**Project Number: 1**

**Project Name: The Sleeping Teaching Assistant**

**Team Members1:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Team Member ID | Team member name (**in Arabic**) | Grade |
| 1 | 20210064 | أحمد صلاح الدين أحمد سيد |  |
| 2 | 20210063 | أحمد تحسين صلاح الدين عبدالمجيد |  |
| 3 | 20210124 | أحمد هاني سعد محمد علي |  |
| 4 | 20210159 | إسلام مجدي محمد الصفي |  |
| 5 | 20210074 | أحمد عبدالغني أحمد عبدالغني |  |
| 6 | 20210081 | أحمد علي إبراهيم سليمان |  |
| 7 | 20210057 | أحمد سعدالدين محمد علي المراكبي |  |

Evaluation Criteria

|  |  |  |
| --- | --- | --- |
| Critera | | Grade |
| **Multithreading (5)** | No multithreading ( 2 out of 5 ) | **………………………………………………** |
| Threads in serial (3 out of 5)  Correct usage of threads, and synchronization mechanisms |
| Multithreading (4 or 5 out of 5)  Correct usage of threads, and synchronization mechanisms |
| **GUI (2)** | No GUI (0 out of 2) | **…………………………………………………** |
| GUI without thread communication or  realtime update (1 out of 2) |
| GUI with correct I/O and Thread communication or realtime update (2 out of 2) |
| **Documentation (1)** |  |  |
| **Understanding (2)** |  |  |

General Criteria

1 1st team member should be the same one in project schedule

إسم العضو الأول في الفريق يجب أن يكون نفس الإسم المعلن في جدول المناقشة

**Project Description:**

A university computer science department has a teaching assistant (TA) who helps undergraduate students

with their programming assignments during regular office hours. The TA ’s office is rather small and has room

for only one desk with a chair and computer. There are three chairs in the hallway outside the office where

students can sit and wait if the TA is currently helping another student. When there are no students who

need help during office hours, the TA sits at the desk and takes a nap. If a student arrives during office hours

and finds the TA sleeping, the student must awaken the TA to ask for help. If a student arrives and finds the

TA currently helping another student, the student sits on one of the chairs in the hallway and waits. If no

chairs are available, the student will come back at a later time.

Using JAVA threads, mutex locks, and semaphores, implement a solution that coordinates the activities of

the TA and the students. The number of disks in the TA’s room (aka no. of TAs), number of chair for waiting

students and number of students that have questions should be provided as an input to your program.

**Example Input:**

Number of TAs = 2

This indicates that there are two teaching assistants available in the computer science department to help undergraduate students with their programming assignments.

Number of Students = 20

This specifies the total number of undergraduate students who may seek assistance from the teaching assistants during regular office hours.

Number of Chairs = 3

There are three chairs placed in the hallway outside the TA's office. Students can sit on these chairs while waiting for their turn to receive assistance from the TA. If there are no chairs available, a student will have to come back later.

**Final Output:**

Number of working TAs: 0

Number of sleeping TAs: 2

Number of Students waiting on chairs: 0

Number of Students that will come later: 0

A computer screen shot of a computer screen

Description automatically generated

Team Member Roles:

GUI: Eslam Magdy, Ahmed Saad

Documentation: Ahmed Abdelghany

TA class: Ahmed Tahseen, Eslam Magdy

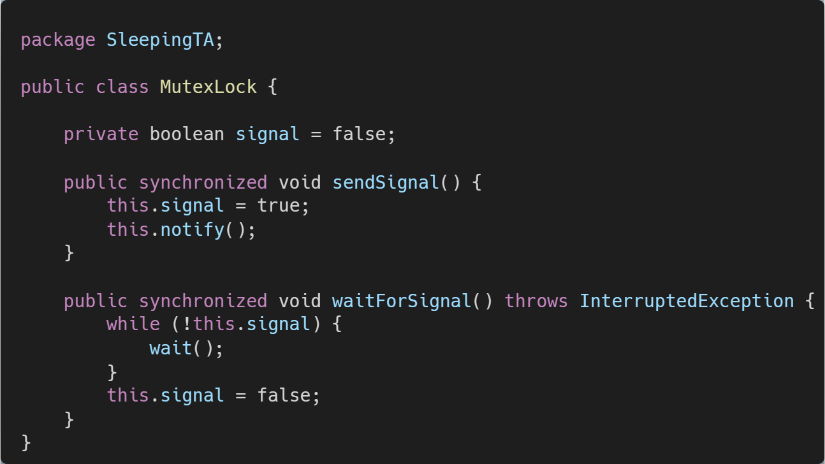
Student class: Ahmed Salah, Ahmed Hany

MutexLock class : Ahmed Ali

Video: Ahmed Hany

Code Documentation:

1- MutexLock:



The MutexLock helps parts of a program work together, making sure they don't step on each other's toes. It's like a friendly way for different pieces of the program to say, "I'm ready!" or "Something's happening!" so they can all work smoothly without causing problems.

sendSignal: This action is like raising a flag. It says, "Hey, something is happening!" It wakes up one waiting part of the program to let it know that things have changed.

waitForSignal: This action is like waiting for the flag to go up. It checks if the flag is down. If it is, it patiently waits until someone raises the flag. Once the flag is up, it goes on to do its job, making sure to put the flag down again for the next time.

2- Student:

the Student class simulates a student's interaction with a TA. The student programs, checks if the TA is free, and either works with the TA or waits in line. The process repeats until the student is interrupted or finishes their tasks. Print statements help understand what the student is doing at each step.

Attributes:

waitToAsk: Time a student spends programming before asking for help.

studentNum: Unique identifier for each student.

wakeup: A signal controller to wake up the TA.

chairs: Semaphore representing the chairs outside the TA's office.

TeacherAvailable: Semaphore indicating if the TA is available.

t: Reference to the current thread.

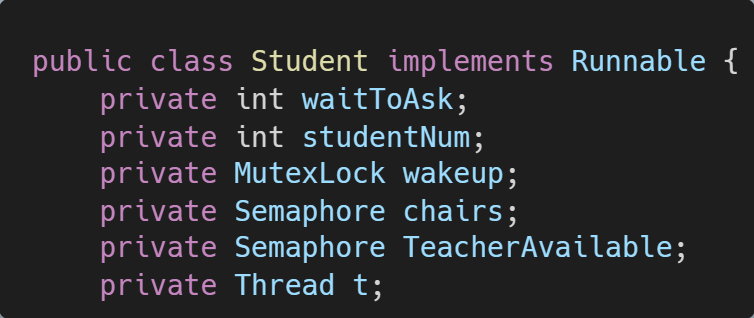
Run Method (Student's Actions):

The student goes into an infinite loop, alternating between programming and seeking help.

If the TA is available, the student wakes them up and works with them for a limited time.

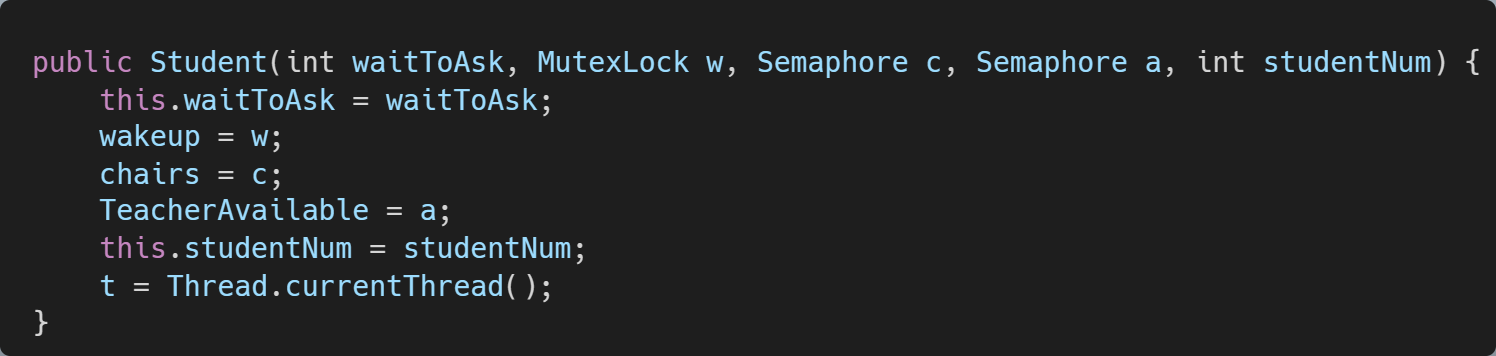
If the TA is busy, the student checks for available chairs; if found, they wait for their turn.

