/100).

Assumptions –

1. Have the initial number of items/stock hardcoded for every item in the system.

2. Have the unit price of every item hardcoded in the system.

Instructions –

The application should

1. Take user input as to how many Books, CDs and Cosmetics he wants to purchase.

2. Calculate the price per category of item and then add it up to give the grand total of the shopping basket (which is inclusive of taxes).

3. Display the final bill which should contain the following information.

· Type of item

· Quantity of item

· Price of each item set in the basket

· Grand total amount

4. Show the output on the console.

5. Throw custom exception with appropriate error message if the quantity entered is negative or more than the initial stock available for that item.

6. Handle appropriate exceptions with meaningful messages whenever required.

7. Usage of database required for this application.

8. The design should be flexible enough so that in future, more type of items can be added and sales tax can vary.

**Case study 3**

You have to design a Library management system for your company. The employees can get different types of books issued from library.

There are three types of books available in the library. All the books can be issued for 7 days only.

1. Data Analytics – The late fee is Rs 5 per day.

2. Technology – The late fee is Rs 6 per day.

3. Management – The late fee is Rs 7 per day.

The application should provide “Return Book” functionality.

Assumptions –

1. Date api can be used.

2. Return date can be taken as today’s date by default.

Instructions –

For return book, the application should

1. Take input as employee details, type of book to be returned and issue date of that book.

2. Based on number of days between issue date and today’s date, if the employee is late in returning the book then calculate late fee as per the rules mentioned with type of book else late fee will be 0.

3. For the positive case when employee is not late in returning the book, print the message as “There is no late fee applicable and the book has been returned”.

4. Display the details with the following information.

· Name of Employee

· Type of Book

· Issue Date

· Return Date

· Late Fee

5. Show the output on the console.

6. Throw custom exception with appropriate message if issue date entered is after the today’s date. Example – Issue date is 16-Jun-2015 and today’s date is 14-Jun-2015.

7. Handle appropriate exceptions with meaningful messages whenever required.

8. Usage of database required for this application.

9. The design should be flexible enough so that in future, more type of books or new functionality can be added.