We are going to use Java monitors to mimic students visiting TA night lab hours. In the night lab there is a TA who is available to help students. A TA brain can only help a certain number of students at a time. If the TA brain is helping the number of students at its capacity, the other students have to wait until the TA finishes with a student and frees up an opening.

For example, let’s say the TA brain can handle at most three students at a time. If the TA brain takes a student to help, it needs to make sure that it is not working past its capacity. If it is, it needs to make those students wait before they get access to the TA brain. Otherwise, it needs to increment the number of students it is helping and then grant access to the student.

Once the TA brain is done helping the student, it needs to decrement the number of students it is helping and notify students that are waiting for access to the TA brain.

We will need to implement a TA\_Brain class.

* It should have the necessary instance variables to make sure that is has a capacity and a number of students it is currently helping.
* It will need a constructor (to set the brain capacity )
* An accessor for the number of students it is currently helping.
* A takeAStudent method that takes a Student parameter. We have not made this class yet – but we will. I want to stress that we are doing this just for informative print statements – it is not needed. This method needs to make sure we do not go over the brain’s capacity and adjusts the number of students the brain is helping accordingly.
* A releaseAStudent method that takes a TA parameter. We have not made this class yet – but we will. I want to stress that we are doing this just for informative print statements – it is not needed. This method needs to decrement the number of students the brain is helping accordingly and notify that a space is free.

We will need to implement a TA class that will extend the Thread class (because we are not worried about inheritance).

* It will have two instance variables: a TA\_Brain and a String name.
* It will need a constructor to set these instance variables
* An accessor for the name
* A giveAdvice method that prints out advice. It will be formatted. It will print the TA’s name then the word offers: followed with your sage advice (be creative, but not offensive). For example, if the TA’s name is Geoff (yes, I am using my cat as an example here yet again), then the printed statement could be:

Geoff offers: Have you thought about switching majors?

* A run method because it is required. While the TA\_Brain has students it can help, then the thread should offer its advice and then have the TA\_Brain release a student it is helping passing it this TA.

We will need to implement a Student class that will also extend the Thread class.

* It will have two instance variables: a TA\_Brain and a String name.
* It will need a constructor to set these instance variables
* An accessor for the name
* A getAdvice method that prints a simple message. It will be formatted. It will print the student’s name concatenated to “ asks for help”. Example:

Student1 asks for help

* A run method because it is required. In this method the brain will take a student passing it this student and then it will get advice.

We will need a driver class. We will call it NightLab.

* Write a main method
  + Instantiate a TA\_Brain with a capacity of 3 students (this can be altered later).
  + Create an integer variable for the number of students that need to be helped in the night lab (make it larger than 3 – try 10 if you need a suggestion).
  + Create a Thread array using the variable you just made to determine the number of student threads
  + Using a for loop fill the thread array with new Student threads. For the name – concatenate the string “student” with the for-loop iterator.
  + Using another for loop start all of the student threads
  + Instantiate a TA, you can select its name. I will use a cat’s name like “Geoff”
  + Start the TA thread.

Answer the following questions and submit this document with your code.

Does the TA\_Brain ever run above its capacity? What causes this behavior?

No, the TA just make student waiting for TA to finish offer

Does the TA\_Brain ever run below its capacity? When?

Yes, some time there is not so many student.

Are all the students helped? Why?

No, although TA makes students wait in line and when one finished, the other one comes in, but when TA’s end and it’s finish

Can you predict the order in which the students will be helped? Why?

Yes, the student wait in line to get to TA, so who is waiting next should be the next be helped.