

Sleeping TA

Project description:

The Sleeping TA System is a Java-based program designed to simulate the interaction between students and Teaching Assistants (TAs) in a dynamic office environment. The system represents a scenario where students require assistance, and TAs are available to provide tutoring while managing limited resources such as waiting chairs and number of TAs.

There are multiple scenarios the program deals with:

First scenario:

The numbers of the total students is not greater than the number of TAs and waiting chairs in this case all the students either gets tutored once they arrive or they wait on the waiting chairs until its turn and if the TA has no student to tutor he/she will sleep.

second scenario:

The numbers of the total students is greater than the number of TAs and waiting chairs in this case a specific number of the students either gets tutored once they arrive or they wait on the waiting chairs and this number is (number of TAs + number of waiting chairs) the remaining students will leave and come back later first check if there is available tutor if not checks if there is available chair in the waiting room and repeats this logic until he/she gets tutored then if the TA has no student to tutor he/she will sleep.

What did we do and how the program actually works ?

Multi-Threading Simulation:

-The program utilizes multi-threading to simulate the concurrent activities of multiple students seeking assistance and TAs providing tutoring.

Office Environment:

- The system models an office setting with a specified number of waiting chairs available TAs.
- TAs provide tutoring to students, and students enter the office to seek assistance.

Resource Management:

- The system manages the availability of TAs and waiting chairs to ensure efficient use of resources.
- It demonstrates how TAs handle multiple students concurrently and efficiently.

Randomized Student Arrival:

- Students arrive at the office at random intervals, introducing variability to the system.
- The randomness adds a realistic touch to the simulation, reflecting the dynamic nature of real-world scenarios.

Time Tracking:

- The system records the total time elapsed for providing tutoring to all students.
- It tracks the total number of students tutored and the number of students who return later.

Exception Handling:

- The program handles exceptions such as waiting room chairs being full and students -leaving to come back later.
- It ensures that the simulation runs smoothly, accounting for unexpected scenarios.

User Input:

-The user can input parameters such as the number of students, waiting chairs, and TAs to customize the simulation.

Output Display:

-The system provides real-time output, displaying the progress of the simulation, including the status of TAs, waiting students, and later students.

How to Use:

1-Run the program and input the number of TAs, students, and the maximum number of waiting students.

2-Observe the dynamic interaction between students and TAs in the simulated office environment.

3-Review the output to understand the utilization of resources and the efficiency of the system.

Team members role:

Four of us searched for the optimal way to make the project as well as the most efficient way to implement it with coding after the deep search they reached the optimal way which we used in the coding of this project . Two of us were responsible for creating the GUI and the documentation and one was responsible for connecting the whole project together and testing it.

Code documentation:

Sleeping TA class

```
public class SleepingTA {

    public static void main(String a[]) throws InterruptedException {
        int noOfTA= 2, customerId = 1, noOfstudents= 100, noOfChairs;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of TA:");
        noOfTA = sc.nextInt();
        System.out.println("Enter the number of waiting room"
            + " chairs(N):");
        noOfChairs = sc.nextInt();
        System.out.println("Enter the number of Students:");
        noOfstudents = sc.nextInt();
        ExecutorService exec = Executors.newFixedThreadPool(12);
        Bshop office = new Bshop(noOfTA, noOfChairs);
        long startTime = System.currentTimeMillis();
    }
}
```

TA class

```
class TA implements Runnable {

    int TAId;
    public TA( int TAId) {
        this.TAId = TAId;
    }
    public void run() {
        while (true) {
            office.tutor(TAId);
        }
    }
}
```

Generating students threads:

```
for (int i = 0; i < noOfCustomers; i++)
{

    Customer customer = new Customer(shop);
    customer.setInTime(new Date());
    Thread thcustomer = new Thread(customer);
    customer.setcustomerId(customerId++);
    exec.execute(thcustomer);
}
```

Random delay between 500 to 600 milliseconds for student arrival

```
try {

    int millisDelay = r.nextInt(1000)+500;
}
```

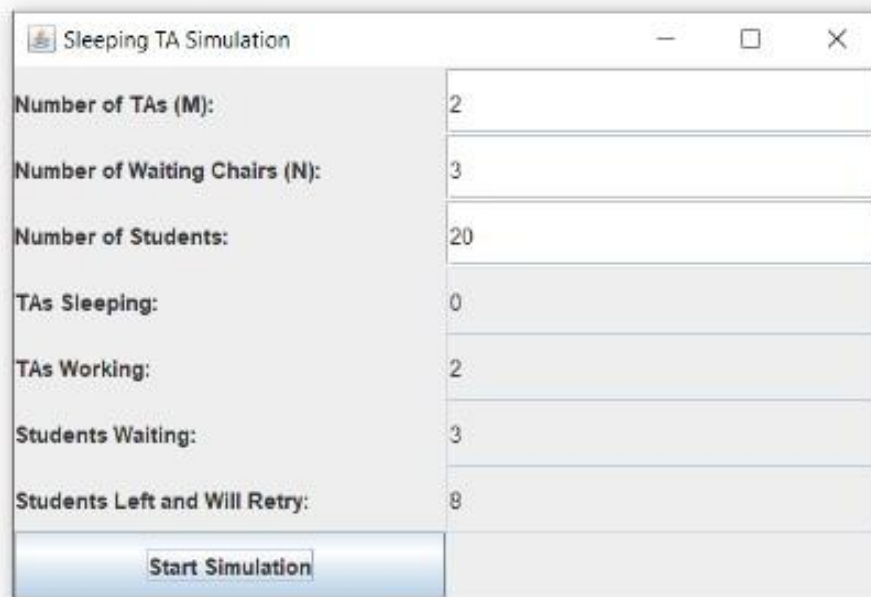
```
        Thread.sleep(millisDelay);
    } catch (InterruptedException iex) {
        iex.printStackTrace();
    }
}
```

TA sleeps if there are no students to tutor

```
try {
    TASleeping.incrementAndGet();
    listStudents.wait();

    TASleeping.decrementAndGet();
} catch (InterruptedException iex) {
    iex.printStackTrace();
}
```

Graphical User interface:



The screenshot shows a Java Swing window titled "Sleeping TA Simulation". It contains a table with 7 rows and 2 columns. The first column contains labels for simulation parameters and current states, and the second column contains their corresponding values. At the bottom of the table is a "Start Simulation" button.

Label	Value
Number of TAs (M):	2
Number of Waiting Chairs (N):	3
Number of Students:	20
TAs Sleeping:	0
TAs Working:	2
Students Waiting:	3
Students Left and Will Retry:	8

Start Simulation