DEPARTMENT OF INFORMATION TECHNOLOGY (TSING YI)

HD in Game Software Development / Telecommunication and Networking

ITP4905 Object Oriented Programming

Assignment

Assignment Due Date and Time:
20 April 2020 Monday 23:50 pm (through Moodle)

Objective

This assignment is to implement a **Train Fare Enquiry System** using Object Oriented Programming Techniques. Your coding must be able to show concepts such as Abstraction, Encapsulation, Inheritance and Polymorphism. Your program must be able to do the following:

1. display the Line and Train information.
2. ask for the user to input the departure station and arrival station.
3. display the available Train(s) based on the user input.
4. ask for the cargo weight for the Cargo Train(s) or the number of passengers for the Passenger Train(s).
5. display the correct total fare amount (including surcharge for Express Passenger Trains – if applicable).

Line and Trains

The followings are typical Line and Trains Information that can be used as data for your **Train Fare Enquiry System** :-

Name of Train Line : HongKong-WuChang-InnerMongolia

Station	Hong Kong	ShenZhen	GuangZhou	ZhengZhou	WuChang	BeiJing	Inner Mongolia
Distance from the first station – Hong Kong	0KM	50KM	200KM	700KM	1000KM	2100KM	2400KM

Trains on the Line:

Type	Code	Start Time	From	To	Max. Quantity per Order
Cargo Train	C001	08:00	ShenZhen	ZhengZhou	100KG
Cargo Train	C002	12:00	Inner Mongolia	ZhengZhou	200KG
Passenger Train	P001	10:00	Hong Kong	Inner Mongolia	10 tickets
Passenger Train	P002	18:00	Inner Mongolia	ShenZhen	6 tickets
Express Passenger Train	X001	14:30	Hong Kong	Inner Mongolia	4 tickets

Train / Station	Hong Kong	ShenZhen	GuangZhou	ZhengZhou	WuChang	BeiJing	Inner Mongolia
C001		→	→	→			
C002				←	←	←	←
P001	→	→	→	→	→	→	→
P002		←	←	←	←	←	←
X001	→	→	→	→	→	→	→

Fare Calculation Rules

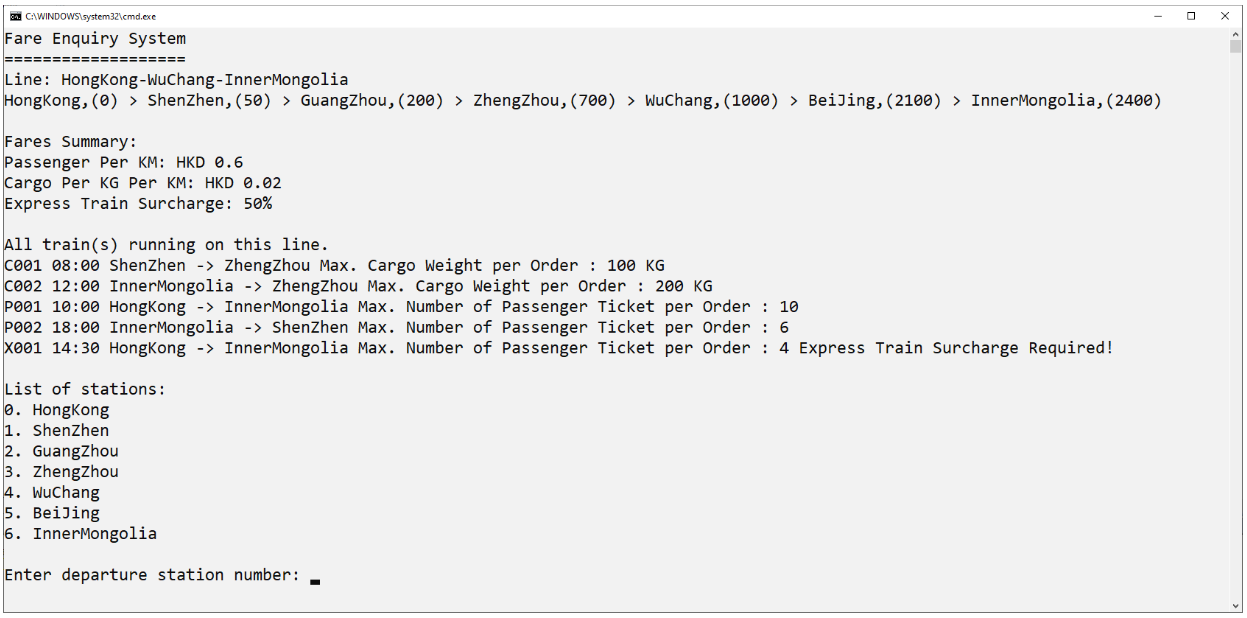
The fare is based on the distance of the journey. Surcharge will be added for Express Passenger Trains. Furthermore, the price for adult passengers and children passengers are the same.

