**Design of Program**

**Overview:**

Initially there are 3 barbers in salon sleeping in their chairs till customer arrives. There are 3 chairs in waiting room. As customer arrives, he checks for available barber. If the any one barber is available, then he goes and wakes him. As customers begin to enter the shop, each barber gets busy. If all barbers are busy, then a customer is offered a seat in waiting area. But if all the chairs are already occupied by other customers, then customer walks out from the shop. As any barber finishes his work, he calls next customer from waiting area.

The program is implemented using wait() and notify() methods. The basic implementation of wait() and notify() was referenced from (Anon., 2014) .

**Detailed explanation of used methods:**

**Barbers**: In this implementation I have considered 3 barbers working synchronously. Each barber is a single thread. They are executed on multiple cores using Executor service. Initially all barbers are waiting for customer to arrive in shop, till then they all sleep in their chairs. Whenever customer arrives, a barber is assigned to cut his hair and a barber is decremented from available barber pool. After serving the customer, barber guides customer to exit door and sleeps till customer leaves the shop. The barber then checks if there are any other customers present in waiting room to get a haircut. If present, then he calls the customer based on first come first serve basis.

**Customers:**  20 customers are generated in this program after specific intervals. Each customer is a separate thread and these threads are executed on multiple cores using Executor service. Customers enter the shop and check the status of barbers(i.e. if barbers are busy or free). If the barber is free and sleeping in his chair, then the customer wakes him and sits in the cutting chair. And if barber is already busy cutting other customers hair, then customer goes to waiting room. I have considered 3 waiting room chairs in this program. Here, customer looks for vacant waiting room chairs. If he finds the vacant chair, then he acquires that chair (which in turn decrements waiting room chair count). If there are no chairs in available in waiting area, then that customer leaves the shop without a haircut.

**Executor Service:** Executor service is used to execute the task concurrently that are assigned by threads. ThreadPoolExecutor implementation of Executor service is used in the program. All barber threads and customer threads are executed by ExecutorService. After all threads are executed, the executor is shutdown.

**Synchronized block:** Synchronized blockareused in program to avoid race condition.This block is synchronized on **listCustomers**, so that this list is accessed by only one thread at a time.Two synchronized blocks are implemented in the code: One where customer is added to the list of customers and second where barber picks up the customer from the list.

**wait() and notify():** wait() forces the current thread to wait until some other thread invokes notify(). Here, I have used wait() on listCustomers so that barber waits for customer to arrive in shop. Once the customer enters the shop, notify() is used to wake up the thread waiting for customer.

**Linked List:** A linked list was maintained for the customers arriving the shop. Size of the linked list was the number of chairs in waiting room. Also, each customer was served by barber on first come first serve basis. Barbers pick up the customer from this list and gives him haircut.

**Use of mean and std deviation:** Customer arrive in shop at random interval with mean: 500 and std dev: 100. Also, a random time period is taken to cut customers hair that has mean: 2000 and std dev: 500. These random intervals are generated using nextGaussian() function.

**Efficiency:**

* In this program, customer does not check the status of barber(free or busy) after specific intervals. This would how not been an efficient implementation. Instead, barber is the one who calls the customer from waiting room.
* Barbers thread goes to sleep state until customer arrives in the shop
* Barbers thread goes to sleep state while holding door for customer to leave the shop after haircut

**Correctness properties:**

1. **Safety properties:**

**Freedom from deadlock:** A deadlock may occur in a case where barber is holding the door for another customer, and at that time a new customer enters the shop. He finds barber is busy, so he goes to waiting room. At the same time barber gets free and looks for customer in waiting area, and finds no one, as customer is still going to waiting room. Thus, barber goes to sleep thinking there are no customers and customer goes to sleep thinking no barbers are available.

To prevent such scenario, we have synchronously updated the barber count and customer arrival time. At a time only one condition may occur, that is a customer may arrive and return to waiting area or barber may get free and looks for customer in waiting area. Both conditions do not occur simultaneously.

1. **Liveliness properties:**

**Freedom from starvation (FIFO):** Each customer is served in a specific order. If there are already customers in waiting room, and if customer10 walks in and barber has just finished cutting hair of other customer, then barber will not pick customer10 for haircut, instead he will pick a customer from waiting room based on first come first serve basis. Therefore, no customer will face starvation in such conditions.

**Example where sleeping barber problem is used in practice**

Consider a movie theater ticket counter. Let us assume there are 3 counters selling the tickets of a movie. Initially all the employees present at counter are relaxing till customer arrive. Now a person comes to watch a movie, so he visits the ticket counter to buy a ticket. He first checks available ticket counter(look for any ticket counter which is not serving any customer). If he finds any such counter, then he goes to that window and buys the ticket. If there are no available counters, then he waits in a queue till a counter becomes available. This queue is a FIFO (First In First Out) queue.

Also, no two persons are allotted a single seat in the theater. Consider 2 people at different counters simultaneously request for the same seat to be booked, the employee locks that seat first will be able to generate the ticket for that seat. And another employee trying to access that seat will be asked to choose another seat.

# **References**

Anon., 2014. *Oracle Java Combo.* [Online]   
Available at: https://orajavasolutions.wordpress.com/tag/sleeping-barber-problem/  
[Accessed 14 03 2020].