Multiple Sleeping Barber Problem

Solution pseudocode:

In main function:

Initializing the number of barbers, customer id, number of customer and number of chairs.

Enter number of number of barbers and number of chairs in waiting room

Initializing with 12 threads as multiple of cores in the CPU

Initializing the barber shop with the number of barbers

Create object from Bshop

Initializing a random number to calculate delays for customer arrivals and haircut

Initializing startTime to know current time millisecond

for (I =1) to (number of customer-1)

    Create object of barber

    Create new thread

    End For

for (I=0) to (Number of customers)

    Create object of customer

    Set to customer in time

    Create new thread for customer

    Set to customer id

    End for

try :

    Initializing random number for haircut

Catch:

    Throw interrupt exception

shutdown thread pool

Initialized elapsed Time

Print Barbershop closed

Print Total time elapsed in seconds + number of Customers + number of Barbers + number of Chairs + TimeUnit.MILLISECONDS

Print number of Customers+ shop.getTotalHairCuts () + shop.getCustomerLost

Create class Barber

Create object shop of Bshop

Declare barber ID

Create constructor

Declare shop and barber ID

Create run function

While (true)

Call cut hair method in shop object

End While

End run function

End Barber class

Create class Customer

Declare customer ID, shop and in Time

Create Customer constructor

Initialize shop

Create getCustomerIdfunction

Get customer ID

Create getInTime function

Get in Time

Create setcustomerId function

Initialize customer ID

Create setInTimefunction

Initialize in Time

Create run function

Call goForHairCut function

Create goForHairCut function

Call add shop function

create class Bshop

intialize totalHairCuts and customersLost

Declare nchairand noOfBarbers and availableBarbers and listOfCustomer

create object from rondom

create constractor from Bshop

intialize nchair ,listCustomer,noOfBarbersand availableBarbers

create function getTotalHairCuts

get totalHairCuts

create function getCustomerLost

get customersLost

create fuction add

Examples of Deadlock:

Deadlock in Java is a condition where two or more threads are blocked forever, waiting for each other. This usually happens when multiple threads need the same locks but obtain them in different orders

Example1: in CutHair function if list of customer(shared resource) can be accessed from multiple threads (barber is busy and another customer is wake up same barber)

Examble2: in Add function if list of customer (shared resource) can be accessed from multiple threads(a lot of customer enter in same time and think its no chair available then leaves but it can an available chair in waiting room)

How did solve deadlock:

1- Java provides a way of creating threads and synchronizing their tasks using synchronized blocks. Synchronized blocks in Java are marked with the synchronized keyword. A synchronized block in Java is synchronized on some object. All synchronized blocks synchronize on the same object can only have one thread executing inside them at a time. All other threads attempting to enter the synchronized block are blocked until the thread inside the synchronized block exits the block.

```
synchronized (listCustomer) {

    while(listCustomer.size()==0) {

        System.out.println("\nBarber "+barberId+" is waiting "
                        + "for the customer and sleeps in his chair");

        try {

            listCustomer.wait();
        }
        catch(InterruptedException iex) {

            iex.printStackTrace();
        }
    }

    customer = (Customer)((LinkedList<?>)listCustomer).poll();

    System.out.println("Customer "+customer.getCustomerId()+
                    " finds the barber asleep and wakes up "
                    + "the barber "+barberId);
}
```

```
    synchronized (listCustomer) {

        if(listCustomer.size() == nchair) {

            System.out.println("\nNo chair available "
                               + "for customer "+customer.getCustomerId()+
                               " so customer leaves the shop");

          customersLost.incrementAndGet();

            return;
        }
        else if (availableBarbers > 0) {

            ((LinkedList<Customer>)listCustomer).offer(customer);
                               listCustomer.notify();
                    }
        else {

            ((LinkedList<Customer>)listCustomer).offer(customer);

            System.out.println("All barber(s) are busy so "+
                               customer.getCustomerId()+
                               " takes a chair in the waiting room");

            if(listCustomer.size()==1)
                listCustomer.notify();
        }
    }
```

Examples of starvation:

EX 1: In Barbershop It's may be there a customer waiting in the waiting room and there were an available barber in the shop.

EX 2: In Barbershop It's may be there were a barber sleep and a new customer enter the shop but the barber still sleep

How did solve starvation:

Solution:

Call notify () method in list customer linked list

We use the notify () method for waking up threads that are waiting for access to this object's monitor.

```
synchronized (listCustomer) {

    if(listCustomer.size() == nchair) {

        System.out.println("\nNo chair available "
                           + "for customer "+customer.getCustomerId()+
                           " so customer leaves the shop");

      customersLost.incrementAndGet();

        return;
    }
    else if (availableBarbers > 0) {

        ((LinkedList<Customer>)listCustomer).offer(customer);
                           listCustomer.notify();
                    }
    else {

        ((LinkedList<Customer>)listCustomer).offer(customer);

        System.out.println("All barber(s) are busy so "+
                           customer.getCustomerId()+
                           " takes a chair in the waiting room");

        if(listCustomer.size()==1)
            listCustomer.notify();
    }
}
```

Real World Application (Clinic System)

clinic has a doctor and waiting room with some chairs for patients

patient enters the clinic if doctor is free then doctor diagnoses him

else if doctor is diagnosing patient the new patient enter to waitnig room if there are empty chairs in waiting room

else if waiting room is full patient will leave the clinic