**Assignment – 1**

**Name: Tejal Vijay Nijai**

**Student No.:19210412**

**Email Address: tejal.nijai2@mail.dcu.ie**

**Module:CA670**

**Programme: MCM1**

An assignment submitted to Dublin City University, School of Computing for module CA670 Concurrent Programming, 2019/2020. I understand that the University regards breaches of academic integrity and plagiarism as grave and serious. I have read and understood the DCU Academic Integrity and Plagiarism Policy. I accept the penalties that may be imposed should I engage in practice or practices that breach this policy. I have identified and included the source of all facts, ideas, opinions, viewpoints of others in the assignment references. Direct quotations, paraphrasing, discussion of ideas from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the sources cited are identified in the assignment references.

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work. By signing this form or by submitting this material online I confirm that this assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study. By signing this form or by submitting material for assessment online I confirm that I have read and understood the DCU Academic Integrity and Plagiarism Policy (available at: https://www.dcu.ie/policies/policies.shtml)

Name: Tejal Vijay Nijai

Date:17/03/2020

**1. Design of the Program:**

My solution contains four classes:

1.BarberShop.java

2.WaitingArea.java

3.Barber.java

4.Customer.java

**BarberShop.java** is the main class containing main method. The shop (barbers and waiting area chairs) and the number of customers coming to the shop are designed based on user input. The number of chairs in waiting area, barbers in the shop and customers coming to the shop are taken by user at runtime. To calculate the customer arrival rate to the shop and the haircutting time taken by barber with their respective mean and standard deviation is also taken by user at runtime. Based on the user inputs, the executorservices get started according to the to the provided number for barbers and customer respectively. An interval (in milliseconds) is generated based on a formula **(random. nextGaussian () \* inputted\_standard\_deviation) + inputted mean)** using provided mean and standard deviation. This time interval is used to start the Customer executorservice one after another to indicate customers are arriving at certain time interval.

**WaitingArea.java** class represents a queue which is accessed by barber and customer threads and contains the customers up to the number given by the number of waiting chairs in the area submitted by user at runtime. This class has a **cutHair ()** method that is called by barber thread to check if any customer is in the waiting area and if anyone is available, then pull that customer and do the haircut for the time determined by the **getHaircutTime ()** feature. This time taken will be determined by user's given respective mean and standard deviation for haircut time generation. **waitForHaircut ()** method adds the customers to the queue coming to the shop. When the queue has customers seated according to the maximum number of chairs permitted, any new customer who comes to the shop must leave because there is no seat available. In **Barber.java class**, **cutHair ()** is being called in thread’s run () method from waitingArea class and similarly, **Customer.java** class calls to **waitForHaircut ()** method from waitingArea.