If no chairs are available, the student goes back to programming.

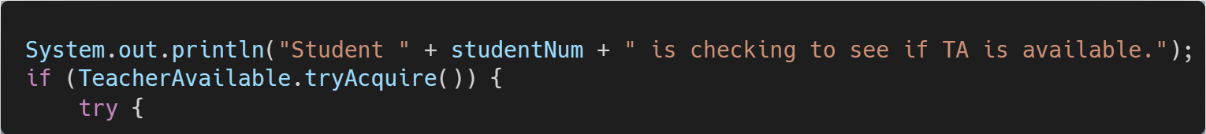


Constructor:

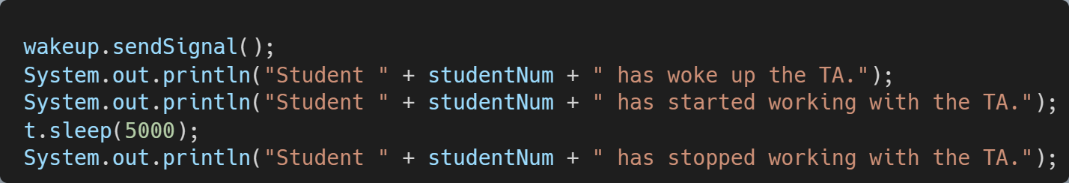
Initializes the student with programming time, signal controller, semaphores, and a unique student number.



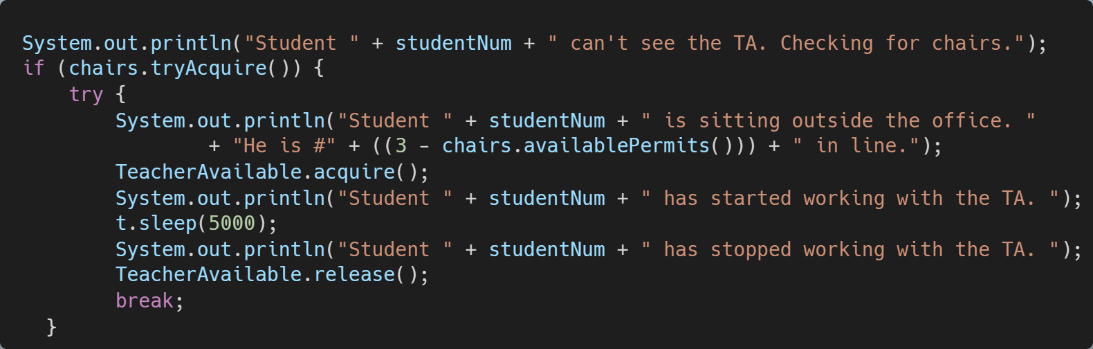
1- Check to see if TA is available first.



2- Wakeup the TA.



3- Check to see if any chairs are available.



3- TA:

Description:

the TA class simulates the actions of a TA in a computer science department. The TA naps when there are no students, wakes up when signaled, helps students, releases chairs, and goes back to nap. Print statements help understand what the TA is doing at each step.

Attributes:

signalTrigger: A signal controller to wake up the teaching assistant.