Basic Fare		
Passenger per KM	HKD $0.6	For Example: The distance between Hong Kong and BeiJing is 2100KM. Fare per passenger = 2100KM * $0.6/KM = $1260
Cargo per KG per KM	HKD $0.02	For Example: Similar to the calculation of distance fare for passenger train. The distance between ShenZhen and ZhengZhou is 650 KM (700KM - 50KM). Fare per KG = 650KM * $0.02/KG/KM = $11.0/KG
Surcharge		
Surcharge for Express Passenger Trains (Note : there is no surcharge for Cargo as there is no Cargo Express Train)	50% of the fare for ordinary Passenger Trains	Example: Travelling from Hong Kong to BeiJing on an ordinary Passenger Train P001, the fare per passenger is $1260 as calculated above. Travelling from Hong Kong to BeiJing on Express Train X001, the fare (including surcharge) for each passenger ticket is $1260 * 1.5 = $1890

Requirements of the Assignment

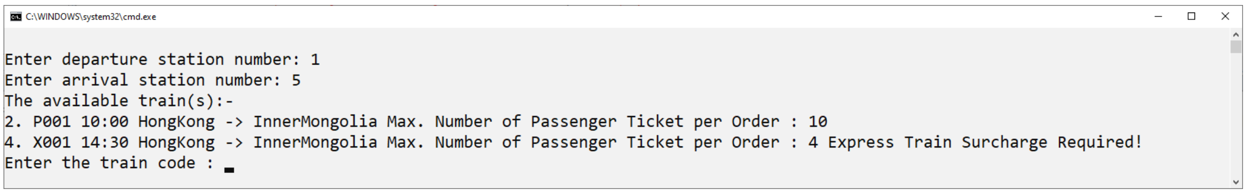
Class FareEnquirySystem – initialize the Line "HongKong-WuChang-InnerMongolia" and Trains information (as above) and provide a user interface to enquire the fare of available train(s) of a specified journey.

When the program starts, Line and Train information are displayed.



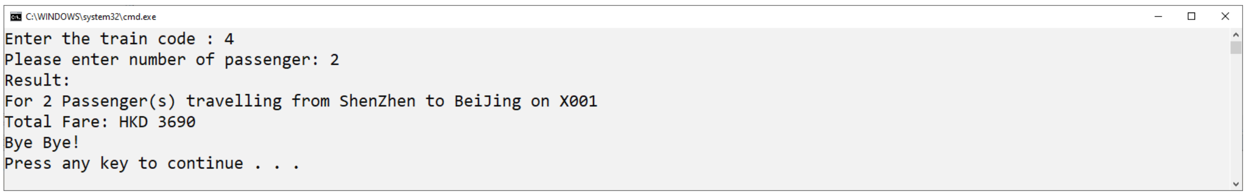
```
Fare Enquiry System
===================
Line: HongKong-WuChang-InnerMongolia
HongKong,(0) > ShenZhen,(50) > GuangZhou,(200) > ZhengZhou,(700) > WuChang,(1000) > BeiJing,(2100) > InnerMongolia,(2400)

Fares Summary:
Passenger Per KM: HKD 0.6
Cargo Per KG Per KM: HKD 0.02
Express Train Surcharge: 50%

All train(s) running on this line.
C001 08:00 ShenZhen -> ZhengZhou Max. Cargo Weight per Order : 100 KG
C002 12:00 InnerMongolia -> ZhengZhou Max. Cargo Weight per Order : 200 KG
P001 10:00 HongKong -> InnerMongolia Max. Number of Passenger Ticket per Order : 10
P002 18:00 InnerMongolia -> ShenZhen Max. Number of Passenger Ticket per Order : 6
X001 14:30 HongKong -> InnerMongolia Max. Number of Passenger Ticket per Order : 4 Express Train Surcharge Required!

List of stations:
0. HongKong
1. ShenZhen
2. GuangZhou
3. ZhengZhou
4. WuChang
5. BeiJing
6. InnerMongolia

Enter departure station number:
```

All stations are displayed and ask the user to input the departure and arrival station numbers. After the departure station and arrival station are input, the system searches for and shows the available train(s).