**2.Correctness and fairness properties of the program:**

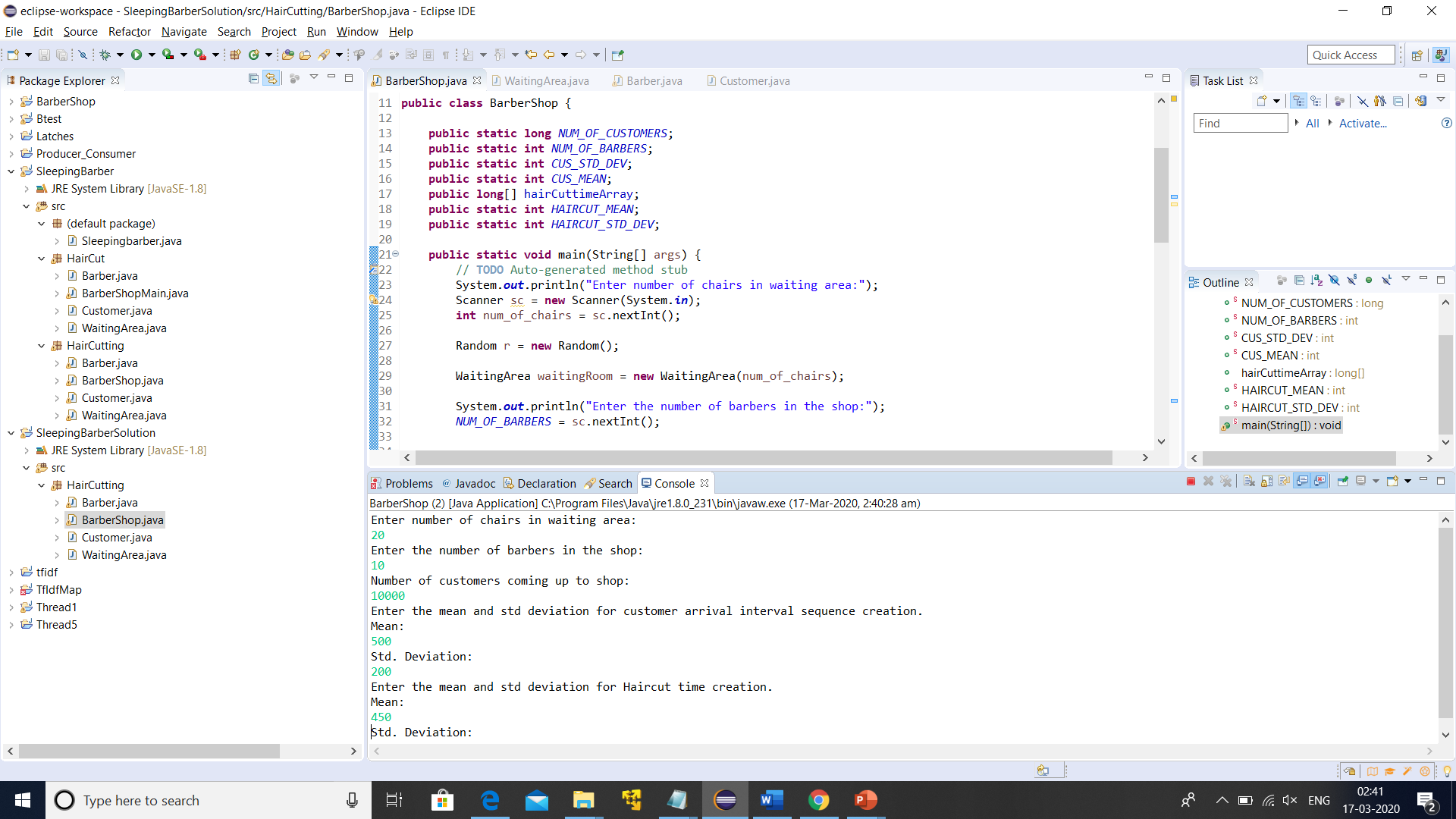
* All the customers get processed in sequence. As soon as the customer comes to the shop, it will check if seat is available or not if it is available it will take a seat and notify the barber thread that someone has been added in the queue.
* If any barber thread is available, it will be processed by the barber thread or the customer will keep awaiting.
* In case of seat unavailability, customer will leave the shop. As per the user’s input of mean and standard deviation values, either all the customers get served(because both the customer arrival time rate and haircut time taken are small and everyone gets served) or some of them leave as the seat is not available because the time of haircut is more than the customer coming to the shop.
* If all the customers expected to come to the shop are gone, barber will go to sleep and continue to wait before somebody shows up. This queue synchronization guarantees consistency for my system.

