CA670 Concurrent Programming

Name	Rashmiranjan Das
Student Number	19210554
Programme	MCM(Data Analytics)
Module Code	CA670
Assignment Title	Java Threads
Submission Date	18th March 2020
Module coordinator	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 set out in the module documentation. 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 recommended in the assignment guidelines.

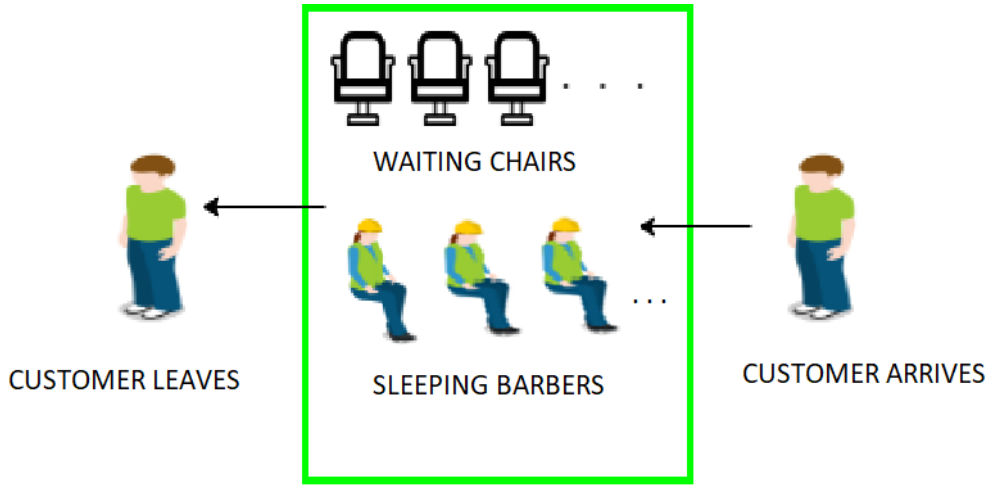
Name: Rashmiranjan Das Date: 18th March 2020

JAVA THREADS
The Sleeping Barbers Problem

I. Problem Statement

A small barber shop has two doors, an entrance and an exit. Inside is a set of M barbers who spends all their lives serving customers, one at a time. Each barber has chair in which the customer sits when they are getting their hair cut. When there are no customers in the shop waiting for their hair to be cut, a barber sleeps in his chair. Customer arrive at random intervals, with mean mc and standard deviation sdc. If a customer arrives and finds that a barber asleep, he awakens the barber, sits in the barber's chair and sleeps while his hair is being cut. The time taken to cut a customer's hair has a mean mh and standard deviation sdh. If a customer arrives and all the barbers are busy cutting hair, the customer goes asleep in one of the N waiting chairs. When the barber finishes cutting a customer's hair, he awakens the customer and holds the exit door open for him. If there are any waiting customers, he awakens one and waits for the customer to sit in the barber's chair, otherwise he goes to sleep.

II. Architecture of the problem



1. **TestMain class:** The driver.cpp is a driver program that tests your sleepingbarbers problem.

Input Parameters

nBarbers	The number of barber working in your barber shop
nChairs	The number of chairs available for customers to wait on
nCustomers	The number of customer who need a haircut service
serviceTime	Each barbers service time (hair cut time)
arrivalInterval	It is the mean interval between arrival of customers

stdDivBetweenCustomers	It is the standard deviation between customer arrival
stdDivServiceTime	It is the standard deviation in the service time of the barber
custLeave	Time taken by the Customer to leave

The test main class defines and initializes all the variables. It defines the blocking queue. It then calls the barbershop class to carry the following process.

2. **CUSTOMER CLASS:** The customer class is responsible to manage the functioning of the customers like
    a. Entering the shop
    b. To wait in the waiting chair (blocking queue)
    c. To leave the shop if waiting chairs are filled
    d. To take barbers chair when available for a haircut

3. **BARBER CLASS:** The Barber class handles all the Barbers activity
    a. Asking customer from waiting list to take the barber's chair
    b. Cutting Customers hair
    c. Waiting for customer to leave once he has got the haircut

4. **BARBERSHOP CLASS:** The barbershop class handles the all the entities in the barbershop (i.e. Barbers and Customers). It calls the Barber class and Customer class to handle their functioning respectively.

