CSCI 344-715 Fall 2022

Project 1 – Due Date: November 29

Note: the projects must be done individually. Use of third-party code is not allowed. Taking from and/or also giving work to others is considered plagiarism/cheating. There are no exceptions!

Halloween

Students commute to school very excited in the anticipation of the Halloween fun (*simulated by sleep of random time*). The teacher **waits** for them in the homeroom. Once **all** students are in, the teacher will give a brief lecture. Students will **wait** for the lecture to end.

As soon as the lecture ends, they will line up and **wait** for the candies (*have each student wait on a different object; use the same approach as in rwcv.java*).

The teacher will give each student in order a small bag of candies. When done, the teacher terminates.

Next it is time for trick-or-treating in the neighborhood. Based on their themes, there will be numGroup groups created. Each student will generate an integer between 1 and numGroups to determine the group he is in. The students will also display the group they belong to.

Before trick-or-treating a house, the students will take a break and wait to be invited by the house (owner). (*have the students belonging to the same group **wait** on the same object*)

There are *numHouses* houses. With each round (there are *numHouses* rounds), a house will be trick-or-treated by one group. With each round the house will pick one group and that group cannot be selected by another house during the same round. In the next round the house should pick a different group. Once the group is picked by the house, the house will signal the students of that group to get their candies. Next the house will **wait** until the students of the group are done picking the candies.

Each student will pick a random number of candies (number between 1 and 10). After picking the candies (sleep of random time), the students will again take a (long) break and **wait** as a group to be invited by a different house.

After *numHouses* rounds the treat-or-tricking ends. The houses terminate as well.

In the end, each group will compute and display its average number of candies (total of candies/number of students in the group). The group with the largest average of candies will be in 1st place. Next, students will terminate in the order of their ranking. Students in first place will terminate first, next the ones in the second place and so on (use wait and signal to enforce the termination order).

==========

Default number for threads:	numStudents	20
	numHouses	4
	numGroups	5

Develop a monitor(s) that will synchronize the threads (student, teacher, house) in the context of the story described above. Closely follow the details of the story and the synchronization requirements.

You can create any other methods/objects you may need. Make sure that any shared variable/structure is accessed in a mutual exclusion fashion.

Do NOT use busy waiting. If a thread needs to wait, it must wait on an object (class object or notification object).

DO NOT use packages.

Use only basic monitors as the ones discussed in class (**synchronized methods, synchronized blocks, wait(), notify and notifyAll**). **Use NO** other synchronization tools like locks, concurrent collections…..or even volatile variables.

Use the **age( )** method and appropriate **println** statements to show how synchronization works. It is important to have print statements before, inside, and after critical events. State clearly what the current thread executing the print statement is doing. Also, be sure to include the name of the thread and a timestamp relative to the start time of the program.

Between major events make sure you insert a sleep of random time. Also make sure you give the information necessary to understand what that event is.

Choose the 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 semesters' projects, a project should take somewhere between 20 seconds and at most 1 minute to run and complete. Set the time in such way that you don't create any exceptional situations.

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.

Besides the synchronization details provided there are other synchronization aspects that need to be covered. You can use synchronized methods or additional synchronized blocks to make sure that mutual exclusion over shared variables is satisfied.

The Main class is run by the main thread. The other threads must be 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 program flow, self-explanatory variable names, correct whitespace and indentations are required.

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_p1

Zip only the source files (no .rar) .Upload the project on Blackboard.