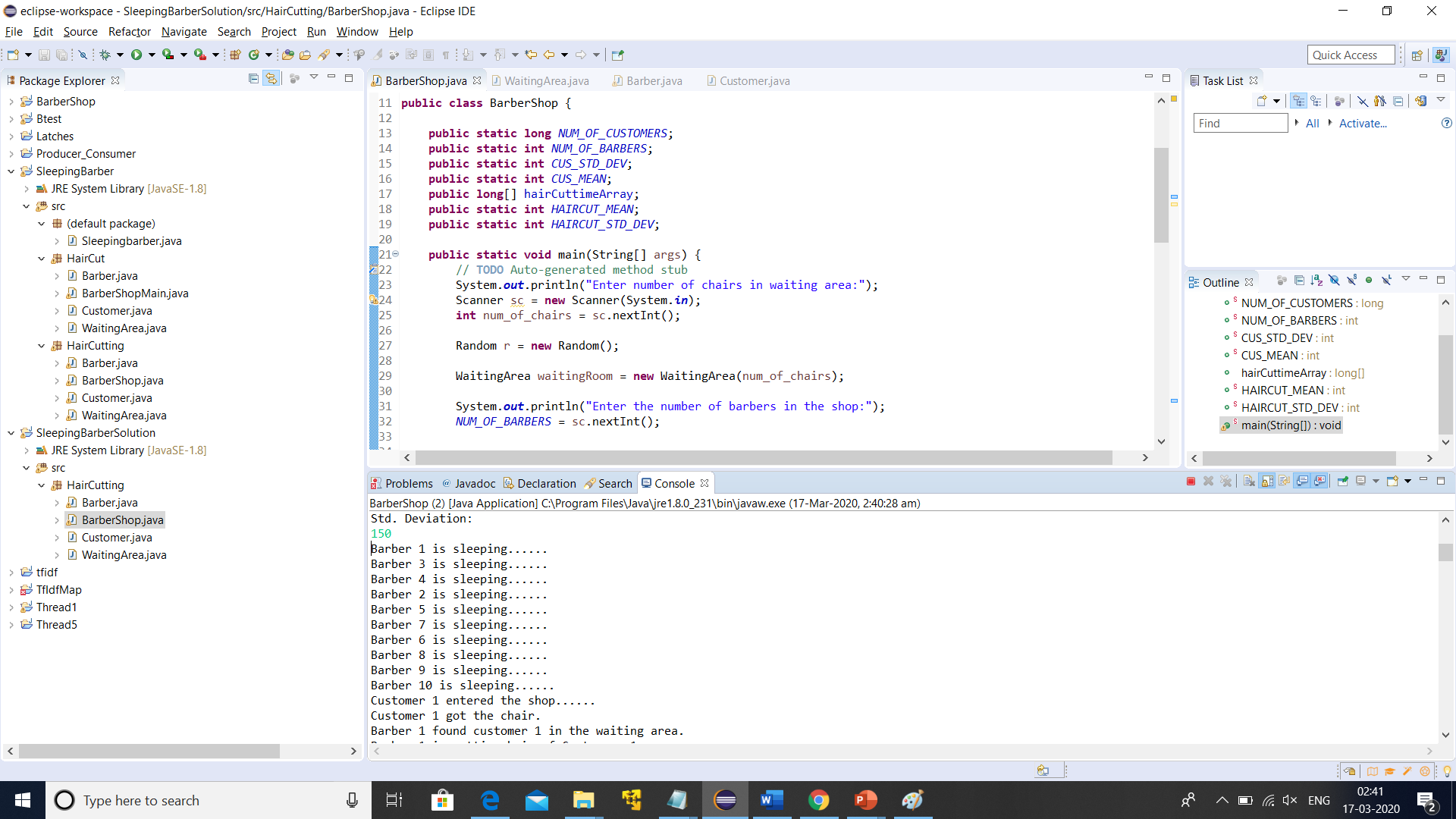
I referred some programs [1][2] for the design of my program.