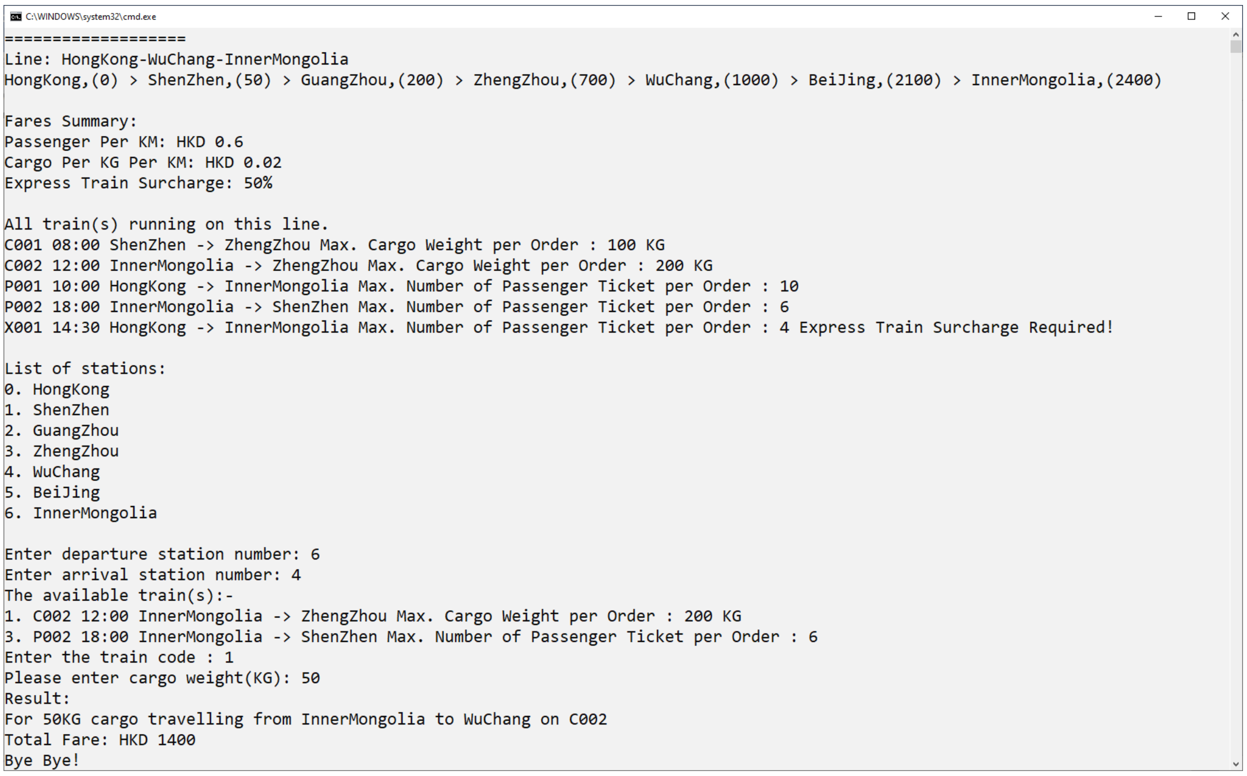
```
Enter departure station number: 1
Enter arrival station number: 5
The available train(s):-
2. P001 10:00 HongKong -> InnerMongolia Max. Number of Passenger Ticket per Order : 10
4. X001 14:30 HongKong -> InnerMongolia Max. Number of Passenger Ticket per Order : 4 Express Train Surcharge Required!
Enter the train code :
```

If a passenger train (including express passenger train) is selected, the system asks for the number of passengers. Lastly, the system prints the calculated total fare (rounded to the nearest dollar) and quit.



```
Enter the train code : 4
Please enter number of passenger: 2
Result:
For 2 Passenger(s) travelling from ShenZhen to BeiJing on X001
Total Fare: HKD 3690
Bye Bye!
Press any key to continue . . .
```

