CPT 187: PROGRAMMING - PROGRAM 6

INTRODUCTION

You have successfully handled a class that simulates a complete operating program for a metropolitan subway train. Now, we must add the functionality of an express train.

What we want to do is update the class to handle a pre-defined situation, where in passengers are shuttled between only two stations, when the user chooses to create and run an express train. For example, sometimes a single train is booked to run a dedicated, restricted route; this known as a shuttle or express. In effect, the train acts as a shuttle between the two stations.

In this case we want the train to be able to move passengers between only two stations: the Home Station and one other station. The station number of the Home Station and the other station will be designated by the user (by numeric value). For example, say we want to transfer all of the passengers waiting at the Home Station to Station 'n' and all of the passengers waiting at Station 'n' to Home Station. Depending on the capacity of the train, and the number of people waiting at each station, this may take several trips.

To create such a solution, we will need included a menu, an overloaded method, functions that work with Strings, and another class, the Station class.

MAIN MENU

For the menu, you must provide users a Main Menu with options to build and simulate an express train or a metropolitan train. User must also be given the option to Exit (using X) or Quit (using Q). After a user has completed a simulation, the user must be returned to the Main Menu. NOTE: This program must run until user choose to quit.

To process a user's request, that is NOT Quit or Exit, you may use an if/else structure or a switch (case) structure; the choice is yours for this program.

STRINGS AND CHARACTERS

So far, we have maintained the use of ints for our station numbers. This approach allows us to easily modify and process train 'movement' in our simulations. However, as true use of data types, a station number would not be a numeric value; rather, it would be a String. As such, for simulations involving an express train, we must take the station number and convert it to a String. In other words, when a user chooses to build and simulation an express train, we must convert station number to String before we process the Station numbers. The process for simulation a metropolis train should be changed.

OVERLOADING METHODS

Now that we are using a String to process the movement of an express train, we must make review the logic in the *moveToStation* method. When properly passed an int, it modifies destination station from current station. However, it cannot properly use a String to process the movement. As such, we will need to create an overloaded version of the *moveToStation* method to take a String. Then, once inside the method, we need only to write one statement to convert the String to an int and we are back in business. This means, for a metropolitan train, we send the station number as an int and leave the exiting *moveToStation* method as is to move the train. And for an express train, we send the station number as an String to the overloaded method and convert the String to an int to move the train.

THE STATION CLASS

For this program, we will have a third class (in addition to our SubwayTrain and MainClass).

This class will represent the train station. The class will need two attributes (private data variables): passengersWaiting and maximumCapacity.

A constructor is needed for the Station class, but only use the variable(s) needed in the construction of any one Station object. You will also need a 'getter' and 'setter' method for the passengersWaiting variable and the maximumCapacity variable.

In keeping with the Java conventions and Coding Standards, these methods should be named *getPassengersWaiting* and *setPassengersWaiting* and *getMaximumCapacity* and *setMaximumCapacity*.

You will need another method, *reduceNumberWaiting*, that accepts an integer as its only parameter. That parameter will be the number of people who are getting on the subway; the method should reduce the number waiting by that amount.

Lastly, you will need to move the random number generator from SubwayTrain to the Station class. This will be more representative of the origin of passengers; as passengers to come to a station to buy a ticket before boarding a train. And though the method was set to twice the maximum capacity of a train, it must now be set to the maximum capacity (of the train) times maximum capacity (of the train),

This class will keep the number of passengers waiting at any time at any stations. To properly process the program operations, we will create two instances of the Station object; one for the Home Station and one for the destination station (aka Station 'n').

THE SUBWAYTRAIN CLASS

The 'getter' methods can still be used as they are. Also, the existing move method, the one that takes one argument, checks it for validity, and moves the train to that station, is just fine for the program requirements.