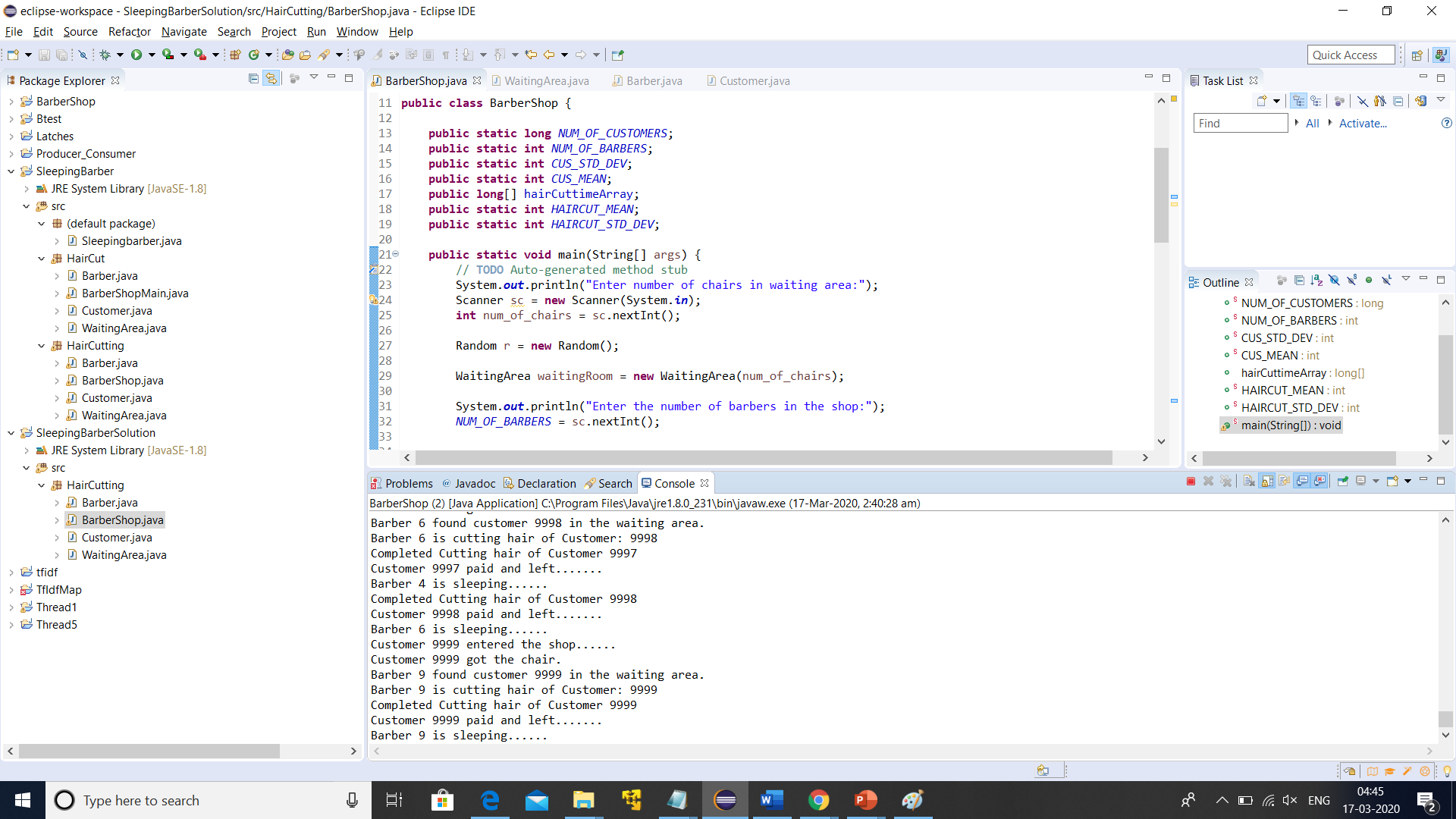
**3.Solution of sleeping barber(s) problem used in practice:**

Sleeping barber problem solution can be used in online movie ticket booking. In online movie ticket booking, seats in the movie hall can be treated as waiting area chairs. If seats are available, as customers come, they will be allocated the seats or else they will leave the theatre. Mutual exclusion is possible in ticket booking as when a person is booking a seat, it’s gets locked by that person and cannot be accessed by any other person. There is fairness and absence of starvation as people come in, they get chance to book ticket as per their requirement and their arrival sequence. There is no deadlock as one person’s ticket booking is not blocking the process of booking ticket of another person’s booking.

**4.Screenshots of my Program Output:**







**5.References:**

1. <https://github.com/ylegat/sleeping-barber/tree/master/src/exercise>
2. <https://orajavasolutions.wordpress.com/tag/sleeping-barber-problem/>