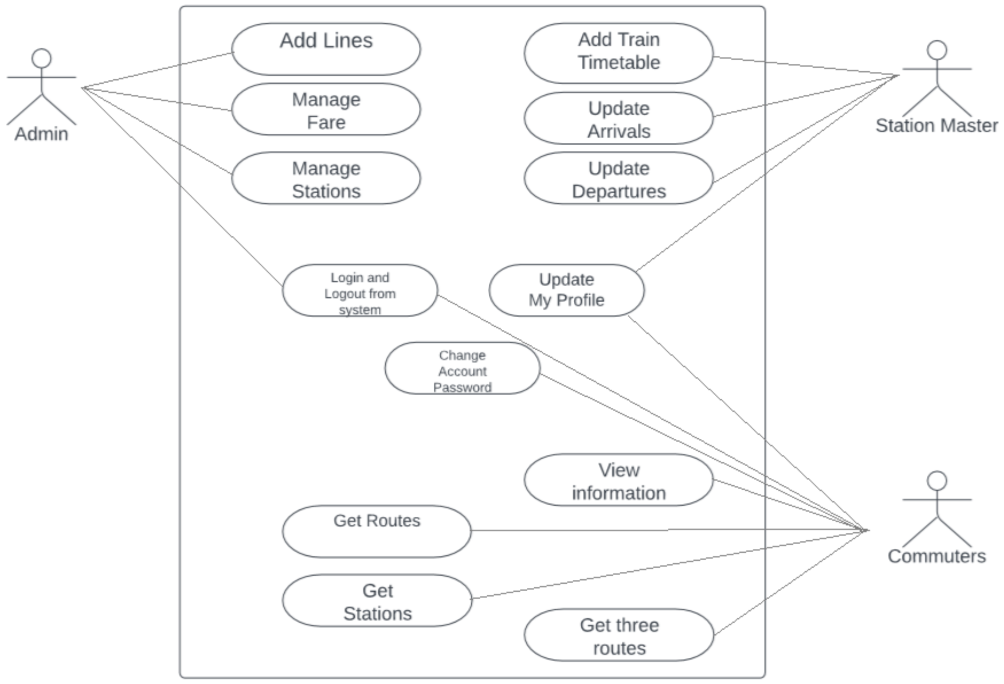
Question 1

(a) Formulate a use case diagram to depict the MRT application design.

Your use case diagram should show the following:

(i) The actors
Administrator: this actor will manage all the activities and will be super user can add lines, add stations also manage fares update fare.
Commuter: this actor will be ultimate user of the application and can check for the station details, check routes.
Station Master: this actor will help commuters those are unable to fetch details from application. This actor also can see the details of the stations routes and fares.
System users: this actor will manage and updates details and information needed to be updated.

(ii) The use cases and their relationships



(b) The following is an extract of the requirements documentation of the MRT application:

"The MRT fares are calculated based on distance travelled. This is a convenient way to pay for train travel. For example, when a commuter has travelled 10 km, he is charged $2.50. However, when a commuter enters and exits from the same station, he is not charged since the distance is 0 km. The fare for a route that includes 2 or more lines are calculated based on the sum of the fare on each line."

Analyse the above requirements and identify and explain any TWO (2) ambiguous, incorrect, incomplete or inconsistent inadequacies for system design. Note that in your answers, you are not supposed to use imagination to add anything not mentioned in the requirements. Further, do not include general commentaries in your answer

1) When the commuter is entered into the station he should be charged for the station staying charge so when he tap his card at the gate then only he would deducted the balance from his card.
2) In this way we can solve this problem that he is not travelling any distance then also he is charged for some amount of money for station services, in this way we can also solve the line problem.

Question 2

Develop a structural model for the MRT application design, by submitting your answers to the following:

(a) Complete the class description by identifying classes, their attributes and any hierarchical relationship(s) that would be required for the application.

