CSCI 2540 Assignment 8

100 points

Due date: Tuesday, Nov. 15 by 11:59pm

In this assignment you will implement bank simulation problem we discussed in class. The problem is also discussed in detail on the textbook. **You need to make sure that you understand the logic of the problem and how to solve the problem on paper before you actually implement it** (especially if you missed the class when we discussed the problem). A PDF file that includes the pages on the textbook describing the problem is also attached for this assignment. Please read the pages if you do not understand how to solve the problem.

The two data structures are needed for this problem. One data structure is the queue that simulates the waiting line in the bank. The queue also includes the customer currently being served which is at the front of the queue. The element type of the queue is an Event object. **Event** is a class that you will need to write.

The other data structure you need to implement is the EventList which is a list of events. This list includes the next arrival and the next departure event. The list is sorted based on the time.

The input is a text file (named "**assg8_input.txt**") of arrival and transaction times. Each line of the file contains the arrival time and transaction time for a customer. The arrival times are ordered by time. A sample input file is attached for this assignment.

For this assignment, you will need to write three classes, the **Event** class, the **EventList** class, and the **Simulation** class (with the main method).

The Event class is for a single event. It should include constructors, get methods, compareTo method, and toString method, as well as methods to check if an event is an arrival event or a departure event.

The EventList class should include methods that allow to insert, remove, or retrieve an event from the EventList. It should also include a default constructor and a method to check if the event list is empty. Make sure that the EventList is sorted based on time when you add a method to the list. If an arrival event and a departure event have the same time, the arrival event precedes the departure event.

The Simulation class should include a method to process an arrival event and a method to process a departure event, in addition to the main method. The process is started by reading the first arrival event and adding it to the EventList. After that you can repeat the process by processing each event in the EventList and updating the queue and EventList accordingly as well as generating the output.

There are two types of events, arrival event and departure event. For example, an arrival event ('A', 1, 5) indicates that this event has arrival time of 1 and transaction time of 5 (minutes). A

departure event ('D', 6) indicates that this departure event has departure time of 6. The Event class will need to include variables that are needed for arrival and departure events.

Your program needs to generate the sequence of the events during the process (see the sample output on the next page). Your program also needs to calculate the average waiting time. To do that, you can calculate the waiting time for each customer during the process and then calculate the average waiting time at the end. A customer's waiting time is the time between the customer's arrival and the time the customer starts the transaction. Your program also needs to generate the sequence of the events during the process (see the sample output on the next page).

Below is a sample input (on the left column) and sample output (on the right column).

Input File **Output**

```
1   5   Simulation Begins
2   5   Processing an arrival event at time:    1
4   5   Processing an arrival event at time:    2
20  5   Processing an arrival event at time:    4
22  5   Processing a departure event at time:   6
24  5   Processing a departure event at time:   11
26  5   Processing a departure event at time:   16
28  5   Processing an arrival event at time:    20
30  5   Processing an arrival event at time:    22
88  3   Processing an arrival event at time:    24
        Processing a departure event at time:   25
        Processing an arrival event at time:    26
        Processing an arrival event at time:    28
        Processing an arrival event at time:    30
        Processing a departure event at time:   30
        Processing a departure event at time:   35
        Processing a departure event at time:   40
        Processing a departure event at time:   45
        Processing a departure event at time:   50
        Processing an arrival event at time:    88
        Processing a departure event at time:   91
        Simulation Ends

        Final Statistics:
        Total number of people processed:   10
        Average of time spent waiting:      5.6
```

Javadoc comments are required for this assignment and all future assignments (for all public methods or constants, as well as comments at the beginning of each file).

Submission instructions:

To submit your programs, you need to submit your programs electronically on Canvas.

Please **use a named package for each of your assignment.** For example, for assignment 8, create a new package and **name your package as assg8_yourPirateId** (use lower case for your pirate id), such as `assg8_smithj19`. You also need to include a statement such as "`package assg8_smithj19;`" at the beginning of each of your .java file. **Please follow this naming convention exactly for all future assignments. You will be deducted points for not doing so.**

When you submit your files to Canvas, please submit a zip file with your package folder inside the zip file. The package folder should include only .java files (make sure you include .java files, not .class files). The name of the folder should match with your package name. You do not need to submit the input text file.