Halloween Parade and Circus Show

Students from two different schools will participate in a Halloween Parade. The students from one school wear blue costumes, and the other students wear orange costumes. Students arrive at the meeting place (simulated by sleep(random)) in front of the stadium and group in formations of two orange and one blue costume. Once a group is formed (students of the same group will wait on the same notification object), it will start parading around the stadium (sleep(random)). Only students in groups may march in the parade. The very last group can start marching even it is incomplete. After each circle around the stadium, students separate from their groups and take a snack break (simulated by sleep(random)). Next, they will gather in front of the auditorium entrance for a circus show.

A new circus show session starts every 1 hour and 30 minutes. If a show is already in session, students will wait (each student should block on its own notification object – similar to rwcv.java). When the previous show session ends, the students leave the auditorium. Students that are waiting to see the coming show then enter the room in a FCFS order (implement a platoon policy) and take one of the available seats. If there are no free seats, students, will attempt to see the next show. When the show is about to start, no student can enter and disturb it.

There will be a total of three shows and students are anxious to see all of them. However, throughout the entire day there will be a total of six parades (one stadium circle) one hour apart, and students must participate in at least three parades; they must alternate between the parade participations and the circus shows. Depending on how long they take to circle the stadium, how long the snack breaks take, and the availability of seats in the auditorium, not all students will have the chance to see all three shows.

Parades and show times
Parades: 11:00AM, 12:00PM, 1:00PM, 2:00PM, 3:00PM, 4:00PM, 5:00PM
Show times: 11:45AM, 1:15PM, 2:45PM

Students will terminate after the last parade ends or if they already managed to participate in three parades and watched 3 shows. Before terminating students will print a review with the following information:
Student (thread) name, what parade he participated in by giving its starting time, what shows he watched by giving its starting time.

Develop a monitor synchronization of three types of threads*: orange-uniform students*, *blue-uniform students*. You need to add a new thread *clock* that will keep track of the parade times and the show times.

Academic integrity must be honored at all times and no code sharing or cheating is permitted. Only submit code that you have written.

Implementation details:
The number of students and the initial number of available seats should be entered from the command line.
Default Values: numOrange = 16
numBlue = 8
numSeat = 7

Take in consideration of possibility of having a last incomplete group.

Clock Implementation:
You can consider 1 second = 10 minutes
Set clock priority to 10

Clock simulates all of start time and end time of shows. The start times are set.
The end times, how long the parades or shows take are generated by a random time.

The clock will signal when the parade starts, when the parade ends, when the show starts and when the show ends. You must create the correct time sequence of events and have the clock signal them at the appropriate times.

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 closely the story 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 its 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, 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

Email the archive with the specific heading: **CS344-715 Project # Submission: Last Name, First Name** to simina.fluture@gmail.com
You should receive an acknowledgement within 24 hours.