CSCI 470/502 **Assignment 4 – Mile Redeemer Console App** **Summer 2021**
150 points

Objective

This assignment's objective is to write a console-based Java application with which a travel agent could present options for travel destinations to a client who wants to redeem his or her accumulated frequent flyer miles. For the sake of simplicity, we will assume all trips depart from O'Hare International Airport and that the miles represented are required for a roundtrip ticket.

Depending on the distance, each destination requires a different number of frequent flyer miles to obtain a roundtrip ticket. Note that, if the client travels during the "off season", he or she may be able to take advantage of the "off season" mileage of a destination that requires fewer miles to obtain a roundtrip ticket to that particular destination.

Input File

A list of destination cities and related ticket redemption information will be read from a file (one named `destinations.txt` is provided for you but you may create your own). The file is formatted with one city, or destination, per line, with the first five items, or fields, separated by a semicolon (**;**) and the last two separated by a hyphen (**-**) as shown in the example below. The six items regarding a single destination represent:

*The destination city name; the normal mileage needed for an economy class ticket; the "off season" mileage needed for an economy class ticket; the additional mileage needed for upgrading from economy to first class; the beginning month of departure during which "off season" mileage **can** be used instead of the normal mileage - the ending month of departure during which "off season" mileage **can** be used instead of the normal mileage*

Example Input File:

```
Berlin;17250;12750;5250;3-5
Hong Kong;30000;20250;17500;2-4
Hyderabad;30000;20000;15000;2-4
Sidney;50000;30000;20000;5-6
Paris;20000;15000;10000;2-4
New York;10000;5000;8000;10-11
Tokyo;29000;16500;5350;2-5
Vienna;17500;12500;5000;2-4
Washington, D.C.;9000;7500;2000;10-12
```

The above are the contents of file `destinations.txt` as attached to the Assignment on Blackboard.

Redemption Algorithm

Your algorithm should 1) try to get tickets that travel the farthest, 2) use "off season" mileage whenever possible, 3) try to display as many different tickets as possible given the information, and, only ***after*** you have determined the destinations to which the customer can fly, 4) determine if you can use the remaining mileage for upgrade for any of the destinations to which the customer is eligible to fly (try to upgrade the longest trip first, then the next-to-longest - if there is one, etc.).

The Classes

`Destination`

Write a class that encapsulates information such as the name of the destination city; normal miles required for a ticket; "off season" miles required to for a ticket during the "off season" months; additional miles for upgrading from economy to first class; start month of the "off season" and end month of the "off season". Private instance variables and public accessor methods should be written for this information.

`MileRedeemer`

Write a class to encapsulate the logic for redeeming mileage. This class should have private instance variables for an array of `Destination` objects, and an integer to represent the remaining miles after the user's Frequent Flyer Miles have been redeemed.

Define the following methods in the `MileRedeemer` class:

- `public void readDestinations(Scanner fileScanner)`

  For this method, we use a `Scanner` object as the input parameter for flexibility and reusability. For example, we could reuse the method to read files from different sources. This method should use the `Scanner` object to read and parse the destination data into an array of `Destination` objects. Before ending, the method should sort the array of `Destination` objects in descending order by normal mileage (see more information about sorting below).

- `public String[] getCityNames()`

  This method should loop through the array of `Destination` objects and create an array of `String` objects from the city names. This array can be sorted in ascending order and returned (to be printed out by the main app) just for the display of all possible destinations.

- `public String[] redeemMiles(int miles, int month)`

  For this method, miles is the total available miles, and month is the desired month of departure. To avoid writing one huge method, you can (and probably should) have the `redeemMiles()` method call some other methods to accomplish subtasks as part of the larger overall algorithm. This method should return an array of `String` objects containing descriptions of redeemed tickets to be printed out by the main app. It should also save the miles remaining after the tickets have been redeemed.

- `public int getRemainingMiles( )`

  This method should return the saved remaining miles.

`MileRedemptionApp`

This is the main app class that will have the `main( )` method and will drive the entire process.

First declare a `Scanner` object for the keyboard and prompt the travel agent for the name of the `.txt` file of destinations. Next, it should declare another `Scanner` object to read the destination records from the

file. This `Scanner` will be the `Scanner` parameter passed to the `readDestinations()` method of the `MileRedeemer` class where the array of destinations will be created.

Once back from `readDestinations()`, use `getCityNames()` to get and print the list of all possible destinations. Prompt the travel agent for the client's Frequent Flyer Miles balance and for the client's month of departure. Using these two integers, call `redeemMiles()` to determine the destinations available to which the client can fly and present them to the travel agent.

Note that, if `redeemMiles()` returns an empty array, print an appropriate message such as:

*** Your client has not accumulated enough Frequent Flyer Miles ***

Either way, then use `getRemainingMiles()` to present the Frequent Flyer Miles remaining after determining the destinations to which the client can fly.