There will be following classes in the system

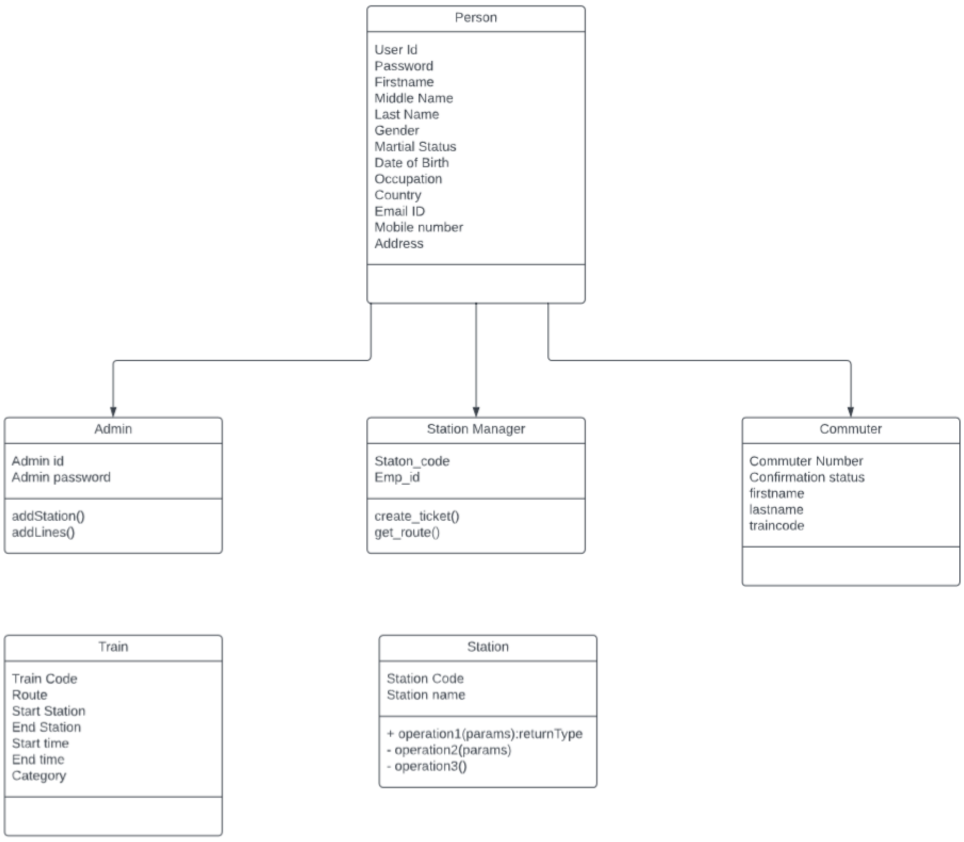
Class Admin
Admin id:int
Admin Password:int

Class Train
Train code:int
Route:string
Start Station:string
End Station:string
Start time:date
End time:date
Category:string

Class Commuters
PNR number:int
Confirmation System:string
First name:string
Middle Name :string
Last Name :string
Age :int
Gender:string
Seat:string

class Station
Station code:int
Station Number:int

(b) Appraise the associations among the classes and hierarchical relationship(s) that would be required for the application. Construct the class association diagram in UML as your answer. Ensure that you do not include any derived or redundant association in your diagram.



Question 3

Further analyses, as well as a few walkthroughs give rise to an updated class diagram, part of which is shown in Figure Q3 below.

