** Noted that Access Modifier Notations can be listed below*

+ (public)
 # (protected)
- (private)

Set-Up Instruction

- Don't forget to set your Eclipse workspace and working set.
- You must submit the JAR file, exported (with source code), from your Eclipse project.
- You must check your JAR file to make sure all the source files (.java files) are present. It can be opened with file compression programs such as 7-zip or Winrar.

Scoring (14 points)

Question 1

1. Objective

1) Be able to implement Objects and Classes.

2. Instruction

1) Create Java Project (I'm assuming you use Eclipse).
2) Copy all folders in "**Q1_toStudent**" to your project directory **src** folder.
3) You are to implement the following classes (detail for each class is given in section 3 and 4)

    a) **CPTSMachine** (package application)
    b) **Station** (package logic)
    c) **Ticket** (package logic)

4) **JUnit for testing is in package test.**

3. Problem Statement: CPTS Ticket Machine

CPTS Ticket Machine is a JAVA application for the customer to buy one-way trip ticket for CPTS (Coordinated Public Transit System), which is inspired from BTS Sky Train. However, the current system's implementation is still incomplete. You are tasked to implement the remaining Station and Ticket classes.

Here's is how the Ticket Machine should work, part of this is already provided.

When the application is open, there will be a welcome message with all the available commands for the user.

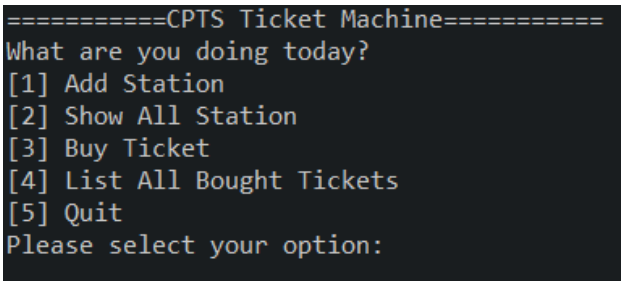

Figure 1. Welcome screen when the application starts

As the current system is still work in progress, the user can choose to add a new station to the system. Please note that the new station name has to be different from other stations. Otherwise, the system must raise an error.

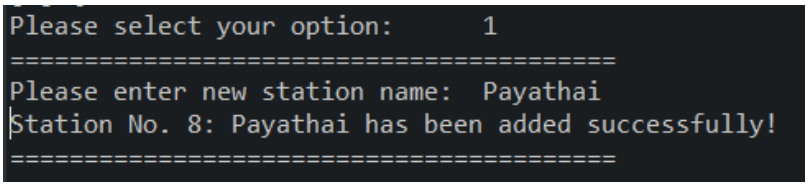

Figure 2. Successfully adding a new station

Users can also choose to list out all the current stations. Each station information displays the station name and its own ID. When first starting, there will already be 8 initial stations in the system.

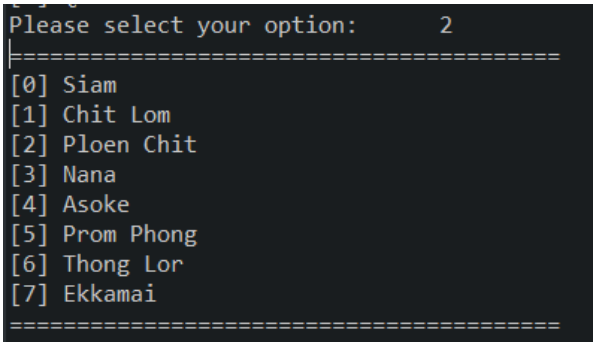

Figure 3. Initial Station Listing

To buy a ticket, the user has to pick the third option from the main menu. Then they will be asked to pick the ticket type, then the starting and ending stations.

There are 3 types of available ticket: **Student, Adult and Elderly**

1. **Student**'s price is 30 Baht per station, has 20% discount if the trip is more than 4 stations.

2. **Adult**'s price is 30 Baht per station and will have regular price calculation, no discount.

3. **Elderly**'s price is 25 Baht per station, has 40% discount. But the ticket cannot be bought if the trip is longer than 6 stations.

The system will raise an error if the ticket is invalid (eg. start and end in the same station.)

