Group 15
Final Project: Metro Calculator

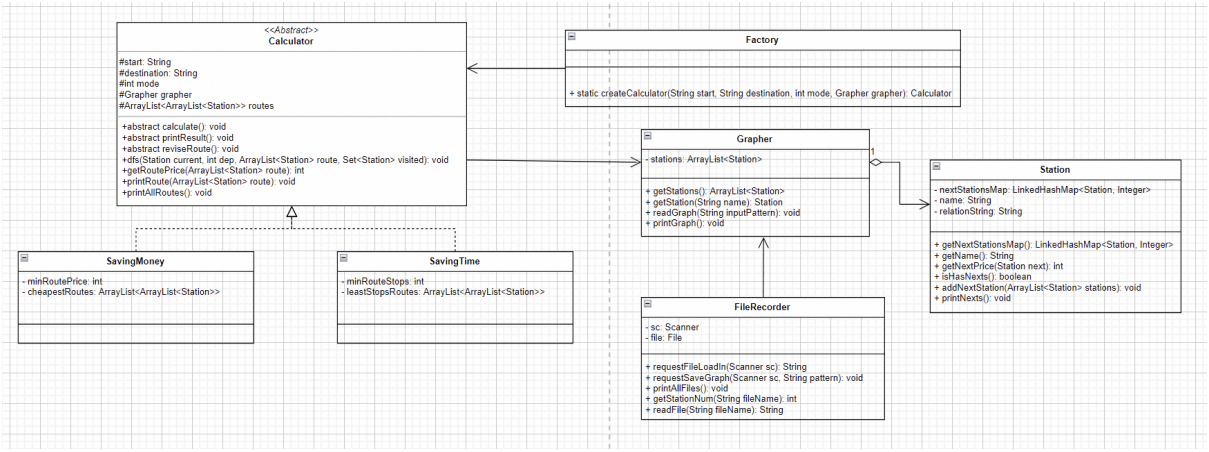
I. Problem Description:

Situation:

1. When we travel to other countries, we get lost frequently. Therefore, we designed a metro calculator that can help us resolve the problems. This calculator assists users in computing the most time-saving routes or the most money-saving routes.

Introduction:

1. In this calculator, we let users input the entire metro graph by using Scanner. Users are required to input the details (exact stations, ticket price, the start station, and the destination).
2. Subsequent to entering the required details, users need to enter whether the mode they want (time-saving or money-saving).
3. Then, the console will show the exact graph, the routes that the user wants, sum price and all stops.
4. After using this calculator, users can decide whether to save the graph or not. The graph will be saved as txt, and it can be accessed next time.



5. We have implemented some class, you just need to implement SavingMoney, Saving Time, Grapher, FileRecorder class

II. Solution Description & Java Documentation

Classes:

1. Create *Calculator* class

Calculator	
Modifier and type	**Method (or Variable) and description**
Instance variable	
String	start Record the departure station.
String	destination Record the destination station.
int	mode Determine which mode users want.
Grapher	grapher Record the graph.
ArrayList	routes Record the routes.
Constructor	
public Calculator(String start, String destination, int mode, Grapher grapher)	
Instance Method	
abstract void	calculate() Override this method in SavingTime class and SavingMoney class.

abstract void	printResult() Override this method in SavingTime class and SavingMoney class.
abstract void	reviseRoute() Override this method in SavingTime class and SavingMoney class.
void	dfs(Station current, int dep, ArrayList<Station> route, Set<Station> visited) Using depth-first-search algorithm to find out all the routes from departure station to destination station.
int	getRoutePrice(ArrayList<Station> route) By the given routes, calculate the sum of the price.
void	printRoute(ArrayList<Station> route) Print out all the routes, the sum of the price, and the number of passing stations.
void	printAllRoutes() Use for-each loop and the printRoute method to print out all the routes.

2. Create *Factory* class

Factory	
Modifier and type	**Method (or Variable) and description**
Instance variable	
Instance Method	
Calculator	createCaculator(String start, String destination, int mode, Grapher grapher)

	Create a null calculator. Instantiate the calculator with mode chosen. (That is, TimeSaving or MoneySaving.)