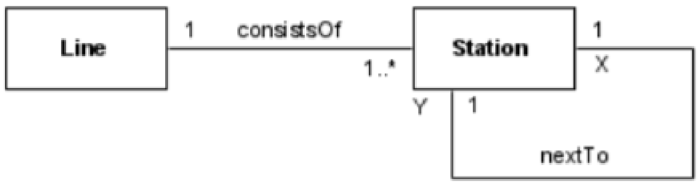


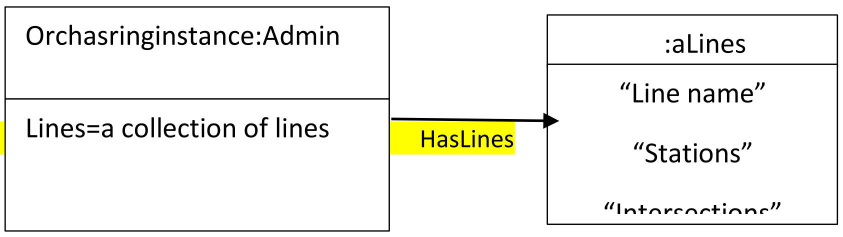
Figure Q3

(a) State the role names, X and Y, of the association nextTo in Figure Q3
Role names X and Y states that X and Y are two stations which are next to each other.
Ans- One instance of station X is associated with one instance of station Y. Also, Station Y is located after station X.

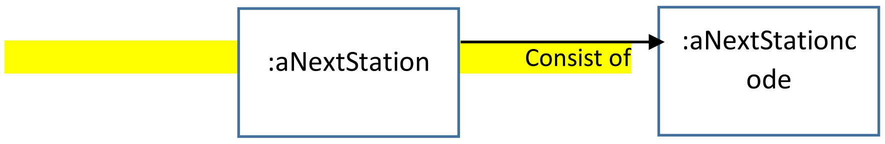
(b) Write an invariant that specifies a station is next to a station which is not itself.
One or more stations situated in a line are located in a distance more than zero km.

(c) Consider the following walkthrough:
Objective: To add a station to a line.
Given: a line name, a station code, station name, next station code, previous station code, distance to next station and distance to previous station.

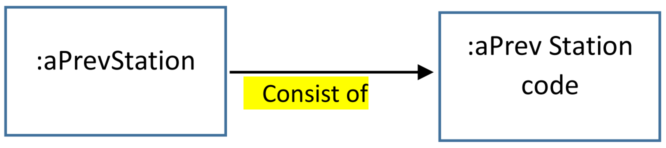
1. Locate the instance, aLine, of Line with the given line name, linked to the orchestrating object via hasLines.



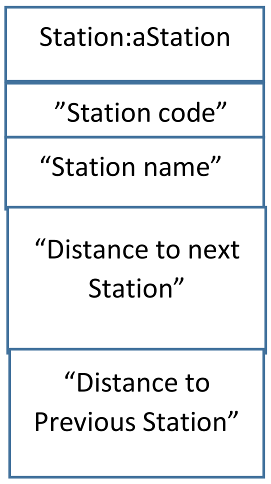
2. Locate the instance, aNextStation object with the next station code, linked to the Line via consistsOf.



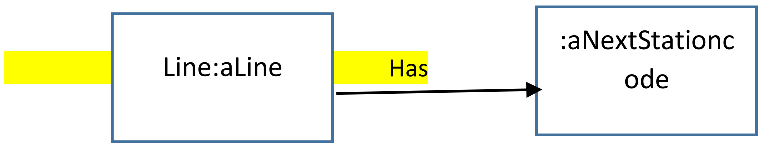
3. Locate the instance, aPrevStation object with the previous station code, linked to the Line via consistsOf.



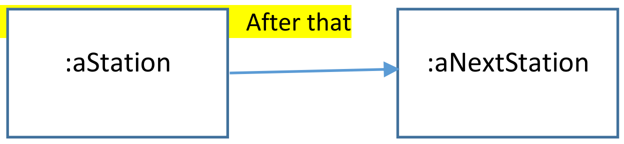
4. Create a new station, aStation object with the station code, station name, distance to next station and distance to previous station.



