344-715 Project 1 – Due SATURDAY 22ND
A Summer Camp Day Commute

Note: the projects must be done individually. No exceptions.

They are ***s*** students, two buses, and ***c*** coordinators (three types of threads).

Students arrive at the Queens College Summer camp by car or by bus. The type of transportation is decided randomly (generate a random number between 0 and 1, if <.5 by car, otherwise by bus).
The commute time for cars and buses is simulated by sleep of random time.

Students who commute by bus are all waiting at the bus station. The bus capacity is ***bc***. Upon arriving at the bus station, students will gather in groups of size **bc** (for blocking use an object for each group).

They are two buses that commute back and forth between the bus station and college until no group is left behind. Once a bus arrives at the station, it signals the first formed group. Students get in the bus and then wait to be signaled when the bus arrives at the college. The bus commute takes a random amount of time. Once the bus is parked, the bus will signal the students of the group that were riding the bus, to get off the bus.

Next, students (independently) head to the gymnasium where they will wait for the campus day to start.

On the other hand, the students that commute by car (car commuting time is simulated by sleep of random time), on arrival at the college, will line up (in their cars) at the drop off station (each student-car blocks on a different object, similar to the implementation of rwcv).

Outside, there are ***c*** coordinators, who will rush to pick up the students in the order of their arrival.
Note: if there are 3 coordinators, these coordinators will help the students of the first 3 cars. They will not all jump to the first car only.
The coordinators will keep doing this until all students arriving by car, have been dropped off.

Once a student is off the car, (s)he will go to the gymnasium where they will wait for the campus day to start.

As soon as a first coordinator figures out that all students are in the gymnasium, (s)he will signal all students that the summer day has started. It will also let the other coordinators know that for now their job is done.

The commute to summer camp ended and all the threads will terminate.

Develop a monitor that will synchronize the three types of threads: student, buses, coordinator, in the context of the problem. Besides the specified notification objects, you should have at least one synchronized method.

The number of students *numStudents* ***s***, bus *capacity* ***bc***, *number coordinators* ***c*** should be consider arguments.

numStudents s 20

bus capacity bc 4

coordinators c 3

Closely follow the story and the given implementation details.

Choose appropriate amount of time(s) that will agree with the content of the story. I haven't written the code for this project yet, but from the experience of grading previous semester's projects, a project should take somewhere between 45 seconds and at most 1 minute and ½, to run and complete.

Do not submit any code that does not compile and run. If there are parts of the code that contain bugs, comment it out and leave the code in. A program that does not compile nor run will not be graded.

Follow the story closely and cover the requirements of the project's description. Besides the synchronization details provided there are other synchronization aspects that need to be covered. You can use synchronized methods or additional synchronized blocks. Do NOT use busy waiting. If a thread needs to wait, it must wait on an object (class object or notification object).

3. Main class is run by the main thread. The other threads must be manually specified by either implementing the Runnable interface or extending the Thread class. Separate the classes into separate files. Do not leave all the classes in one file. Create a class for each type of thread.

Add the following lines to all the threads you make:

```
public static long time = System.currentTimeMillis();
public void msg(String m) {
   System.out.println("["+(System.currentTimeMillis()-time)+"] "+getName()+": "+m);    }
```

There should be printout messages indicating the execution interleaving. Whenever you want to print something from that thread use: msg("some message here");

NAME YOUR THREADS or the above lines that were added would mean nothing.
Here's how the constructors could look like (you may use any variant of this as long as each thread is unique and distinguishable):

```
// Default constructor
public RandomThread(int id) {
    setName("RandomThread-" + id);
}
```

Design an OOP program. All thread-related tasks must be specified in their respective classes, no class body should be empty.

DO NOT USE System.exit(0); the threads are supposed to terminate naturally by running to the end of their run methods.

Javadoc is not required. Proper basic commenting explaining the flow of program, self-explanatory variable names, correct whitespace and indentations are required.

Tips:
-If you run into some synchronization issue(s), and don't know which thread or threads are causing it, press F11 which will run the program in debug mode. You will clearly see the thread names in the debug perspective.

Setting up project/Submission:

In Eclipse:
Name your project as follows: LASTNAME_FIRSTNAME_CSXXX_PY
where LASTNAME is your last name, FIRSTNAME is your first name, XXX is your course, and Y is the current project number.

For example: Fluture_Simina_CS344-715_p1

To submit:
-Right click on your project and click export.
-Click on General (expand it)
-Select Archive File
-Select your project (make sure that .classpath and .project are also selected)
-Click Browse, select where you want to save it to and name it as LASTNAME_FIRSTNAME_CSXXX_PY
-Select Save in zip format**, Create directory structure for files and also Compress the contents of the file should be checked.**
-Press Finish

Upload the project on BlackBoard.