Name: Aniket Suryakant Angwalkar

Student ID Number: 23269100

Email: aniket.angwalkar2@mail.dcu.ie

Program of Study: M.Sc. in Computing (Secure Software Engineering)

Module Code: CA670

Module Name: Concurrent Programming

Date of Submission: 01/02/2023

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. Any use of generative AI or search will be described in a one-page appendix

including prompt queries.

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 DCU Academic Integrity and Plagiarism Policy.

Signed Name: Aniket Suryakant Angwalkar

Date: 20/10/2023

ASSIGNMENT 1

Design of the Sleeping Barber problem:

The program for this solution has been divided into three different Java Classes, namely:

1. Barbers
2. BarberShop
3. Customers

The BarberShop is the class that contains the main class and is the execution point for the program.

BarberShop.java:

BarberShop class is the main class that handles creation of threads for customers and barbers.  
This class also contains execution of functionalities such as enterShop(), leaveShop() and serveCustomer().

This class also consists of the main class that starts the execution of the program.

Barbers.java:

The Barbers class is responsible for creation of barbers for the program as specified in the command line arguments.   
This class also overrides the run method from Runnable import of the java library.   
The run method executes the haircut service functionality for the customers and sleeps for 2 units of time to simulate the hair cutting service.

Customers.java:

Similar to the barbers class, this class handles the creation and simulation of customers.  
The run method that is overridden from the Runnable import handles the customer entering the shop and customer leaving the shop.

Justifications:

1. Mutual Exclusion:

The synchronized keyword with the lock in the functions: “enterShop()”, “leaveShop()”, and “serveCustomer()” ensure that only one thread will be allowed to execute these sections of the code at any given moment.   
This prevents access to shared resources to multiple threads simultaneously.

1. Deadlock Absence:

The synchronized blocks in the BarberShop class ensure that the BarberShop class is accessed by the threads in the same order.  
Maintaining this order prevents a deadlock occurrence and indefinite waiting.

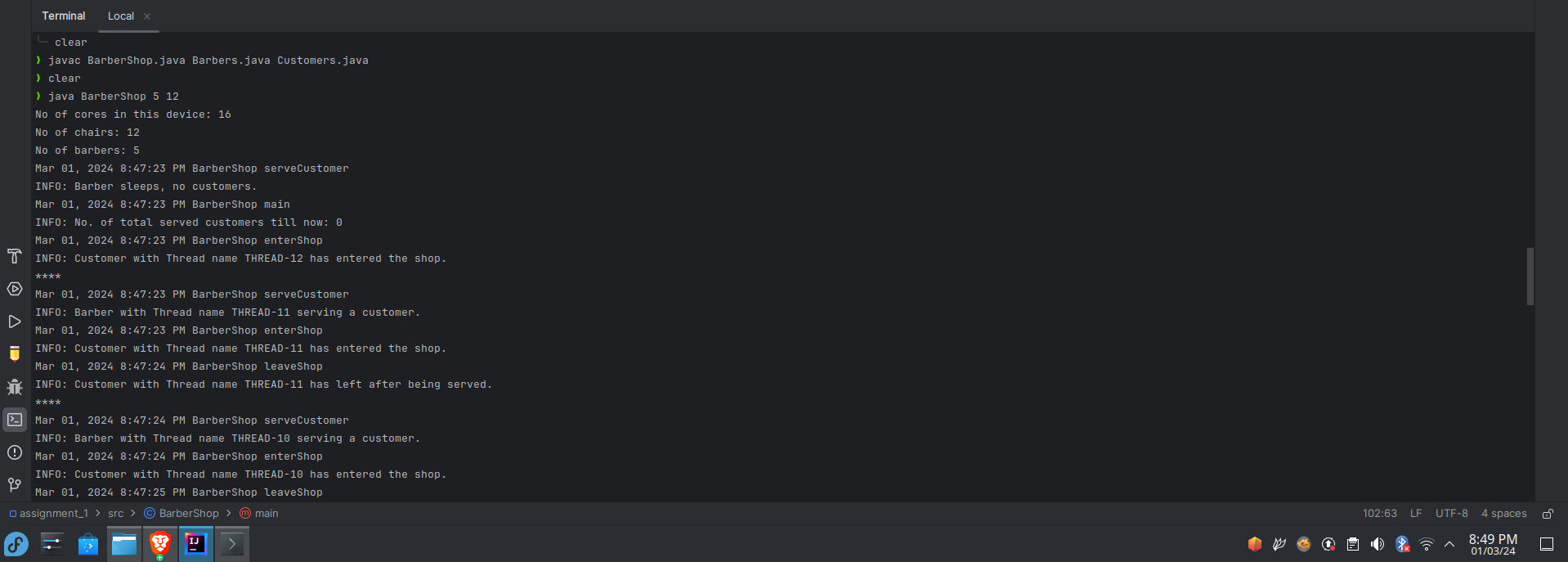
1. Absence of Starvation:

In the case that the new customer arrives, and all barbers are busy serving, the customer is added to the waiting list to be served.  
This ensures that all customers waiting will be eventually served on a FIFO basis, preventing starvation of any of the threads.

1. Fairness:

The FIFO mechanism preserves the fairness of the program by serving customers in the order that they arrive.   
The implementation of synchronized blocks also ensures that each thread gets access to the shared resource in an equal manner to ensure justified resource allocation.

Output:



Use Case:

The Sleeping barber problem efficiently bring into notice, the problem or resource management for distributing the workload among these resources.

One extremely common scenario where the Sleeping barber problem can be extended to be used is the workload management scenarios of Microservice Applications.

The load balancing mechanism of Microservices applications are the perfect use case for this problem as the sleeping barber problem allows the efficient management of resources.

This can be utilized for managing web servers for these applications by making request receiving process utilize the techniques of this problem and formulate the workload for each request to be handled by the web servers.