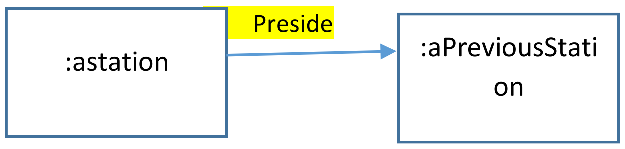
5. Create the association between aLine and the new Station object, aStation.



6. Create the association between the aStation and aNextStation.



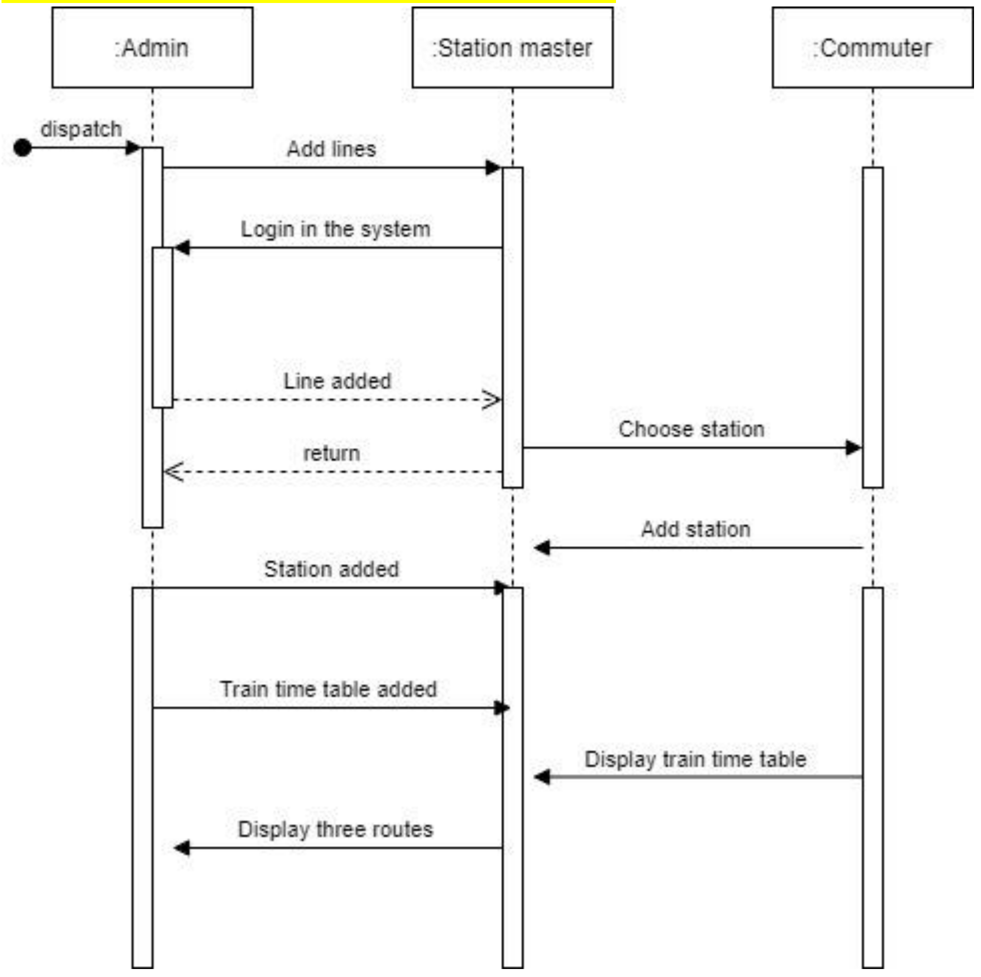
7. Create the association between the aStation and aPrevStation.



(i) Identify the objects in the walkthrough by stating its name and its class.

- Station: aNextStation object
- Admin: Commuter Object
- Station: aPrevStation object
- Station:aStation

(ii) Using the objects identified in Q3(c) (i), develop the dynamic model for the application function, by drawing the sequence diagram for the walkthrough to assign a station to a line.

