Instructor: Dr. Simina Fluture

CSCI 344-715 Spring 2021

School Day

Students follow a hybrid school schedule. On the in-face learning day students arrive at school (simulated by sleep of random time) and ***wait*** in the schoolyard to be allowed to enter the school. Prior to their arrival, students must complete and submit by phone a health questionnaire. Not everybody remembers to do so, about 15% of them forget. Keep track of which students completed their questionnaire and which didn't, by generating a random number for each student (let's say if the random number is below 15% total students – the student didn't complete their questionnaire). Once the school day starts (after most students arrive) the principal will call each student in the order of their arrival ((each student should block on a different object; use an implementation similar to the one in **rwcv.java)**. If the principal finds that the student didn't complete his/her questionnaire he will send the student home (the student thread terminates). If a student is late for school she/he will terminate as well.

Once the student is called and if his/her questionnaire has been completed, the next step is for the principal to decide if the student will be tested for COVID or not. This will be done randomly and 1 in 3 students will be tested. Students that are picked to be tested for COVID will all move on to the nurse's room and will **wait** for the nurse to arrive. The room can accommodate at most two students at a time, so students will **wait** for their test in pairs. The nurse will test a pair of student and then invite the next pair of students. The probability of being tested positive is 3%. If the student tested positive, he/she will be sent home (the thread will terminate) else he/she will move on to the classroom. If there are more than maxSick (default value 3) students testing positive the school will have to close and everybody, students and staff will go home (terminate).

Once all students that needed to be tested have been tested the nurse will leave (terminate)

Students who don't have to get tested will move on to the classrooms. The school day has 3 sessions and students will attempt to attend each of them: ELA, MATH, and PHYSICALEDUCATION in the schoolyard. There is a classroom for the ELA class, and a classroom for the MATH class. Each classroom has the capacity of studentsCap (default 4). Students will attempt to take the ELA class first. If the classroom is full, they will attempt to take the Math class and if that is also full they will go in the backyard for physical-education.

There are two teachers one for ELA, and one for MATH, the principal will conduct the PHYSICAL-EDUCATION session. Once the classrooms are full and all students are assigned to one of the activities the principal will be notified (you can have the last student to be assigned **signal** the principal). Everybody will **wait** for the session to start and complete (students and the 2 instructors)

The principal will wait by **sleep**ing a fixed interval of time, the ELA teacher and students in the ELA classroom will **wait** on one object, the MATH teacher and students in the MATH classroom will **wait** on a different object, the students in the backyard will wait on another object).

When the session's time elapses the principal (wakes up) notifies the teachers that the class has ended. The teachers will notify the students. Everybody will take a brief break and get ready for

the next session. If a student attended the ELA class she/he will attempt to attend the Math class, and the other way around. If the student doesn't find a seat in the classroom she/he will attend PHYS-ED in the backyard.

After the three sessions have been completed everybody is ready to go home (terminate). Before leaving each student should display in the order of attendance the session that they attended.

Develop a monitor(s) that will synchronize the threads (principal, instructor, nurse, students) 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. Default number for threads: principal 1 instructor 2

nurse 1
students 20

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

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 50 seconds and at most 2 minutes 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.

Instructor: Dr. Simina Fluture

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.