

Project_#1 Sleeping Teaching Assistant

The Problem

1. **Teaching Assistant (TA):** Responsible for helping undergraduate students with programming assignments during regular office hours.
2. **Student:** Seeks assistance from the TA for programming assignments.

Rules:

1. The TA's office has limited space for one desk, chair, and computer.
2. There are three chairs in the hallway outside the TA's office for students to wait if the TA is busy helping another student.
3. When there are no students needing help, the TA may take a nap at the desk.
4. If a student arrives and finds the TA sleeping, the student must awaken the TA to request assistance.
5. If the TA is currently helping another student, incoming students must wait in the hallway.
6. If there are no chairs available in the hallway, the student will have to return at a later time.

Objective: Design a system to efficiently manage student interactions with the TA during office hours, considering the limited space, the TA's availability, and the need for an orderly waiting system.

Functional Requirements:

1. Track the TA's current status (available, helping a student, sleeping).
2. Maintain a count of available chairs in the hallway.
3. Implement a mechanism for students to request assistance and wait if necessary.
4. Ensure that students waking the TA from a nap are handled appropriately.
5. Provide a notification when a student is next in line for assistance.

Non-functional Requirements:

1. The system should be user-friendly for both the TA and students.
2. Ensure the system operates efficiently to minimize wait times.
3. Maintain the privacy and security of student information.

The solution steps

1. Identify Shared Resources:

Identify shared resources that need to be accessed by multiple threads (TA and students). In this scenario, shared resources include the TA's desk, chairs in the hallway, and the TA's status (whether the TA is available, helping a student, or sleeping).

2. Use Locks:

Employ locks to ensure exclusive access to shared resources. In this case, you can use locks to control students that access to the TA's desk, chairs, and the TA's status. For example, a mutex (mutual exclusion) lock can be used to control access to critical sections of code, and Semaphores that initialized with the

number of each of that resources and will be decreased when the student using any of them.

3. State Transitions:

Define and manage state transitions for the TA. States could include "**Busy**," and "**Sleeping**." Use locks to ensure that the TA's state is updated atomically, preventing inconsistencies.

4. Waiting Mechanism:

Implement a waiting mechanism for students when the TA is busy helping another student or sleeping. This can be achieved using conditional variables. When a student arrives and finds the TA busy, the student should wait on a condition variable until the TA becomes available which is defined as the "**Waiting Chairs**".

5. Notification Mechanism:

Ensure that waiting students are notified when the TA becomes available. This involves using conditional variables to signal waiting students that they can now access the TA for assistance.

The Final project

The Implementation is based on:

- Using semaphores as the guard of each level the student will face.
- Using the random values to give the model the character of realism like the number of students that will asking permission to enter the TA room, and the time each student will spend with the TA to get his answer.

The Input:

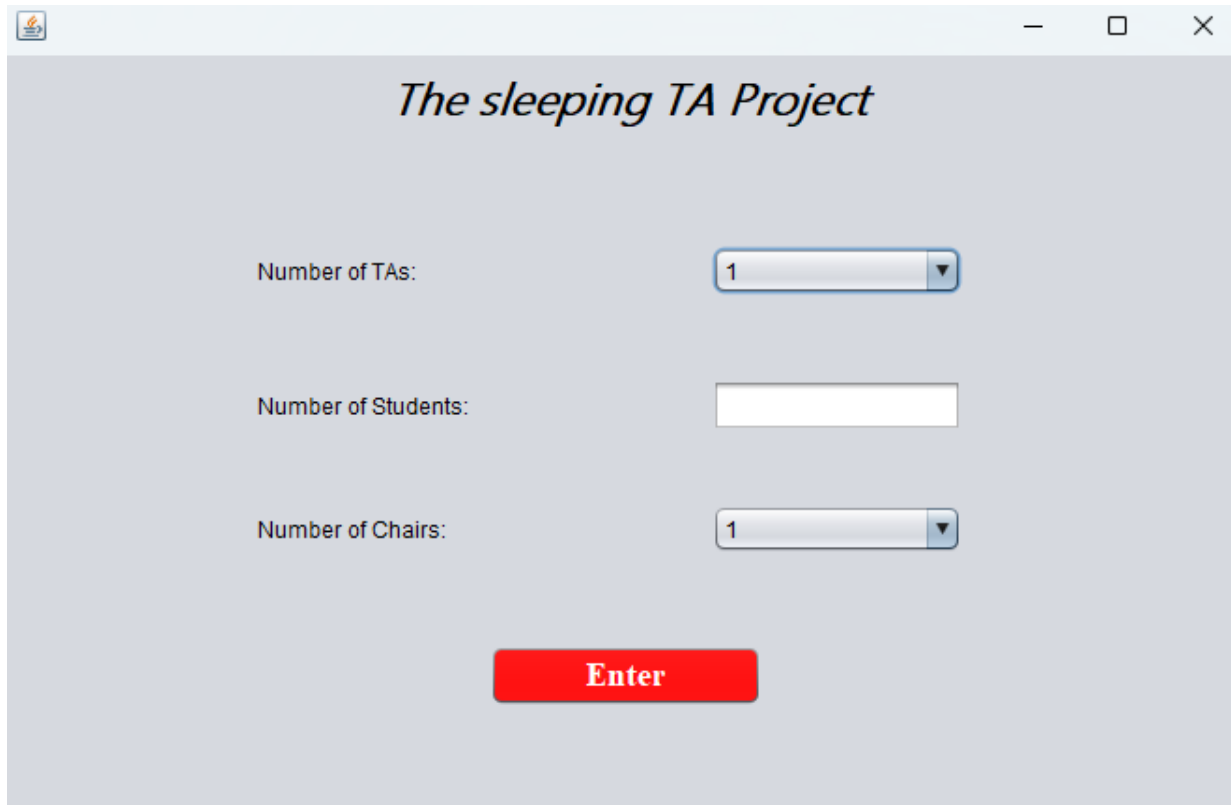
1. The number of Teaching Assistants.
2. The number of chairs (waiting list).
3. The number of students.

The Output:

1. The working number of Teaching Assistants.
2. The sleeping number of Teaching Assistants.
3. The number of used chairs (num of students are waiting).
4. The number of students that will come later.

GUI:

The input page



The screenshot shows a window titled "The sleeping TA Project". Inside the window, there are three input fields with labels to their left: "Number of TAs:" followed by a spinner box containing the number "1"; "Number of Students:" followed by a standard text input box; and "Number of Chairs:" followed by another spinner box containing the number "1". At the bottom center of the window is a prominent red button with the word "Enter" in white text.

The running panel