3. Create *FileRecorder* class

FileRecorder	
Modifier and type	**Method (or Variable) and description**
Instance variable	
Scanner	sc Handle input and output
File	file The file that loaded by the user
Instance Method	
String	requestFileLoadIn(Scanner sc) Request the user whether to load an existing file and use the user's scanner object to read choices.
void	requestSaveGraph(Scanner sc, String pattern) Request the user whether to save the graph.
void	printAllFiles() Print all the existing files if the user chooses to load the existing file.
int	getStationNum(String fileName) Count how many stations in a specific file.
String	readFile(String fileName) Read particular file that user input

4. Create *SavingMoney* class

SavingMoney	
Modifier and type	**Method (or Variable) and description**
Instance variable	
int	minRoutePrice Store the minimum price of all routes
ArrayList<ArrayList<Station>>	cheapestRoutes Store all the cheapest routes in an ArrayList **P.S.** a route is an ArrayList of Station
Constructor	
SavingMoney(String start, String destination, int mode, Grapher grapher) Extends the Calculator class, so must call super() first. Instantiate the two instance variables.	
Instance Method	
void (override)	calculate() Override the calculator's calculate method to calculate the cheapest route.
void (override)	printResult() Print all the cheapest routes stored in the cheapestRoutes. The pattern must follow the sample output.
void (override)	reviseRoute() Revise the cheapest route because some routes may not match the cheapest price depending on your calculate() algorithm.

5. Create *SavingTime* class

SavingTime	
Modifier and type	**Method (or Variable) and description**
Instance variable	
int	minRouteStops Store the minimum stops of all routes
ArrayList<ArrayList<Station>>	leastStopsRoutes Store all the least numbers of stops routes in an ArrayList **P.S.** a route is an ArrayList of Station
Constructor	
SavingMoney(String start, String destination, int mode, Grapher grapher) Extends the Calculator class, so must call super() first. Instantiate the two instance variables.	
Instance Method	
void (override)	calculate() Override the calculator's calculate method to calculate the least stops route.
void (override)	printResult() Print all the least stops routes stored in the cheapestRoutes. The pattern must follow the sample output.
void (override)	reviseRoute() Revise the least stops route because some routes may not match the least stops depending on your calculate() algorithm.

6. Create *Grapher* class

Grapher

Modifier and type	Method (or Variable) and description
Instance variable	
ArrayList<Station>	stations Store all the stations in the graph
Constructor	
Grapher() Instantiate the stations variable.	
Instance Method	
ArrayList<Station>	getStations() Return the stations instance variable
Station	getStation(String name) Return the specific station in stations depends on the name that user input
void	readGraph(String inputPattern) Read the user input graph. The inputPattern must follow the input format.
void	printGraph() Print the graph matching the pattern in the example output.

7. Create *Staion* class

Station	
Modifier and type	**Method (or Variable) and description**
Instance variable	

String	name The name of stations.
String	relationString The String typed in by users which contains the stations that a specific station can get to ***eg***. A: B(20), C(10) then the relationString is " B(20), C(10)"
LinkedHashMap<Station, Integer>	nextStationsMap A LinkedHashMap that store the stations and their price according to relationString
Constructor	
public Station(String name) Instantiate an object of Station with a given *name,* initialize *nextStationsMap* and set *relationString* as an empty string.	
public Station(String name, String relationString) Instantiate an object of Station with a given *name* and *relationString*, initialize *nextStationsMap.*	
Instance methods	
String	getName() Getter of the name attribute
int	getNextPrice(Station next) Return the price of the input station ***Hint:*** the method LinkedHashMap.get(Station) will return the corresponding integer
LinkedHashMap<Station, Integer>	getNextStationsMap() Getter of nextStationsMap
boolean	isHasNexts() Determine whether nextStationsMap is empty or not. If it's not

	empty, return true, if it is, return false. ***Hint***: you can use the reverse of the method “isEmpty()”
void	addNextStation(ArrayList<Station> stations) Store the price of a specific station to nextStationsMap
void	printNexts() print the next stations which start from a selected station, the data is come from the metro graph that users type in

8. Create *Tester* class

   (Paste it to eclipse, you cannot change the code below.)

Code

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Scanner;
public class Tester {
     public static void main(String[] args) throws IOException
{
          /*
               example:
                    台北車站: 西門(20), 忠孝新生(20), 中正紀念堂
(30)
                    西門: 台北車站(20), 中正紀念堂(10)
                    忠孝新生: 台北車站(20), 東門(10)
                    東門: 忠孝新生(10), 中正紀念堂(10)
                    中正紀念堂: 西門(10), 台北車站(30), 東門(10)
           */

          //initialize
          Scanner sc = new Scanner(System.in);
          int stationNum = 0;
```

```
            String pattern = "";
            Grapher grapher = new Grapher();
            String startStation = "", destinationStation = "";
            int mode;
            Calculator calculator;
            FileRecorder recorder = new FileRecorder();

            System.out.println("-".repeat(40));
            String welcome = "Welcome to Metro Calculator ! ! !";
//32格
            System.out.printf("%36s\n", welcome);
            System.out.println("-".repeat(40)+"\n");

