**Assessment**

Internal Assessment

Objective/ Vision

Code for a bus booking system where you can read the bus related information from CSV files and create and manage bookings for customers.

Functional Requirements

* Code a BusMaster class with the attributes which are fields from the input file Bus-master.csv [CSV files are just plain text files with comma separated values. You can use your buffered input streams / buffered readers to read a line at a time and split them].
* Code a BusDetail class with the attributes which are fields from the input file Bus-detail.csv. This should be mapped to the BusMaster.
* The BusMaster and BusDetail classes should have annotation-based validations. The rules for validation are: -   
    
  Bus No – Should have 3 numbers first followed by 1 alphabet (Ex – 215A is **valid**, 2122W is **invalid**)  
    
  Number of Seats (in any field) should be a positive integer.  
    
  **Hint: -** Use custom annotations (in POJO class) in combination with Jakarta validation APIs (Google is your best friend)
* If any data field in a row is invalid, it should be logged in an **error.txt** file and the subsequent rows should be processed.
* Write JDBC code to write the details into BusMaster and BusDetail tables in your local H2 DB (in-memory mode).  
    
  BusMaster table would have all the header fields of BusMaster.csv as its column and the rows of the CSV file would be the rows of the table.  
    
  BusDetail table would have all the header fields of BusDetail.csv as its column and the rows of the CSV file would be the rows of the table.

The class definition of the BusMaster should be:

*class BusMaster {*

*<<attribute1>>*

*<<attribute2>>*

*….*

*BusDetail busDetail;*

*<<all getters and setters>>*

*<<toString()>>*

*}*

* Create a PassengerInfo class, as per the description below:

class PassengerInfo {

long passengerId;

String name;

char sex;

int age;

String startingPoint;

String endingPoint;

Date travelDate;

boolean isSpecialSeatNeeded;

boolean isWomanOnlySeatNeeded;

}

Write getters and setters and override toString() as well. Also create at least 6 instances of PassengerInfo to start with.

* Create a Reservation class which has the following attributes:

*class Reservation {*

*int seatNumber;*

*PassengerInfo passengerInfo;*

*BusMaster busMaster;*

*}*

Write getters and setters and override toString() to show an output as shown below:

From: Chennai To: Madurai Date: 02-Aug-2019 Time: 20:30 Seat No: 7

Passenger Name: Ramkumar Age: 34 Sex: Male Seat Preferences: Normal [other values may be Woman Only or Special].

* Create a class ReservationManager which is more of a utility class and has the following attributes:

class ReservationManager {

List<BusMaster> busMasterList;

List<PassengerInfo> currentPassengerList;

List<Reservation> currentReservationList;

}

Add getters and setters for this class.

Code for the following methods in the ReservationManager class.

**Write code and Test-cases for the following requirements**

* 1. Create a method writeCsvDetailsToDB(). This method should write the valid row details of the aforementioned CSV files into their relevant tables.
  2. Create a method loadBusMasterList(). This method should read from the Bus Master table and create a list of BusMaster objects. This method should also call another method you will code – loadBusDetailList() that will read from Bus Details table and associate the correct BusDetail object to the BusMaster object [look for the busNumber attribute in both objects for this]. The return type of both methods is void.
  3. Write a method to reserve ticket as follows:

*public Reservation bookTicket*(*PassengerInfo passengerInfo*);

* + 1. This method should take a passengerInfo instance and check if it is already available in the currentPassengerList available. If it is not, then you need to add this PassengerInfo list to the currentPassengerList first.
    2. After that, you need to check if there is any bus available based on the travelDate, startingPoint, endingPoint and seat requirements from the PassengerInfo object and check if a bus is available for the passenger requirement. If it is not available, return NULL. Else, if you find a bus available and seat exists as per the passenger preference, you should create a Reservation object the structure of which is given earlier. Assign a seat number and populate this Reservation object and return that.
    3. Reduce the available seat [common / women only / special] in that bus by 1 after creating a Reservation.
    4. Write Junit test-cases for:
       1. Booking a common ticket
       2. Booking a woman only ticket
       3. Booking a special ticket
       4. Booking not possible due to lack of seat
       5. Booking not possible due to lack of any bus from starting point to ending point.
  1. Write a method to cancel the ticket booked. The method signature is as follows:

***public void cancelTicket(Reservation reservation)***

This method should check if the reservation is present in the currentReservationList of the ReservationManager. If so, it should delete / remove that instance and update the available seat by 1 in the respective category and returns “Reservation Cancelled”. If the reservation is not available, it should display, “Reservation Doesn’t Exist”.

Write test-cases for both scenarios.

* 1. Write a method getAllReservationsForBus(String busNumber) to return a list of reservation objects, based on the bus number passed as input. Write test-cases for both positive and negative scenario.
  2. Write a method suggestAlternateTravelPlan(PassengerInfo passengerInfo). This method should return a collection of BusMaster objects. The logic is as follows:
     1. Take the starting point and ending point of travel from the PassengerInfo object. Check if there is any bus that matches this requirement from bus master. If there is a match, create a list of BusMaster, add that instance first.
     2. If you find a match only for the starting point, search for the ending point from customer against the other BusMaster objects to find a match. If you find it, then add the first BusMaster instance where you had only the starting point matched and add the second one, which had the ending point that matched. For example, if a passenger wants to go from Chennai to Madurai, the following ways are possible.
        1. **146A** **Chennai** **Madurai** **02-Aug-2019** **20:30 and**
        2. **201A** **Chennai** **Trichy** **04-Aug-2019** **20:45**
        3. **201C** **Trichy** **Madurai** **05-Aug-2019** **04:00**
  3. Write a method writeAllReservationsToFile() which should take the currentReservationList attribute and write that data in to a file called “Reservation.csv” [Hint – write a method similar to the toString() method and use the delimiter as “,”]. The heading and the first line of file should be similar to the one shown below:

BusId, From, To, Date, Time, Pass Name, Age, Sex, Seat Number, Seat Preference

122S, Chennai, Bangalore, 01-Aug-2022, Renuka P, 35, F, 12, W

**FAQ:**

* You are expected to follow Test Driven Development.
* You are expected to submit the assessment on or before 13th Oct 2021. Push the code to GitHub and share the repository link in email to Hema (shemama2@ford.com).