IST 411 – Chapter 32 : Elevator Simulation using Threads

In a medium-sized city somewhere in Pennsylvania stands a building with *eleven(11)* floors, numbered from one on up, and an elevator. The *five(5)* people who work in the building use the elevator to move from one floor to another. At exactly 8 o'clock every weekday morning, these people enter the building and take the elevator to the floors they work on. The elevator is waiting with open doors at 8 o'clock and is big enough to hold all five people at once. From 8 o'clock in the morning until 5 o'clock in the afternoon, the people wander around the building, using the elevator to move from floor to floor as needed.

Your assignment is to write a multi-class multithreaded Java program that simulates the movement of the people in the building and their use of the elevator. Your program should use synchronized methods for the elevator thread and the people threads and include error processing for user input as needed.

When a person wants to use the elevator to go to another floor, the person pushes the elevator call button, waits for the elevator to arrive and open its doors, enters the elevator, pushes the button of the destination floor, and waits for the elevator to arrive at that floor. Because this is an old building, each floor has just a single button to call the elevator, not separate "up"' and "down"' buttons we are more accustomed to seeing.

The elevator continually goes up and down, picking up and dropping off passengers. After unloading, the elevator waits where it is if there are no waiting passengers on other floors; if there are people waiting, the elevator picks them up.

What algorithm does the elevator use when on floor *f* and there are people waiting on floors *x* and *y*? The elevator does not use "nearest floor next"' because that could leave people on the upper or lower floors waiting longer than is fair. Instead, the elevator goes up until it reaches the highest floor needed to pick up or drop off a passenger and then goes down to the lowest floor needed to pick up or drop off a passenger. While going up and down like this, the elevator picks up and drops off passengers along the way. The important feature of the elevator algorithm is that the elevator does not reverse direction until it has gone as far in one direction as needed to pick up and/or drop off passengers.

The elevator takes one second to move from a floor to the next higher or lower floor. If the elevator needs to stop at a particular floor to pick up or discharge passengers, it takes at least one second to do so. After a person disembarks, that person cannot board the elevator again until the elevator leaves and returns to that person's floor **unless** the elevator is idle (no call buttons pushed and no floor buttons pushed) right after that person disembarks.

Your program will have an elevator class from which an elevator object containing the elevator thread is instantiated, and a person class from which person objects containing a person thread each are instantiated. In the elevator class, place the elevator state variables as private data fields. A person object calls a method in the elevator object to indicate the person is waiting on a floor to be picked up. The person thread then waits until picked up by the elevator. When picked up, the person object calls another method to indicate the desired destination floor. The person thread waits until the elevator arrives at the floor. When arriving at a floor, the elevator thread sleeps for one second to let passengers get on and off, then the elevator thread blocks until all passengers have safely (dis)embarked.

Your program should provide a mechanism for each person to call the elevator as well as select a floor to which they would like to travel. In addition, your program should provide feedback to the user to let them know the status of what is happening inside the building (i.e. Person 1 on floor 11 has requested service, Person 2 entering elevator on floor 5, Person 2 exiting elevator on Floor 10, Elevator waiting for passengers, Elevator on floor 1, etc). You may use Java Swing or Java FX for your application and input information from the user via the GUI.

Sample Thread projects have been posted to CANVAS for your reference. These projects do not utilize Thread Pools (ExecutorService interface or the Executors class) that were introduced in Chapter 32 of your textbook. Using Thread Pools in this project is optional as there are a finite number of threads to manage.

The variables and methods defined within your Java classes must **include comments**. Your program should also be **properly formatted**, which includes indentation of code.

Milestones:
(1) Homework Assigned: **Thursday, January 30, 2020**
(2) In-Class Lab: **Thursday, February 6, 2020**
(3) Completed Homework Due: **Tuesday, February 18, 2020 by 11:00pm.** You are to zip your entire NetBeans Project and upload to the HW2_Threads Drop Box.

Good Luck!