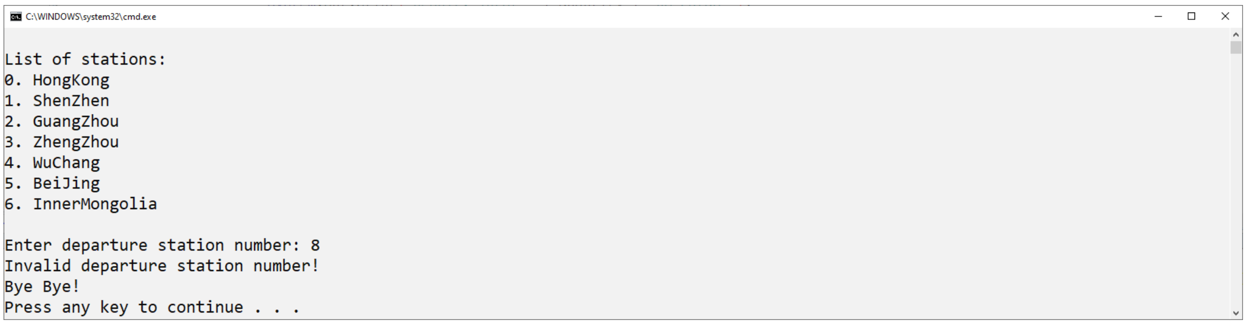
If a cargo train is selected, the system asks for the cargo weight in KG. No decimal place is allowed to be entered for Cargo Weight. Lastly, the system prints the calculated total fare (rounded to the nearest dollar) and quit.



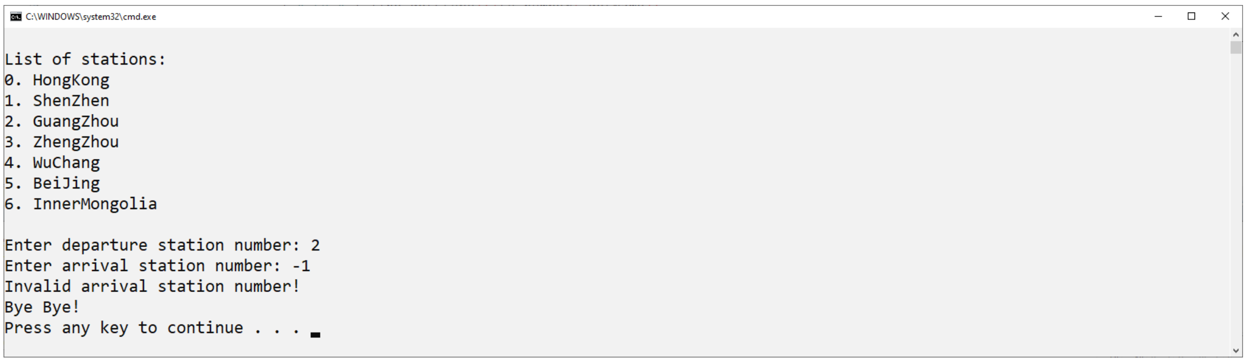
```
===================
Line: HongKong-WuChang-InnerMongolia
HongKong,(0) > ShenZhen,(50) > GuangZhou,(200) > ZhengZhou,(700) > WuChang,(1000) > BeiJing,(2100) > InnerMongolia,(2400)

Fares Summary:
Passenger Per KM: HKD 0.6
Cargo Per KG Per KM: HKD 0.02
Express Train Surcharge: 50%

All train(s) running on this line.
C001 08:00 ShenZhen -> ZhengZhou Max. Cargo Weight per Order : 100 KG
C002 12:00 InnerMongolia -> ZhengZhou Max. Cargo Weight per Order : 200 KG
P001 10:00 HongKong -> InnerMongolia Max. Number of Passenger Ticket per Order : 10
P002 18:00 InnerMongolia -> ShenZhen Max. Number of Passenger Ticket per Order : 6
X001 14:30 HongKong -> InnerMongolia Max. Number of Passenger Ticket per Order : 4 Express Train Surcharge Required!

List of stations:
0. HongKong
1. ShenZhen
2. GuangZhou
3. ZhengZhou
4. WuChang
5. BeiJing
6. InnerMongolia

Enter departure station number: 6
Enter arrival station number: 4
The available train(s):-
1. C002 12:00 InnerMongolia -> ZhengZhou Max. Cargo Weight per Order : 200 KG
3. P002 18:00 InnerMongolia -> ShenZhen Max. Number of Passenger Ticket per Order : 6
Enter the train code : 1
Please enter cargo weight(KG): 50
Result:
For 50KG cargo travelling from InnerMongolia to WuChang on C002
Total Fare: HKD 1400
Bye Bye!
```

The system should provide input error checking including the following cases:

Case #1, Invalid station number inputted.

