Concurrent Programming
Practical 4: The Sleeping Tutor

The aim of this practical is to model the following scenario using monitors and/or semaphores. A tutor is sleeping in his room, waiting for two students to arrive for a tutorial. The first student to arrive waits for her tutorial partner. The second to arrive wakes the first, and one of them wakes the tutor. The two students then sleep while the tutor gives the tutorial. At the end of the tutorial, the two students wake up and leave. The tutor sleeps until the next tutorial.

For simplicity, we will assume that there is just a single pair of students, but they may return for multiple tutorials.

Figure 1 contains a trait that encapsulates the protocol followed by the tutor and students. It also contains a simulation of the scenario. Your task is to provide an implementation of the SleepingTutor trait. The two requirements are:

1. The tutor starts to teach only after both students have arrived;
2. The students leave only after the tutor ends the tutorial.

You should implement these procedures either:

- using a monitor, or
- using semaphores.

Optional: do both.

In addition, you should write test code for your implementation(s), in particular to test that the above two requirements hold.

Your report should be in the form of a well commented program, describing any design decisions you have made, and how your testing captures the requirements. **Deadline:** practical sessions in Week 8.

```
/** The trait for a Sleeping Tutor protocol. */
trait SleepingTutor{
  /** A tutor waits for students to arrive. */
  def tutorWait

  /** A student arrives and waits for the tutorial. */
  def arrive

  /** A student receives a tutorial. */
  def receiveTute

  /** A tutor ends the tutorial. */
  def endTeach
}

object SleepingTutorSimulation{
  // Some implementation of SleepingTutor
  private val st: SleepingTutor = new SleepingTutorMonitor

  private def student(me: String) = thread("Student"+me){
    while(true){
      Thread.sleep(Random.nextInt(2000))
      println("Student "+me+" arrives"); st.arrive
      println("Student "+me+" ready for tutorial"); st.receiveTute
      println("Student "+me+" leaves")
    }
  }

  private def tutor = thread("Tutor"){
    while(true){
      println("Tutor waiting for students"); st.tutorWait
      println("Tutor starts to teach"); Thread.sleep(1000)
      println("Tutor ends tutorial"); st.endTeach
      Thread.sleep(1000)
    }
  }

  private def system = tutor || student("Alice") || student("Bob")

  def main(args: Array[String]) = run(system)
}
```

Figure 1: The SleepingTutor trait, and a simulation.