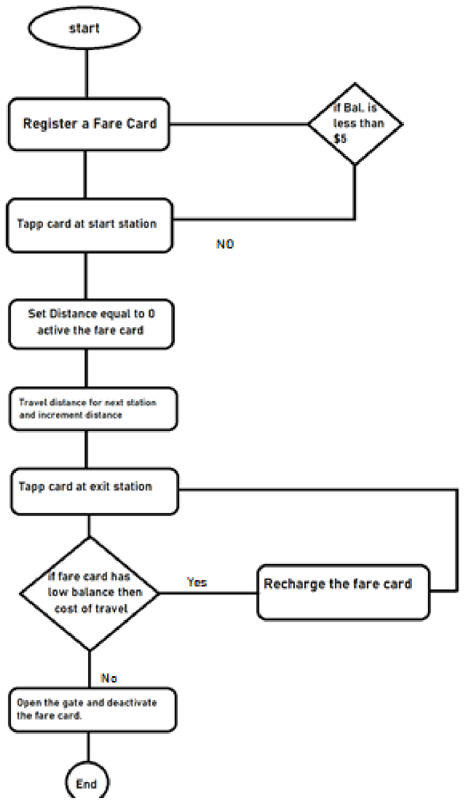


(iii) Implement the method in the orchestrating class that is responsible for assigning the station to the line To add line between any two stations we make use of class Lines. Which has several functions like has line which can check for the existence of line.

```
Class orchestrating
{
i= orchestrating ()
{
i.addStation()
i.addLine()
i.existance_of_line()
```

Question 4

Commuters travelling on the MRT trains can buy a fare card with a minimum stored value of $10. The fare card can be topped up with a minimum of $5 at any time. When a fare card is tapped at a boarding station gate, its status is Active and the distance value in the fare card is set to 0. When a fare card is tapped at an alighting station gate, the MRT application would calculate the distance travelled and deduct the fare from the stored value in the fare card. The fare card then becomes Inactive. If there is insufficient stored value in the fare card, the station gate would not open and the card continues to be Active. The commuter needs to top up the fare card before he can tap at the alighting station gate. A commuter may choose to return a fare card to the ticketing office of any station and the stored value (if any) in the fare card would be refunded to the commuter. Construct a component of the MRT application in UML, by drawing the state diagram for a fare card as it passes through the application.


start
Register a Fare Card
if Bal. is less than $5
Tapp card at start station
NO
Set Distance equal to 0 active the fare card
Travel distance for next station and increment distance
Tapp card at exit station
if fare card has low balance then cost of travel
Yes
Recharge the fare card
No
Open the gate and deactivate the fare card.
End

Question 5

Commuters on the MRT trains can choose to pay for their fares using a stored value fare card, a standard ticket or a concession pass. At an alighting station gate, the respective deduction is effected as follows:

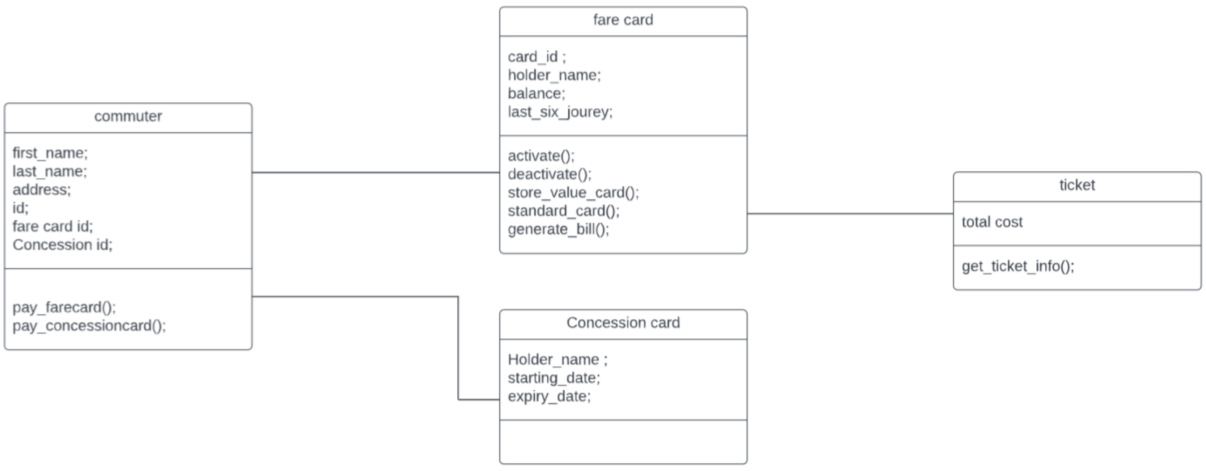
• Stored value fare card – the distance calculated fare is deducted from the stored value in the card. You may assume that there is sufficient stored value in the fare card.

