**ASSIGNMENT 1 Sleeping barber with multithreading and multicore**

**Declaration**

|  |  |
| --- | --- |
| Name | Vikrant Singh |
| Student no. | 21262315 |
| Programme | MSc. In computing (secure software engineering) |
| Module Code | CA670 |
| Assignment title | Assignment 1 - Java Threads |
| Submission Date | 25th Feb, 2022 |
| 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: Vikrant Singh Date: 25 Feb 2022

**Approach**

**Step 1.** First let’s create classes for Baber and Customer so that we can create multiple Barber and customer instances out of it, and they are runnable because we are going to create multiple threads out of those.

**Explanation of Barber.java (It’s class which is specifically handle data and task of barber)**

**Step 2. *Run ()*** function handle locks which are semaphore with Boolean in this case on the customer and barber objects.

* Customers\_ready.acquire(); // if barber is free get the lock on the customer
* Barbers\_ready.release(); // release the barber lock when he is finished with the haircut

**Step 3. *Cut\_hair ()*** is a function which take out customer from the double ended queue and perform hair cut and make customer leave after having the haircut and show time taken to perform hair cut of every customer and the barber which serves the customer.

**Explanation of customer.java (It’s class which is specifically handle data and task of customer)**

**Step 4.**  Similarly, as we did in baber.java we create a class with runnable and make a run function which acquire locks as follows:

* Customers\_ready.release(); // release the lock on the customer when customer is ready for the hair cut
* Barbers\_ready.acquire(); // acquire the lock on the barber if he is free to cut the customer the hair

Also, check if the availability of Baber so that we can add customer to the waiting queue if required.

**Explanation of class sleeping\_baber.java (This class handles the multithreading and multicore operations)**

**Step 5. *Main ()*** starts by taking two user inputs which are number barber in the shop and other input for the number of waiting chair.

**Step 6.** And then checked and print the number of cores available in a system using

* **int** cores = Runtime.*getRuntime*(). availableProcessors();

**Step 7.** Then using Executer Service for multicore processing as follows:

* **final** ExecutorService executor = Executors.*newFixedThreadPool*(cores);

A thread pool is created according to the number of cores available in the system.

**Step 8.** Then create multiple instances of Baber class as specified by number\_of\_baber from user input and using execute method spread them across multiple cores.

**Step 9**. Now create a infinite while loop which keep on creating an customer thread instances using a sleep generated randomly and also check for the availability of space in waiting area and if no space make customer leave the shop.

**Correctness**

Program is working as I tested it with multiple test input lie changing the number of barber and availability of number of chairs in waiting area and, I am using a barber number/id and customer/id to keep track that the customer get haircut from the assigned barber. And if no space available then customer look for waiting chair and if it’s also not available then customer leaves the shop, I also tested this by increasing the customer generated so that customer enters the shop at faster rate. Everything is working fine.

**Fairness/ Starvation**

As we are using blocking double ended queue so every thread will get a chance i.e., means every customer will get the barber in the order they inserted because we are inserting at one end and taking element from other end and also double ended queue stops illegal insertion and removal so its better to use with threads.

**Deadlocks**

As we are using semaphore to acquire and release the shared resources, where customer can acquire lock and call the ***cut\_hair ()*** and after haircut barber release the lock and also customer release the lock and look for availability and if available then barbe acquire a lock, otherwise customer send to the waiting area. So, they kind of synchronizing locks with each other so only one customer can access the single barber at a particular time.

**Output Samples**

**Sample 1: With barber = 1 and waiting chairs = 10**

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

**Sample 2: With barbers = 3 and waiting chairs = 3**

Text

Description automatically generated

Text

Description automatically generated with medium confidence

**Sample 3: With barber = 5 and waiting chairs = 1**

This case is quite interesting as my system has only 4 cores so only 4 barber instances are created**.** So, we can create barbers according to number of cores as mentioned above;

Text

Description automatically generated

**Practical use sleeping barber**

I think the use of sleeping barber is like how some servers handle the request and resource allocation on cloud. As, they allow time to each person/request to access the resources and no other can access those resources in that time and everyone is given a time frame to work and if all the resources are in use the user must wait or they can leave the server.

# **References**

|  |  |
| --- | --- |
| [1] | C. Turkoglu, “Java Concurrency with Barbershop Problem,” 30 May 2020. [Online]. Available: https://turkogluc.com/java-concurrency-sleeping-barber/. [Accessed Feb 2022]. |
| [2] | ManasiKirloskar, “BlockingDeque in Java,” 24 Sep 2020. [Online]. Available: https://www.geeksforgeeks.org/blockingdeque-in-java/. [Accessed Feb 2022]. |