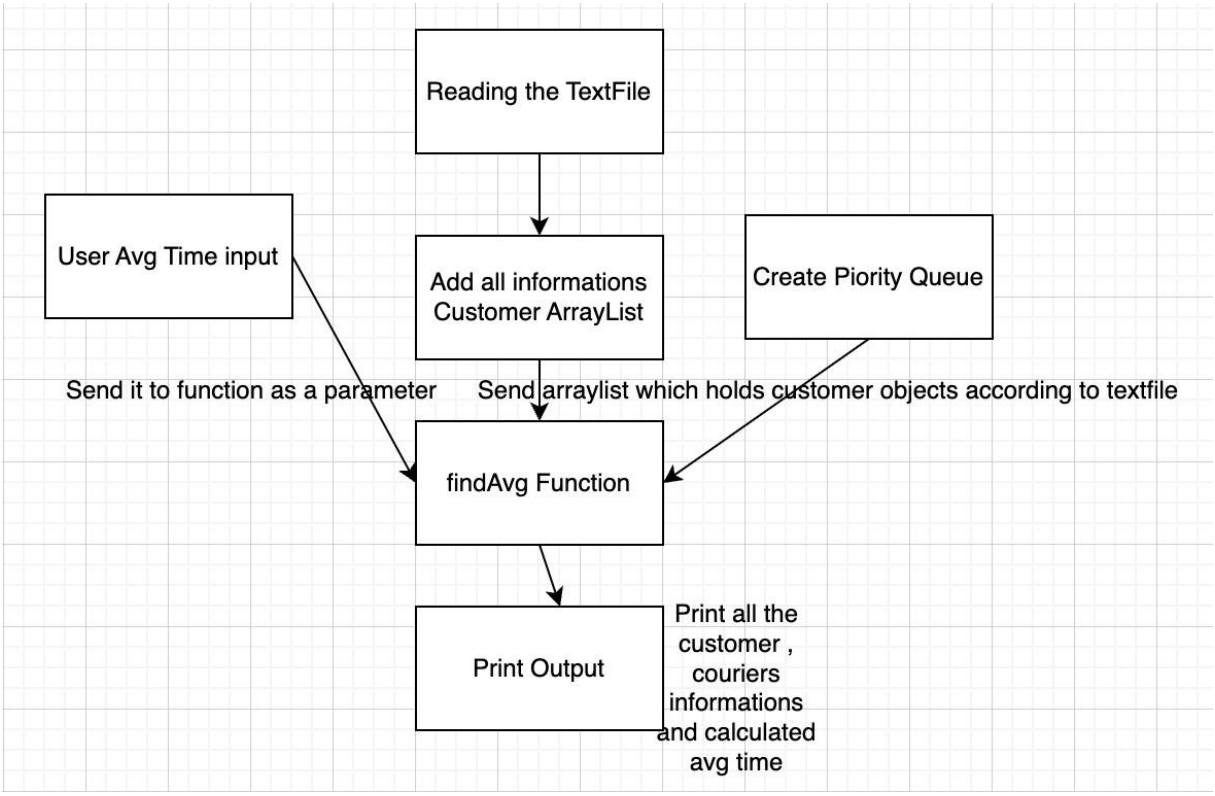
PROGRAMMING ASSIGNMENT REPORT - 4

Problem Statement and Code Design

In this assignment, we created a heap-based priority queue for assigning order to the couriers of online order delivery company. According to given data of delivery times, we aimed to write an algorithm which optimize the suitable waiting time for customers. Heap based Priority queue logic is useful in advance for storing the data by considering levels. Since we need to follow the company standards, we can assign higher priority to customers who have been waiting for a longer period. Eventually, after the system loop completed, the program prints the minimum number of couriers required, the simulation details and the final average waiting time.

Furthermore, the code contains sub-components to provide adaptability. These subcomponents have shown by utilizing a structure chart below.



Implementation & Functionality

To provide a clear and efficient code design we create 4 classes which are main class, customer class, courier class and pq class. Since we don't allow to use any external library, we write our own implementation of priority queue.

In Priority Queue Class we used a heap data structure to store customer objects.

Courier Class has getters and setters for id and availabilityTime properties, as constructor initializes the ID and sets the availability time to 0.

Customer class has getters and setters for some properties which are: id, priority, arrivalTime, serviceTime and waitingTime. These values initialize based on the input data. Moreover, findAvg method has used **delMax:** picks any maximum key

findAvg: These methods take three parameters: an arraylist of type "customer" that holds them, an empty priority queue object, and the average time taken from the user. There is a while true loop in the method that allows the process to be iterated until break. First of all, the system resets the courier order because it tries to find the average for the courier numbers between 0 and 10, and as a result, it gets it from the most efficient avgTime.

It creates courier objects and, by looking at their availability, calculates according to the arrival time and service time of the customers and matches the couriers with the customers in a way that minimizes the wait. The availability value returns true after a courier finishes the delivery.

Finally, to calculate avgWaitTime while making these mappings, we add totalWaitTime and overall waiting times and divide by the number of customers. Finally, we print the most optimal time and information so that this time is less than the "avg" we take as a parameter.

Main class designed for generating delivery process by reading the input data and printing output according to the calculations.

Testing

Firstly, when we brainstormed for the implementation of the algorithm, this was possibly the most difficult assignment. After conducting a thorough discussion on how to implement priority queue and heap structures and how they work, we developed our Heap-Based Priority Queue class. Initially, we wanted to save data using an array format. There was a final testing process in the VPL once all the classes and methods were created. The main class reads text files, creates customer objects, and adds them to an array list. Then use the findAvg() function to calculate the average. Our code was working clearly in Eclipse IDE. But when we uploaded the VPL, we got an error. Additionally, we again tried and had no issues and correctly obtained the result when we tried it using our data separate from the data in the VPL. Unfortunately, we couldn't solve the VPL problem.

FINAL ASSESSTMENTS

1. While working on this assignment, it was challenging to find the correct algorithm for assigning the highest priority customer to the courier.
2. However, we have gained a deeper level understanding of priority queue data structure and several ways of implementing this structure like heap based, arrays BST or linked list.
3. After this assignment, it was clearer to decide when we can use priority queue data structure.
4. Thanks to this assignment, we can chance to solve a real-world problem and implement our own algorithm.
5. At the end, we learn how to sort the heap structure according to a certain priority