**Declaration on Plagiarism**

This form must be filled in and completed by the student(s) submitting an assignment

|  |  |
| --- | --- |
| Name: | Rebekah Jennifer Manimaran |
| Student Number: | 19210518 |
| Programme: | Msc Computing – Data Analytics |
| Module Code: | CA670 |
| Assignment title: | Concurrent Programming |
| Submission Date: | 16/03/2020 |
| Module Coordinator | Dr. David Sinclair |

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. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. 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.

I have read and understood the referencing guidelines found at <http://www.dcu.ie/info/regulations/plagiarism.shtml>, https://www4.dcu.ie/students/az/plagiarism and /or recommended in the assignment guidelines

Name: Rebekah Jennifer Date: 16-03-2020

**JAVA THREADS – SLEEPING BARBER PROBLEM**

**Architecture and Design of the Program**

The Sleeping barber problem is designed with multi-threaded and multi-core solution in Java. In this program, the requirements mentioned in the assignment are all satisfied.

The project is constructed in such a way that, in a small barber shop with a set of M barbers and C Customers with N Waiting chairs that are given by users in runtime. Later, each barber has a chair in which the customer sits when they are getting their haircut. Then, When there are no customers in the shop the barber sleeps. The customer goes asleep when the barber is cutting their hair. If the waiting chair is full, the incoming customers exits the shop. Also if the customers are waiting the barbers calls them and waits for them to come and sit in the chair.

The java program consist of classes, each of these classes are explained below with their functionality.

**Barber.java:**

In this java class, customer arrival interval time and haircut service time has been calculated with mean and standard deviation in cHair () method. This has been performed using random number generator nextGaussian () method to convert the numbers into normally distributed one.

customer\_ArrivalTime = Math.round(Math.abs(Mean + fRandom.nextGaussian() \* StdDev) \* 100.0) / 100.0;

haircut\_ServiceTime = Math.round(Math.abs(Mean + fRandom.nextGaussian() \* StdDev) \* 100.0) / 100.0;

A barber shop closure time is specified to close the shop and leave by the end of the day. Firstly, In the run method the queue checks if it is empty or not, (i.e) it verifies if there is any customer waiting in the waiting room(queue) – if they are waiting, the customer calls them for a haircut else the barber goes to sleep in his chair.

**Customer.java:**

Here the customer awakens barber to take a chair when it is free for a haircut mentioned as getHcut () method. Also when the seats in waiting room are full the customer goes out of the barber shop else customer gets added to the queue mentioned in the run () method of customer java class.

**BarberShop.java:**

This class introduced the scanner function to take the input given by users in runtime. It takes number of barbers in the shop, number of customer coming to the shop and number of waiting chairs available. It initialises the array blocking queue in first in first out manner. It also creates new barber and new customer every time with the queue and starts the simulation process.

**Justifications:**

* **Absence of Starvation and Fairness**

In my program, I have used Array Blocking Queue which have managed the threads to be fair without any starvation. This method queues the customer who are entering from the entrance and allocates the haircut chair one by one till the barber becomes free. The barber thread that becomes free first takes the next customer thread from the queue. This proves the absence of starvation and fairness.

* **Absence of deadlocks**

I have made my program that will never have a situation where two threads block each other. The queue is performed in that way where barber and customer don’t block each other and terminates the program.

* **Correctness**

Hence, the program is correct if the above points are satisfied which is free of starvation and deadlocks.

**Solution of Sleeping Barber Problem in Practice:**

Here are the few examples where Sleeping Barber Problem are used in real time practice

1. **Train and Air Ticket Booking System**

In a booking system for a train or flight in different countries, there would be a waiting list of passengers who acts like a queue and they are neither confirmed customers/passengers nor cancelled ones. There shouldn’t be any kind of starvation or deadlock present there. This acts in First in First out manner where customers comes first gets the ticket confirmed and who comes late are put on waiting list like a queue, once the capacity of the queue is exceeded they are cancelled from the ticket purchasing system.

1. **Retail Market**

In many retail markets like Aldi, Tesco, Lidl sleeping barber problem methods are used to solve their complex problems based on their requirements. When there is more customers the counter gets opened one by one whereas when there are less customers the counter queue gets closed. **For instance:** **Counter 4 is open now for customers**. This acts as a queue without any starvation and deadlock.

1. **Sensor Lights in organisations**

In many offices and organisation, there are sensor lights which gets turned on only if the customer enters which acts like a queue. When there are no customers or people its gets turned off automatically based on sensor. I feel this also comes under sleeping barber problem methods. Where the people entering shouldn’t suffer without lights (i.e.) starvation.

1. **Warehouse Stock Management**

In some warehouse stock management, the stock count acts like a queue where once the stock goes down, “please alert me”. Then we try to fill stocks in the queue once it’s empty or less. This gets related with the sleep time of the haircut service.

**Test Case 01:**

**Aim:** To show that the program is free of deadlock or thread starvation

**Scenario:**

Number of Barber: 3

Number of Customers: 5

Waiting chair: 4

**Expected Output:**

The thread does not block each other and runs one after the other in the queue without any starvation.

**Actual Output:**

Successfully ran free of starvation and deadlock.

**Screenshot:**