However, the class will need two new methods to operate an express train:

- *loadFromStation*: this method accepts a Station object as its parameter. The method should then load as many passengers as it can – do not overfill the Subway! It will need to get the number of passengers waiting at the station (using the Station object it was passed), determine how many passengers can be loaded, load the passengers and reduce the number of passengers waiting at the station by the number loaded. Make certain you don't forget that last part or your program will run forever.
- *unloadAll*: This method simply unloads all the passengers on board. NOTE: This method can be used for the express and the metropolis train, if desired.

NOTE: The existing loader and the unloader will still work for this program, since they load and unload based on random numbers for the metropolitan train. Or, if you wish, you can modify the metropolis train logic in one Station object as the 'current' station and the other Station object as the 'destination' station.

But for the express train, you must use the *loadFromStation* method to load the train with as many passengers as the train will hold, and an unloader that unloads all passengers at the destination. The idea of a shuttle train is to move as many people as possible from one station to the other until everyone has reached their destination.

What we must consider now are the changes to be made in the main method.

UPDATE THE APPLICATION (THE MAINCLASS)

There is a significant difference in how this program used to run (Program 5) and how this program must run now (Program 6).

The functionality should reflect the following:

- When the program starts, there must be a Welcome Banner and a request for the user's name
- Then the main menu must be presented and include options for:
    - Metropolis train operations
    - Express train operations
    - Quit/Exit
- Now, all users (who have not chosen to Quit/Exit) must provide the following information:
    - The Home Station number and the Maximum Capacity of the train
- Then based on the user's selection there needs to a request for:
    - The number of stations the metropolis train will serve
    - Or
    - The number of Station 'n' (which is one other station) the express train will serve,
- For the number of passengers:
    - The metropolis train will continue to request a random number to unload and load
    - Or
    - The express train will request a random number of passengers waiting to load only twice (once for the Home Station and once for Station 'n').
- All user communication must continue to include the following:
    - At arrival, the station number, the number of passengers unloaded, and the number of passengers loading
    - At departure, the station number, the number of passengers on board, and the destination station number
- To end the simulation:
    - The metropolis train will continue to run until it returns to the Home Station (just as before)
    - Or
    - The express train will continue to run until all passengers at the Home Station have been transported to Station 'n' and all passengers at Station 'n' have been transported to the Home Station, and the train returns to the Home Station.
- Once the simulation has ended, the user must receive an 'End of Simulation' report to include:
    - The state of the object: current Station, number of passengers on board, and maximum capacity of the train
    - Optional information: Number of stations served, and number of passengers transported
- Lastly, the user must be directed to the Main menu.
- This program must continue to run until the user chooses to Quit or Exit.
- Once the user has chosen to Quit or Exit, they must receive a 'Usage Report' to include:
    - Number of metropolis trains simulated, and number of express trains simulated
    - Optional information: Total number of stations served, and total number of passengers transported

Again, you will need to instantiate three objects, two Station objects (one for the current station and one for the destination station) and only ONE SubwayTrain object (which will be built and rebuilt as a metropolis train or express train, as needed).

For an express train, the passengersWaiting attribute for each Station object should be set using the random number generator. Then the train will simply run between the Home Station and Station ‘n’, carrying as many passengers as it can hold, until the number of passengers waiting are down to 0 at both stations and the train returns to the Home Station.

Where for a metropolis train, the random number generator in the Station class will be used to generator the number of passengers to load and unload at any station. Though, as before when you instantiate the train object, set it up to hold a max capacity of passengers, start it at the Home Station. And then, the train will simply run through the number of stations to serve, carrying as many passengers as it can hold, until the train returns to the Home Station serving each station along the way.

So, as you test the program, be sure to use both train options, with multiple sets of data to be sure your simulations produce the desired results.

As before, you need to announce what is happening as the train goes about arriving and departing and loading and unloading passengers. Then at the end of each simulation, the ‘End of Simulation’ report must be output. And once a user chooses to Quit/Exit, the ‘Usage Report’ must be output. Be sure to watch the formatting of your output throughout your program! Remember, we must work to write fully functional, profession, and business like programs. Also, including efficiency in your program code would be a bonus for the user and your programming skills!