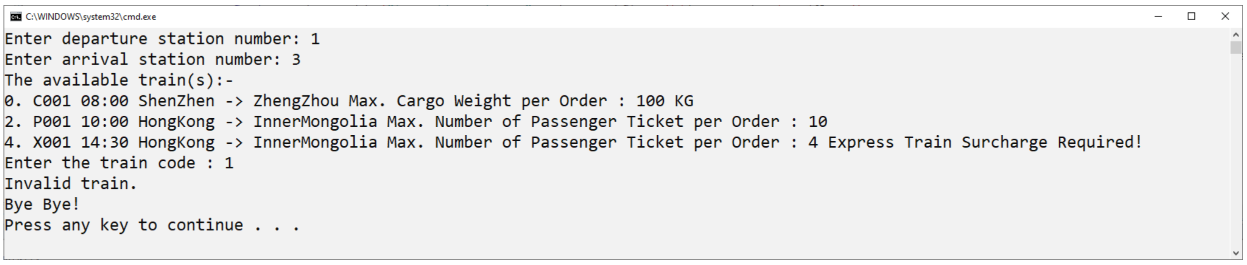


```
List of stations:
0. HongKong
1. ShenZhen
2. GuangZhou
3. ZhengZhou
4. WuChang
5. BeiJing
6. InnerMongolia

Enter departure station number: 8
Invalid departure station number!
Bye Bye!
Press any key to continue . . .
```



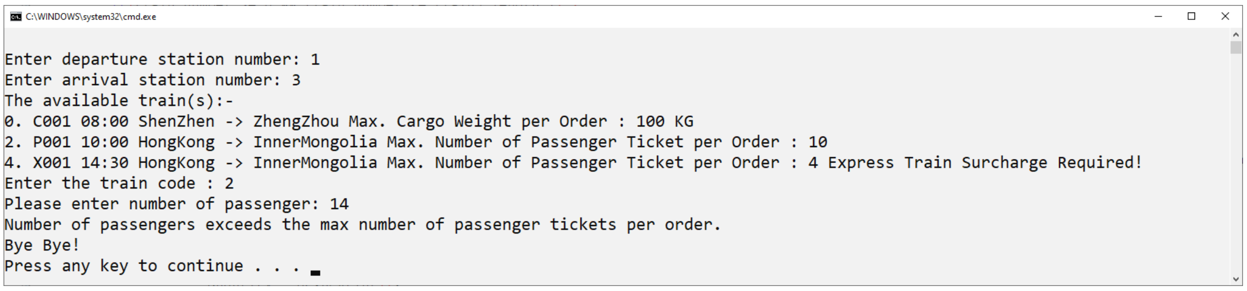
```
List of stations:
0. HongKong
1. ShenZhen
2. GuangZhou
3. ZhengZhou
4. WuChang
5. BeiJing
6. InnerMongolia

Enter departure station number: 2
Enter arrival station number: -1
Invalid arrival station number!
Bye Bye!
Press any key to continue . . .
```

Case #2, Invalid train selected.