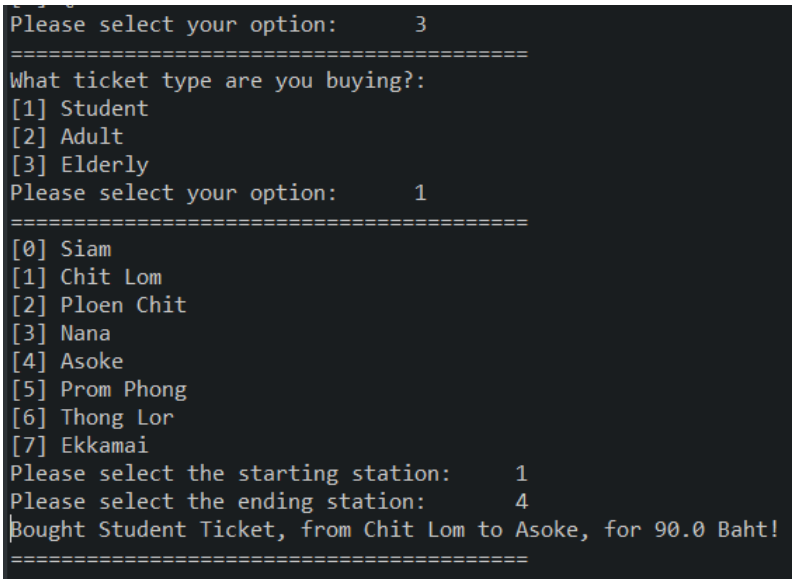

Figure 4. Buying a ticket

Lastly, the user can list out all the tickets they have bought.

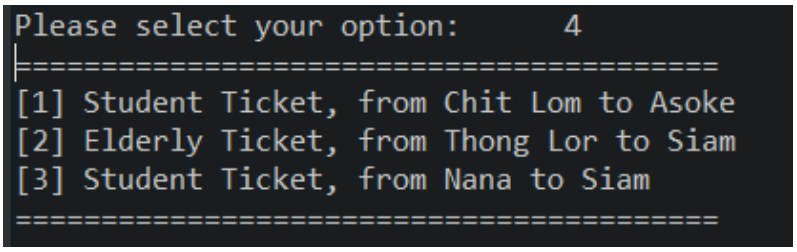

Figure 5. Listing all bought tickets

4. Implementation Detail

The class package is summarized below.

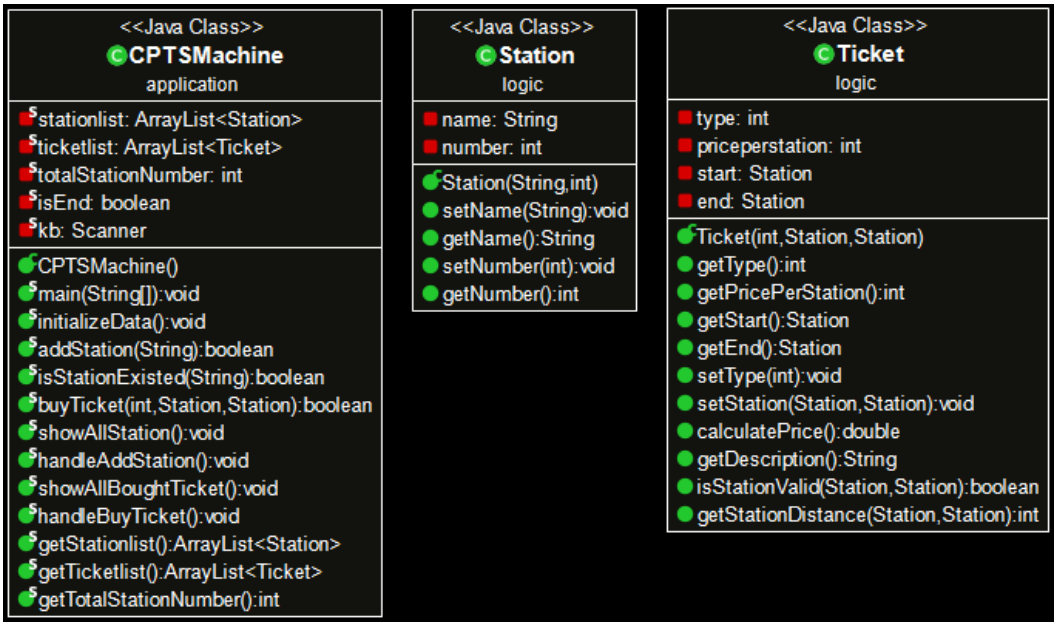

Figure 6. Class Diagram

You must write java classes using UML diagram specified above.