III. Justifications

Liveness properties

1. Absence of thread starvation and fairness

One can confidently say that a program doesn't starve / choke its threads if the threads are well managed. In my program, I have used the **ExecutorService** and **Executors** interface which manages the threads created fair and square. It queues the tasks that are yet to be executed until at least one of its threads become free. The thread that becomes free first takes up the next task in the queue. This ensures **absence of thread starvation** and **fairness**.

Safety properties

2. Absence of deadlocks

I have used **blocking queue** to ensure that there is **no deadlocks**. The barber thread will pool trough customer queue when customer are there. Any one of the barber will pick the customer and perform the haircut task on it. Upon arrival of a critical section, only one thread picks the task and executes it and ensures that other threads don't gain access to the critical section until the first thread leaves the critical section. There will **never** be a situation when two or more threads block each other.

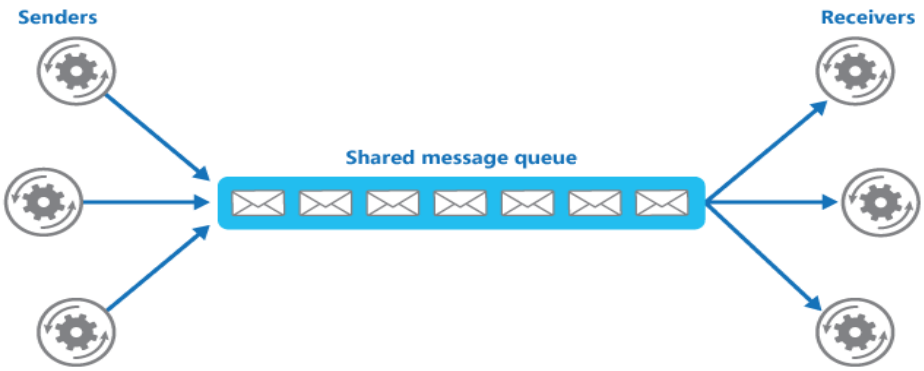
4. Mutual Exclusion

Each Barber is a thread of the thread pool and customer is part of the blocking queue. Once a customer is available , any one of the barber will pick the task waiting for the longest time (FIFO) and execute it.

IV. Real life examples

1. Call Centre: A simplest example of a sleeping barbers' problem are call centres. Call centre department is commonly observed in many industries e.g. Telecom, Airlines, E-commerce. It is a closest reference to a sleeping barbers' problem. Where the call executives act as the barber, attending all the queries of the customer. One customer at a time. There is a queue which is like a waiting room in the sleeping barber problem. Each call centre executives and customer can be assumed as a independent thread

2. Messaging Queue: Considering the computing system, message queue is a good example of a sleeping barber problem. A Message Queue is a messaging infrastructure to allow different systems to communicate through a shared set of interfaces (message queue). Message Queue contains FIFO (first in first out) rule. It consist of multiple messages (customers from sleeping barbers) and multiple receivers (Barbers). A shared message queue which is similar to the waiting list.



V. TEST CASSES:

Using VisualVM we can see the working of threads which are going into active and sleep state.

The pool threads are the Barber threads based on the number of Barbers while the rest thread is a customer thread.

References:

[1] "Guide to java.util.concurrent.Locks | Baeldung." [Online]. Available: https://www.baeldung.com/java-concurrent-locks. [Accessed: 18-Mar-2020].

[2] "Java Language - Multiple producer/consumer example with shared global queue | java Tutorial." [Online]. Available: https://riptutorial.com/java/example/13011/multiple-producer-consumer-example-with-shared-global-queue. [Accessed: 18-Mar-2020].

[3] "Java Thread Pool Executor Example," *HowToDoInJava*. [Online]. Available: https://howtodoinjava.com/java/multi-threading/java-thread-pool-executor-example/. [Accessed: 18-Mar-2020].

[4] SVyatkin, "Sleeping Barber Problem," *Express Notes by Sergey Vyatkin*, 22-Dec-2013. [Online]. Available: https://vyatkins.wordpress.com/2013/12/21/sleeping-barber-problem/. [Accessed: 18-Mar-2020].

[5] "Message queue," *Wikipedia*. 28-Jan-2020.

[6] "Thread Pools in Java - GeeksforGeeks." [Online]. Available: https://www.geeksforgeeks.org/thread-pools-java/. [Accessed: 18-Mar-2020].

[7] "Synchronization in Java - javatpoint." [Online]. Available: https://www.javatpoint.com/synchronization-in-java. [Accessed: 18-Mar-2020].