**Multiple Sleeping Barber problem**

1. once you run code you should define number of barber and number of waiting seats in shop and then open shop

**Pseudocode Solution**

**//let's speak about shop class**

public class Customer implements Runnalble{

@Override

public void run() {

Shop.getInstance().CustomerEnterSop(customerName,Id);

}

}

Customer EnterShop{

Semaphore BarberIsReady;

Semaphore CustomerIsReady;

Semaphore WeaitingSeatsAvailable;

BlockingDeque<String> customersQueue;

// custome enter shop

// check if No Seats Available enter condion and will be in weating list

// and waite untill be Available chair from weating seats

if (WeaitingSeatsAvailable.availablePermits() == 0) {

System.out.println(customerName + " in Waeting List");

}

WeaitingSeatsAvailable.acquire();

// send signal to barber to tell him Iam Here

//frame.ChangeLabelText(customerName+" Enter Shop");

CustomerIsReady.release();

customersQueue.put(customerName);

// if no chair availble i will set in weating Area

if (BarberIsReady.availablePermits() == 0 && customersQueue.size() <= WeatingSeats) {

System.out.println(customerName + " in waeting seats");

frame.UpdateRowWaitingList(Id,"");

frame.UpdateColumnWaitingSeats(Id,"Now Here");

}

BarberIsReady.acquire(); c

WeaitingSeatsAvailable.release();

}

public class Barber implements Runnable {

Semaphore BarberReady;

Semaphore CustomerAvailbel;

Semaphore ChairAvailbel;

private BlockingDeque<String> customersQueue;

private String barberName;

@Override

public void run() {

while (true) {

try {

CustomerAvailbel.acquire();// customer enter shop

ChairAvailbel.acquire(); // if chair availble

getHaircut();

ChairAvailbel.release();// barber chair no is empty

BarberReady.release(); // barber ready for anthor customer

} catch (InterruptedException e) {

}

}

}

}

Once you run project should enter number of barbers in shop and number of waiting seats

what we used in project :

1. Singlton Pattern to create only one object from shop
2. CustomerReady oncec customer enter shop send signal to barber to wake him up if
3. BarberIsReady inilaized to barber count : check State of barber idle or not
4. WaitingSeats inilaized to weating seate : may be some customer enter shop and

see barber chair busy and waiting seats, also busy he will stay in waiting list once any chair of waiting seats will be free

1. use blockQueue to handle problem starvation to add customer once they arrive

**Examples of Deadlock:** when customers waits in the waiting room for the barber and both sleeping

**Solve the problem of deadlock:** we use CustomerReadySemaphore to send signal to barber if he sleep to wake up and BarberReady to see if he idle wakehim up if he busy wait in waiting seats

**Example of starvation problems 1:** when a customer waits too long time waiting when the customer don't follow order

**Example of starvation problem 2:** when the barber calls out customer randomly

**Example of starvation problem 3:** customer 3 and customer 2 on the waiting seats

And customer 3 enter first to the barber

**solve the problem of starvation :** add customer once they arrive and baber take from it each customer in order, we solve the problem number 3 using a queue.

**explain for real world application:**

In this section we will apply sleeping barber problem on a bank system image we have a bank and two accountants that see the people who wants services and we have many chairs for customers in the bank.

if we have n chairs and those chairs are now n-2 and two customers went to the bank, then we don't have any available chairs for newcomers, so if we have new customers they will exit the bank as there is no chairs for them, now the accountant has finished the service of a customer and other customer must go to the window of the accountant, now we have again n-1 chairs so if a newcomer came from outside the bank he will stay in the queue of services,

**we will represent the deadlock** as the turn of each customer with the accountant, and we represent the deadlock when the accountants refuse to make any progress or they went to other place,

**represent the starvation:** we will represent the starvation as when a customer number 10 go to the accountant before customer number 7, 8 and 10.