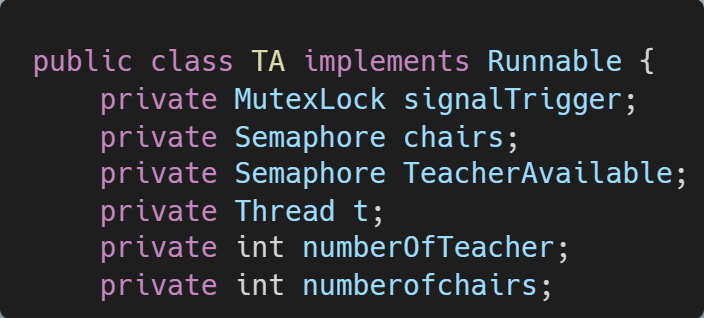
chairs: Semaphore representing the chairs outside the TA's office.

TeacherAvailable: Semaphore indicating if the TA is available.

t: Reference to the current thread.

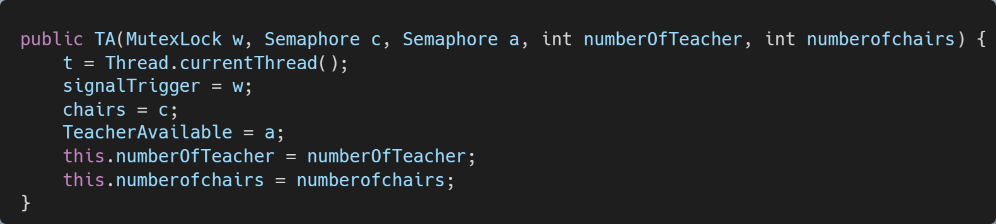
numberOfTeacher: Unique identifier for each teaching assistant.

numberofchairs: Number of chairs available outside the TA's office.



Constructor:

Initializes the teaching assistant with the signal controller, semaphores, and unique identifiers.



Run Method (TA's Actions):

The teaching assistant goes into a loop, checking if there are students waiting.

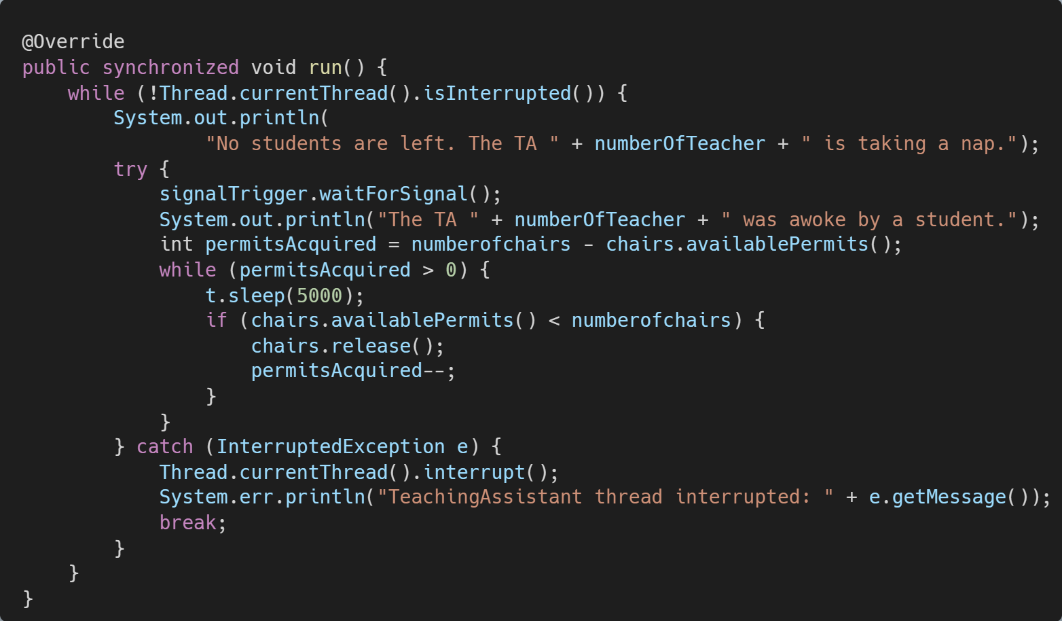
If no students are waiting, the TA goes to nap.