            String fileName = recorder.requestFileLoadIn(sc);

            if(!fileName.equals(""))
            {
                  stationNum = recorder.getStationNum(fileName);
                  pattern = recorder.readFile(fileName);
            }else
            {
                  System.out.print("How many stations do you want
to type in: ");
                  stationNum = sc.nextInt();
                  System.out.println("Please input the metro graph
(eg. A: B(20), C(10)) : ");
                  sc.nextLine(); //skip current line;
                  //All nodes should be inputed
                  for(int i=0; i<stationNum; i++)
                  {
                        pattern += sc.nextLine()+"\n";
                  }
```

```
            }

            grapher.readGraph(pattern);

            System.out.println("\n"+"-".repeat(40));
            System.out.println("\nThe below is your metro graph:
");
            grapher.printGraph();
            System.out.println("");

            System.out.print("Which station do you want to start:
");
            startStation = sc.next();

            System.out.print("Which station is your destination:
");
            destinationStation = sc.next();

            System.out.print("Which mode do you want to choose?
(1)most saving money (2)most saving time(least stations): ");
            mode = sc.nextInt();
            System.out.println("\n"+"-".repeat(40)+"\n");

            calculator = Factory.createCaculator(startStation,
destinationStation, mode, grapher);
            calculator.calculate();
            System.out.println("All routes: ");
            calculator.printAllRoutes();
            calculator.printResult();
            System.out.println("-".repeat(40)+"\n");

            recorder.requestSaveGraph(sc, pattern);
```

```
            System.out.println("Thanks for using metro calculator
! ");
      }

}
```

Sample input/output:

input	台北車站: 西門(20), 忠孝新生(20), 中正紀念堂(30) 西門: 台北車站(20), 中正紀念堂(10) 忠孝新生: 台北車站(20), 東門(10) 東門: 忠孝新生(10), 中正紀念堂(10) 中正紀念堂: 西門(10), 台北車站(30), 東門(10)
output	---------- Welcome to Metro Calculator ! ! ! ---------- Do you want to load present file of metro graph? (y/N): N How many stations do you want to type in: 5 Please input the metro graph (eg. A: B(20), C(10)) : 台北車站: 西門(20), 忠孝新生(20), 中正紀念堂(30) 西門: 台北車站(20), 中正紀念堂(10) 忠孝新生: 台北車站(20), 東門(10) 東門: 忠孝新生(10), 中正紀念堂(10) 中正紀念堂: 西門(10), 台北車站(30), 東門(10) ---------- The below is your metro graph: 台北車站 --> 西門(20), 忠孝新生(20), 中正紀念堂(30)

西門 --> 台北車站(20), 中正紀念堂(10)
忠孝新生 --> 台北車站(20), 東門(10)
東門 --> 忠孝新生(10), 中正紀念堂(10)
中正紀念堂 --> 西門(10), 台北車站(30), 東門(10)

Which station do you want to start: 台北車站
Which station is your destination: 東門
Which mode do you want to choose? (1)most saving money (2)most saving time(least stations): 2

All routes:
- 台北車站 --($20)--> 西門 --($10)--> 中正紀念堂 --($10)--> 東門 | sum price: 40, all stops: 4
- 台北車站 --($20)--> 忠孝新生 --($10)--> 東門 | sum price: 30, all stops: 3
- 台北車站 --($30)--> 中正紀念堂 --($10)--> 東門 | sum price: 40, all stops: 3

Among these routes
The most saving time ways are:
- 台北車站 --($20)--> 忠孝新生 --($10)--> 東門 | sum price: 30, all stops: 3

  Cost $30, Stops 3

- 台北車站 --($30)--> 中正紀念堂 --($10)--> 東門 | sum price: 40, all stops: 3

  Cost $40, Stops 3

Do you want to save the graph? (y/N): N

	Thanks for using metro calculator !

TeamWork Table:

Simple Table:

	黃茂勛	吳冠勳	蘇建安	陳彥融
Coding	✔			
Report	✔	✔	✔	✔
Slide	✔	✔	✔	✔

Detail Table:

	Coding	**Report**	**Slide**
黃茂勛	1. program content design 2. class structure design 3. code implementation	1. class table 2. class UML diagram	1. page 2-9 introduction flow structure simple demo
吳冠勳		1. problem description 2.class Calculator 3.class Factory 4.beautify the report	1.beautify the slides 2.page 10-18

蘇建安		1.class FileRecorder 2.class SavingMoney 3.beautify the report	1. modify the slides 2.page 10-18
陳彥融		1.class Station 2.beautify the report	1.page 19-20 2.slightly beautify the slides