After presenting the possible destinations and remaining miles, prompt the travel agent asking if he or she would like to start over. Do not ask for a new file name, do not print the header statement and do not print the list of all possible destinations again. Let the travel agent answer with 'y' or 'Y' or 'yes' or 'Yes' and, if he or she answers in the affirmative, the next thing he or she should see is a prompt to enter the client's Frequent Flyer Miles and departure month and the process of determining a new list of possible destinations.

`MileageComparator`

Because the cities in the input text file will probably be out of order based on normal miles, the `readDestinations()` method will need to sort them in descending normal miles order before any redemptions can be done. The class `java.util.Arrays` has a static `sort()` method that takes an array and an implementation of `Comparator`. The latter is an object of a class that defines the ordering:

```
import java.util.Comparator;

public class MileageComparator implements Comparator<Destination>
{
   @Override
   public int compare(Destination d1, Destination d2)
   {
      return (d2.getNormalMiles() - d1.getNormalMiles());
   }
}
```

Note that this class is optional as you may be able to find a different way to implement the Java interface `Comparator`. For example, create an inner class.

Note: The array of city name strings created and returned by `getCityNames()` can also be sorted using a version of `Arrays.sort()`. There's no need to write a `Comparator` in this case, though, as the “natural ordering” of the strings will be sufficient.

(continued)

Programming Notes

Exception Handling and Input Validation

For simplicity's sake, please do not add try catch blocks for exception handling and do not do any sort of input validation.

File I/O and `String` Parsing

The `Scanner` class can be used to read the lines from the file. Before reading the lines, a `java.util.ArrayList` should be created:

```
ArrayList<Destination> destinationList = new ArrayList<Destination>();
```

As lines are read from the input file, the lines need to be parsed, or broken into fields, using ";" as the delimiter. Instances of `Destination` are created and added to the array list. There are a variety of different ways to parse a `String` into fields. One easy way is to use the `split()` method of the `String` class. Note that the last two fields, i.e., begin month and end month for the "Frequent Flyer" months, are separated by a hyphen.

To change a `String` to an integer to assign to the three miles instance variables and the beginning and ending month instance variables, use wrapper class `Integer`'s method `parseInt()`.

After the whole file is read, the `ArrayList` can be converted to a normal, fixed-length array of objects:

```
Destination[] destinationArray = (Destination[]) destinationList.toArray(new
                                        Destination[destinationList.size()]);
```

By doing this, we can take advantage of Java's built-in methods for sorting an array.

Reading the File

So that you will not have to do any exception handling, add the following at the end of the method declaration that reads the input file (no semicolon at the end of it):

```
throws IOException
```

IMPORTANT

Please read the **Java Coding and Documentation Guidelines** document in Course Documents on Blackboard **before** submitting your final app for grading.

Submitting the Assignment

The assignment will be submitted on Blackboard. Please submit the three (or four) .java files on Blackboard. **DO NOT ZIP THEM AND DO NOT SUBMIT THE INPUT FILE!**

(Exact output follows)

Exact App Output Using the File `destinations.txt` Provided

```
Please enter the name of the file:  destinations.txt

---------------------------------------------------------------
       WELCOME TO THE JAVA AIRLINES MILES REDEMPTION APP
---------------------------------------------------------------

List of destination cities your client can travel to:

Berlin
Hong Kong
Hyderabad
New York
Paris
Sidney
Tokyo
Vienna
Washington, D.C.

---------------------------------------------------------------

Please enter your client's accumulated Frequent Flyer Miles:  49600

Please enter your client's month of departure (1-12):  4

Your client's Frequent Flyer Miles can be used to redeem the following tickets:

* A trip to Hong Kong in Economy Class
* A trip to Hyderabad in Economy Class
* A trip to Washington, D.C. in Economy Class

Your client's remaining Frequent Flyer Miles: 350

---------------------------------------------------------------

Do you want to continue (y/n)? y

---------------------------------------------------------------

Please enter your client's accumulated Frequent Flyer Miles:  3421

Please enter your client's month of departure (1-12):  3

*** Your client has not accumulated enough Frequent Flyer Miles ***

Your client's remaining Frequent Flyer Miles: 3421

---------------------------------------------------------------

Do you want to continue (y/n)? y

---------------------------------------------------------------

Please enter your client's accumulated Frequent Flyer Miles:  154351

Please enter your client's month of departure (1-12):  7
```

```
Your client's Frequent Flyer Miles can be used to redeem the following tickets:

* A trip to Sydney in Economy Class
* A trip to Hong Kong in Economy Class
* A trip to Hyderabad in Economy Class
* A trip to Tokyo in First Class
* A trip to New York in Economy Class

Your client's remaining Frequent Flyer Miles: 1

-----------------------------------------------------------------

Do you want to continue (y/n)? n

-----------------------------------------------------------------------
       THANK YOU FOR USING THE JAVA AIRLINES MILES REDEMPTION APP
-----------------------------------------------------------------------
```