If a student signals for help, the TA wakes up and helps the student for a limited time.

The TA then checks for available chairs, releases them, and goes back to nap if no students are waiting.

Exception Handling:

Handles interruptions gracefully, printing error messages if an exception occurs.



3- GUI:

Description :

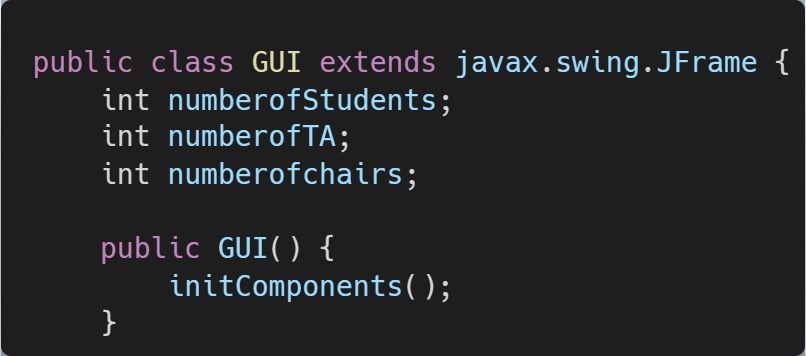
the code sets up a GUI for a teaching assistant system, takes user inputs for the number of TAs, students, and chairs, and uses threads to simulate students seeking help and TAs assisting them. The GUI continuously updates to show the current state of the system.

GUI Initialization:

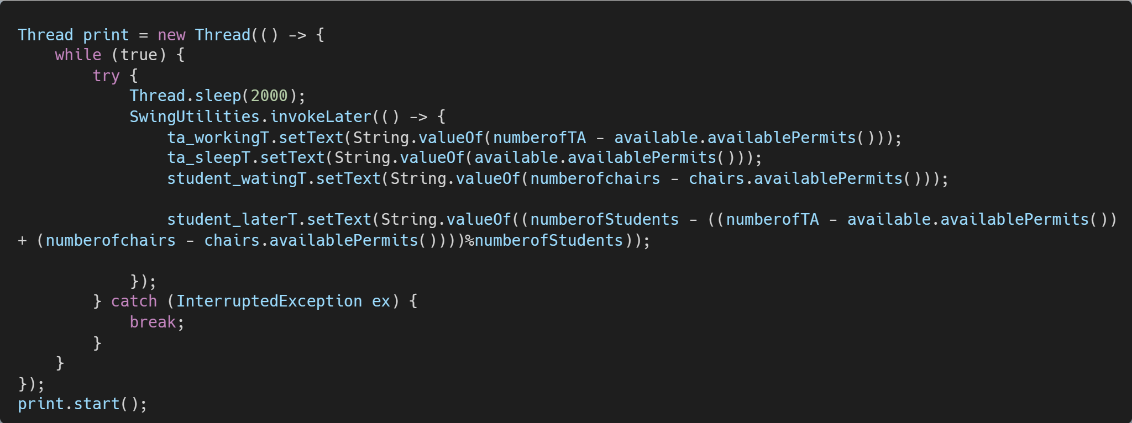
The GUI class represents the graphical user interface.

It has text fields (taT, studentT, chairT) for entering the number of teaching assistants, students, and chairs.

There are labels (ta\_workingT, ta\_sleepT, student\_watingT, student\_laterT) to display information about the system state.



Initialization and Threads:



When the user inputs the number of TAs, students, and chairs and clicks a button (assuming it exists), the program initializes:

A MutexLock for waking up the TA.

Semaphores for chairs and TA availability.

A random wait time for students.

Threads for students and TAs are created and started. Each student and TA run in their own thread.

The GUI displays the difference between the total number of TAs and available TAs, as well as the available TAs.

A separate thread continuously updates the GUI to show the current state of TAs working, TAs sleeping, students waiting, and students waiting for a later turn.