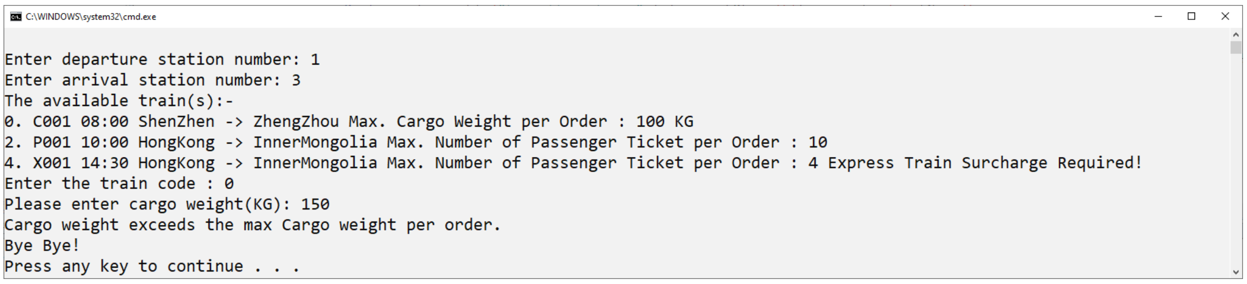


```
Enter departure station number: 1
Enter arrival station number: 3
The available train(s):-
0. C001 08:00 ShenZhen -> ZhengZhou Max. Cargo Weight per Order : 100 KG
2. P001 10:00 HongKong -> InnerMongolia Max. Number of Passenger Ticket per Order : 10
4. X001 14:30 HongKong -> InnerMongolia Max. Number of Passenger Ticket per Order : 4 Express Train Surcharge Required!
Enter the train code : 1
Invalid train.
Bye Bye!
Press any key to continue . . .
```

Case #3, Maximum number of tickets/cargo weight per order exceeded.

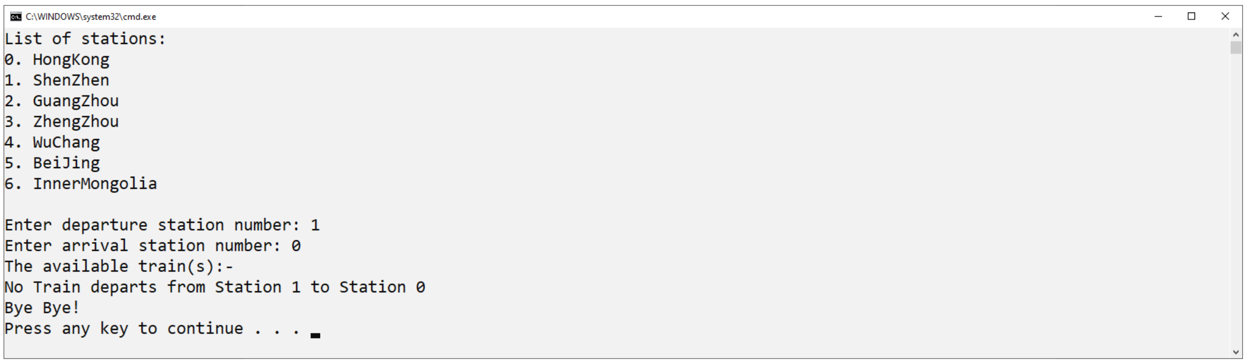


```
Enter departure station number: 1
Enter arrival station number: 3
The available train(s):-
0. C001 08:00 ShenZhen -> ZhengZhou Max. Cargo Weight per Order : 100 KG
2. P001 10:00 HongKong -> InnerMongolia Max. Number of Passenger Ticket per Order : 10
4. X001 14:30 HongKong -> InnerMongolia Max. Number of Passenger Ticket per Order : 4 Express Train Surcharge Required!
Enter the train code : 2
Please enter number of passenger: 14
Number of passengers exceeds the max number of passenger tickets per order.
Bye Bye!
Press any key to continue . . .
```



```
Enter departure station number: 1
Enter arrival station number: 3
The available train(s):-
0. C001 08:00 ShenZhen -> ZhengZhou Max. Cargo Weight per Order : 100 KG
2. P001 10:00 HongKong -> InnerMongolia Max. Number of Passenger Ticket per Order : 10
4. X001 14:30 HongKong -> InnerMongolia Max. Number of Passenger Ticket per Order : 4 Express Train Surcharge Required!
Enter the train code : 0
Please enter cargo weight(KG): 150
Cargo weight exceeds the max Cargo weight per order.
Bye Bye!
Press any key to continue . . .
```

Case #4, No trains available between two stations.



```
List of stations:
0. HongKong
1. ShenZhen
2. GuangZhou
3. ZhengZhou
4. WuChang
5. BeiJing
6. InnerMongolia

Enter departure station number: 1
Enter arrival station number: 0
The available train(s):-
No Train departs from Station 1 to Station 0
Bye Bye!
Press any key to continue . . .
```

Coding Requirements of the Assignment

Class ***Station*** - represents a Station on a Line.

