CSCI 340 FALL 2013

THIS PROJECT **MUST BE DONE INDIVIDUALLY**.

PROJECT 1: Due: Nov. 20th

If you have question regarding submission requirements and programming please email to JLAIQC@GMAIL.COM

Through its implementation, this project will familiarize you with the creation and execution of threads, and with the use of the Thread class methods. In order to synchronize the threads you will have to use (when necessary), run( ), start( ), currentThread( ), getName(**), join( ),** yield( ), sleep(time), **isAlive( ),** getPriority( ), setPriority( ), interrupt( ).
In synchronizing threads, do NOT use any semaphores, wait( ), notify( ) or notifyAll( ).

ELLIS ISLAND

Tourists go to Ellis Island to learn about the beginnings of America, and about their ancestors who immigrated to this country.

Every one and a half hours a documentary movie is presented. A movie session contains a presentation period, and the movie. Before the movie session starts, interested visitors **wait** in the lobby (use **busy waiting**). When the session starts, visitors check if there are available seats, and if yes, they take one of the available seats. Checking and updating the number of available seats represents a critical section and it should be done from inside of a **synchronized** method. If there are no free seats, the visitors leave the room.

They will attempt to attend the next coming movie session (use **busy waiting**). They are, a bit, worried that they might not be able to attend any movie session, so they rush and check if there are available seats in the next presentation. In order to do so, they increase their priority (use **getPriority** and **setPriority**). Note: this should happen for a very short time and after the check, the priority is reset back to default. If the next session is already full they give up (use **yield()** to simulate give up) and move on in visiting the museum.

Visitors that attend the movie session will next listen to the speaker's presentation (busy waiting). After the watching the movie they will also move on in visiting the museum.

Before the movie starts, a speaker will give a short introduction to the present audience. When the presentation ends, everybody watches the movie (**sleep** of fixed time).

At the end of the day, when the museum is closing, all visitors will leave. They will do this in decreasing order of their name (id) (this must be implemented using **isAlive()** and **join()**)

The speaker waits for the time to start the presentation (by **sleeping** for a time that is long enough so that it will be interrupted when times comes. (use **interrupt( )** and **isInterrupted()**). At the end of the day the speaker will leave.

In order to keep track of the time, we need an additional thread, named *clock*. The *clock* will signal when a session starts, when a session ends, and when it is the end of the day. (This will be implemented by having the clock **sleep** for fixed interval of time) Between movie sessions we should have a break of 15 minutes.

Initial values: theater capacity: 6
Num_visitors: 15
Number_sessions/day: 2

Using Java programming, synchronize the three types of threads, visitor(s), speaker, clock, in the context of the problem. Closely follow the implementation requirements.

*Use appropriate **System.out.println( )** statements and the **age( )** method to reflect the time of each particular action done by a specific thread. This is necessary for us to observe how the synchronization is working.*