* In the following class description, only details of IMPORTANT fields and methods are given. *

4.1 Package application

4.1.1 Class CPTSMachine: This class represents the machine. It also contains main method. This class is partially provided. You only need to fill the code where necessary. You may consider doing this class after you finish other classes.

Field

Name	Description
- ArrayList<Station> stationlist	ArrayList of Station containing all stations
- ArrayList<Ticket> ticketlist	ArrayList of Ticket containing all tickets you have bought

Method

Name	Description
+ void main()	Handle all ticket machine process
- boolean addStation(String name)	/* FILL CODE */ Adding a new station with the specified name and the new ID (which can be obtained from totalStationNumber) into stationlist. The new station should not have the same name as the other station. This method returns true if the new station is added successfully. Returns false otherwise.
+ boolean isStationExisted(String name)	/* FILL CODE */ Check if the station with the specified name exists or not. This method returns true if the station with the specified name exists. It returns false otherwise.
+ boolean buyTicket(int type, Station start, Station end)	/* PARTIALLY PROVIDED */ Buy a specified type ticket with the specified starting and ending station. - The start and end stations must be the ones in the stationlist. - Type must only be integer from 0 to 2. - For Elderly Ticket, do not forget that the distance between start and end stations must not be more than 6 stations. - If not satisfying the above conditions, the buying fails and the method returns false. Otherwise, the method returns true. Please see handleBuyTicket() for more information on this

	method's usage. Don't forget to also add the bought ticket to ticketlist as well if successful.

4.2 Package logic

4.2.1 Class Station: This class represents the station.

You have to create this class from scratch.

Field

Name	Description
- String name	The name of the station
- int number	internal ID of the station

Constructor

Name	Description
+ Station(name,number)	Create a new Station object with the specified name and number Station number can only be a non-negative integer. Set it to 0 if otherwise.

Method

- getter/setter for each variable

4.2.2 Class Ticket: This class represents the ticket. This one will have a template provided.

You only need to fill the code where necessary.

Field

Name	Description
- int type	the type of the ticket 0 is for Student 1 is for Adult 2 is for Elderly For the other values, the ticket is invalid.

- int pricePerStation	price per station of the ticket
- Station start	starting station
- Station end	ending station

Constructor

Name	Description
+ Ticket(type, start, end)	Create a new ticket object with the specified type, starting and ending station. It uses setters to ensure correct range of field values.

Method

Name	Description
+ void setType(type)	/* FILL CODE */ Set the type of the ticket. If the type is invalid, set the type to 1. This function should also set the proper price per station too (please look at how each ticket type has its price).
+ void setStation(Station start,Station end)	/* FILL CODE */ Set starting and ending stations for the ticket
+ double calculatePrice()	/* FILL CODE */ Calculate the price according to the specification. If the ticket is not valid, returns the price as -1 Hint: You can use isStationValid and getStationDistance given below to help with the implementation.
+ String getDescription()	/* PARTIALLY PROVIDED */ return the string that describes the ticket in the form of "TYPE Ticket, from A to B"

+ boolean isStationValid(start,end)	Check if the starting and ending station is valid or not.
+ int getStationDistance(start,end)	Returns the numeric distance between start and end station
others remaining getter/setter	

Criteria (Test cases that used to mark will be slightly different from the one given in class).

- TestCPTSMachine
  - testIsStationExisted() 1 mark
  - testAddStation() 1 mark
  - testBuyTicketIllegal() 1 mark
  - testBuyTicketLegal() 1 mark
- TestStation
  - testConstructor() 1 mark
  - testSetName() 1 mark
  - testSetNumber() 1 mark
- TestTicket
  - testSetTypeLegal() 1 mark
  - testSetTypeIllegal() 1 mark
  - testSetStation() 1 mark
  - testCalculatePrice() 2 mark
  - testGetDescription() 2 mark