Data Attributes	Data type	Description
name	*String*	a private attribute which stores the name of the Station.
distance	*int*	a private attribute which stores the distance from the starting point Station.
Methods		
Station()	a constructor of the class Station accepts 2 input parameters (name - String and distance – int) and initializes the data attributes accordingly	
a ***getter*** method for **EACH** data attribute		

Class ***Line*** - contains a sequence of stations for a pre-defined train route.

Data Attributes	Data Type	Description
name	*String*	a private attribute which stores the name of the Line.
stops	*Station[]*	a private attribute which stores the sequence of stops of the Line.
passengerFarePerKm	*double*	a private attribute which stores the fare per KM of the Line for one passenger.
cargoFarePerKm	*double*	a private attribute which stores the fare per KM of the Line for one KG of cargo.
expressTrainSurcharge	*double*	a private attribute which stores the percentage of Express Train surcharge when traveling on Express Train per journey for one passenger (for example, 0.5 means 50% surcharge)
Methods		
Line()	a constructor of the class Line accepts five input parameters (name – String, stops – Station [], passengerFarePerKm – double, cargoFarePerKm- double, expressTrainSurcharge – double) and initializes the data attributes accordingly	
a ***getter*** method for **EACH** data attribute		
toString()	returns the information of the Line (return type - String) e.g. "Line: HongKong-WuChang-InterMongolia … Express Train Surcharge: 50%" Hints: You only need to return the Line information and Fares Summary in this method. Station Information (First Station, Second Station, etc…) and Distance between two stations are handled by the main method in the TestLine class Refers to the sample runs for the details and format required.	
calculateDistance()	accepts two stations (from – Station, to – Station) and returns the distance in KM (type - int) between specified stations.	

Note: Refer to test driver program *TestLine.java* and output file *Expected_Output_for_Sample_Test_Programs.docx* for details.

Class ***Train*** **-** an **abstract** class represents a train

Data Attributes	Data type	Description
code	*String*	a private attribute which stores the train code.
startTime	*String*	a private attribute which stores the start time in 24-hour format. For example, "20:00" means 8:00pm.
line	*Line*	a private attribute represents the Line for that train.
start	*Station*	a private attribute represents the first station of the train
end	*Station*	a private attribute represents the last station of the train
Methods		
Train()		a constructor of the class Train accepts five input parameters (code – String, startTime – String, line – Line, start – Station, end – Station) and initializes the data attributes of the Train.
a ***getter*** method for **EACH** data attribute		
calculateFare()		an **abstract** method which accepts two input parameters (from – Station, to - Station) and **returns the fare rate (type - double)**. This method will be overridden by sub-classes' methods as there are different calculations for different types of train.
isValidRoute()		accepts two arguments input parameters (from – Station, to - Station) **returns true or false depending on whether the specified journey is valid or not**.
calculateTotalFare()		accepts three input parameters (from – Station, to – Station, int quantity) and **returns the total fare (type - double)** by the following formula: Total Fare = fare rate * quantity
toString()		**returns the basic information of the Train (return type - String) including the train code, the start time, the first station name and the last station name**. e.g. "C001 08:00 ShenZhen -> ZhengZhou" Refers to the sample runs for the details and format required.

Note: Refer to test driver program *TestTrain.java* and output file *Expected_Output_for_Sample_Test_Programs.docx* for details.

Class ***CargoTrain*** **-** a sub-class of the class Train represents a cargo train

Data Attributes	Data type	Description
maxCargoWeight	*int*	a private attribute which stores the maximum cargo weight allowed for each order.
Methods		
CargoTrain()		a constructor accepts six input parameters (code – String, startTime – String, line – Line, start – Station, end – Station, maxCargoWeight – int) and initializes the data attributes accordingly
calculateFare()		overrides the abstract method of the super-class which accepts accepts two input parameters (from – Station, to - Station) and **returns the fare rate which is the fare per Kg for the journey (type - double)** by the following formula: fare per Kg = fare per KG per KM * distance of the journey. Hints: the value of <u>fare per KG per KM</u> should be obtained from the line of current train object, but not hard-coded
a ***getter*** method for the data attribute ***maxCargoWeight***		
toString()		**returns the basic information of the Train including the train code, the start time, the first station name and the last station name** (by calling the super class's toString method) together with the maximum cargo weight per order (return type - String).

Note: Refer to test driver program *TestTrain.java* and output file *Expected_Output_for_Sample_Test_Programs.docx* for details.