• Standard ticket – each trip is deducted from the ticket. There may be up to 6 trips stored in a ticket. You may assume that there is at least 1 trip stored in the ticket.

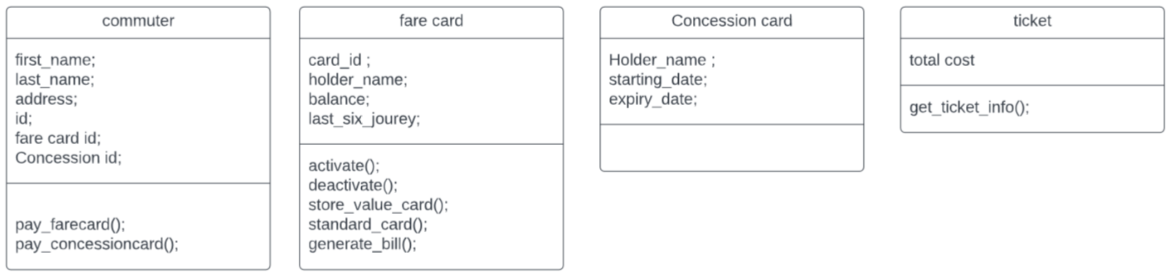
Concession pass – no deduction if valid. Unlimited train rides within a period is allowed. A valid period is 30 days between a start date and an end date (inclusive) stored in the concession pass.

Demonstrate the application of the strategy pattern in the MRT application by submitting your answers to the following:

(a) Develop a structural model of the system that uses the strategy pattern by constructing the class association diagram (including attributes and operations) that allows commuters to pay for their train ride.



(b) Implement the classes for the structural model in Q4(a). For stored value fare card, the amount paid is printed; for standard ticket and concession pass, an appropriate message is to be displayed.



When user has less amount than cost of the journey he should be prompt to recharge the fare card.
But for the concession card he should not be prompted for any message as he has been paid for the entire month before only.
When user has sufficient amount of fund present in the fare card he should be prompted thank you message.

```
Class Commuter
{
String first name;
String last_name
String_address
Int Farecard_id
Int consession_id
{
Pay_farecard()
Pay_consession_card()
}
}

Class Card
{
int card_id;
string holder_name;
float balance;
```

```
string last_six_journey;
{
Activate()
Deactivate()
Store_value_card()
}
}

Card extend Concession_Card
{
date starting_date;
Date expiry date;
}

Class Ticket
{
Int total_cost;
{
getTicket_info();
}

Class Structural
{
Public static void main String args[]
{
Commuter c = new Commuter();
Card C1=new Card ()
Ticket T1= new Ticket()

System.out.println ("Enter amount to be paid")
Scanner sc=new Scanner (system.in)
balance=sc.balance()
if (balance<=0)
System.out.println ("Recharge the card")
Elseif (Card=="concession card")
System.out.println ("Payment already done")
Else
{
System.out.println(“Thank you”)
}
}
```