Class ***PassengerTrain*** **-** a sub-class of the class Train represents a passenger train

Attributes	Data type	Description
maxTicketNumber	*int*	a private attribute which stores the maximum number of tickets allowed to be purchased per order for the passenger train.
Methods		
PassengerTrain()	a constructor accepts six input parameters (code – String, startTime – String, line – Line, start – Station, end – Station, maxTicketNumber – int) and initializes the data attributes accordingly	
calculateFare()	overrides the abstract method of the super-class which accepts accepts two input parameters (from – Station, to - Station) and **returns the fare rate which is the fare per passenger for the journey (type - double)** by the following formula: fare per passenger = fare per passenger per KM * distance of the journey. Hints: the value of <u>fare per passenger per KM</u> should be obtained from the line of current train object, but not hard-coded	
a ***getter*** method for the data attribute ***maxTicketNumber***		
toString()	**returns the basic information of the Train including the train code, the start time, the first station name and the last station name (by calling the super class's toString method) together with the maximum number of tickets allowed to be purchased per order (return type - String)**.	

Note: Refer to test driver program *TestTrain.java* and output file *Expected_Output_for_Sample_Test_Programs.docx* for details.

Class ***ExpressPassengerTrain*** **-** a sub-class of the class PassengerTrain represents an express passenger train.

Methods	
ExpressPassengerTrain()	a constructor accepts six input parameters (code – String, startTime – String, line – Line, start – Station, end – Station, maxTicketNumber – int) and initializes the data attributes accordingly
calculateTotalFare()	overrides the method of the super-class which accepts accepts three input parameters (from – Station, to – Station, quantity - int) and **returns the total fare (type - double)** by the following formula: total fare = total fare for Ordinary Passenger Train Fare * (1 + Express Train Surcharge) Hints: the value of <u>Express Train Surcharge</u> should be obtained from the line of current train object, but not hard-coded
toString()	**returns the information of the Passenger Train (by calling the super class's toString method) together with the message "Express Train Surcharge Required!" (return type - String)**

Note: Refer to test driver program *TestTrain.java* and output file *Expected_Output_for_Sample_Test_Programs.docx* for details.

Note :

(i) The classes ***Station***, ***Line***, ***Train***, ***CargoTrain***, ***PassengerTrain*** and ***ExpressPassengerTrain*** must be compatible with test driver programs – ***TestLine.java*** and ***TestTrain.java***. Therefore, when the system is deployed to other lines, only a few lines of source code related to ***Line*** and ***Train*** information in the class ***FareEnquirySystem*** needs to be modified.

(ii) Apart from the above specification, **you may add other methods to the classes if necessary**.

Instructions to Students

1. This is an End of Module Assessment and the weighting of this assignment is 20% of the Module Mark.
2. This assignment should be done by each **individual** student. Plagiarism will be treated seriously. All assignments that have been found involved wholly or partly in plagiarism (no matter these assignments are from the original authors or from the plagiarists) will score Zero mark.
3. You must use Java JDK8 or above to develop the programs.
4. Your programs must follow the coding standard stated in Java coding standard published by Oracle and Sun Microsystems. Marks may be deducted if the coding standard is not followed.
5. You are required to hand in
   - 5.1 A 1 to 2 pages word document briefly explains how your program shows the concept of Object Oriented Programming (Abstraction, Encapsulation, Inheritance and Polymorphism).
   - 5.2 Source code of all classes which should be well-commented.
   - 5.3 The **evidence of testing**.

     Prepare a word document with a number of test cases showing different inputs for different situations that your program may encounter and how your program responses to show the capability of your program.

     For each test case, states the objective of the test case, input data and expected result. You should also include screen dump for each test run as evidence.
6. Submit all your works (in a zip file under the name of your student ID – e.g. 19xxxxxxx.zip) to the Moodle website (http://moodle.vtc.edu.hk) by 23:50, 20 April 2020 (Monday). Late submission may score ZERO mark.
7. Mark Distribution
   - System Implementation and Programming Style (70%)
     - *Line.java*, *Station.java*, *Train.java*, *CargoTrain.java*, *PassengerTrain.java*, *ExpressPassengerTrain.java* and *FareEnquirySystem.java*
   - Validation on the input data and display appropriate error messages (10%)
   - Documentation on the use Object Oriented Programming (10%)
   - Test Plan with test cases and test results. (10%)