**Department of Computer Science.**
**Georgia State University**

**CSc 3410** **J. L. Bhola**

**Fall 2017 – Assign#2**
**Due September 15th, 2017 at 11:p.m. in iCollege.**

## Allocation of points for all assignments:

1. 35% of the points are allocated to Documentation.
2. 65% of the points are allocated to programming, including pre and post conditions.

### Documentation:

1. Explain the purpose of the program as detail as possible - **8%.**
2. Develop a solution for the problem and mention algorithms to be used - **12%.**
3. List data structures to be used in solution. - **5%.**
4. Give a description of how to use the program and expected input/output - **5%.**
5. Explain the purpose of each class you develop in the program. - **5%.**

### Programming:

1. For each method, give the pre and post conditions and invariant, if any - **10%.**
2. Program execution **according to the requirements given** - **50%.**
3. Naming of program as required - **5%.**

### Description of program:

You are to write a program name **elevator.java** that simulates one elevator services in a 12-floor building. **NOTE**: There is only one elevator working this building.

### Here are some requirements for the program:

1. The floors of the building are numbered 1 to 12.
2. At the start of the program, randomly generate 8 of the 12 floors to stop at while going up. These must be placed in an ArrayList which should be sorted in ascending order. Also generate another set of 5 numbers representing floors to stop at when going down and place it in another ArrayList. Sort this ArrayList in descending order.

3. Also at the start of the program the elevator must be located on the 1st floor and the first request is made from the ArrayList which must be a request to go up.

4. From the 1st floor, the elevator can only go in one direction – up. And from the 12th floor the elevator can only go in one direction – down. For the other floors, the elevator can go in either direction i.e. if it is going down when you get on and you want to go up and the down ArrayList is empty, then the elevator could start going up from that floor. The same will be true if the elevator is going up when you get on, and you want to go down and the up ArrayList is empty, then it can start going down from that floor.

5. **The direction in which the elevator is going must be known at all times and the floor to which it is going.** If the elevator is going up, the **request** could be anywhere from floor 2 to 12. If it is going down, it can be from 11 to 1, **with 1 being the default**.

6. The location of the elevator must be known at all times. The program should provide output regarding the current floor, whether the elevator is going up or down, which floor it is starting from and which floor it is going to, as the following output will show:

7. Each time you stop at a floor, you must stay there for 3 seconds (allowing the people to either enter the elevator or leave the elevator). Display this time in a counting format as well as an appropriate message. The elevator must take 2 seconds to move from one floor to another. No need to display this time.

Up_ArrayList

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 7 | 9 | 3 | 6 | 10 | 11 | Randomly generated ArrayList |
| 2 | 3 | 4 | 6 | 7 | 8 | 10 | 11 | Sorted ArrayList |

Down_ ArrayList

| | | | | | |
|---|---|---|---|---|---|
| 3 | 5 | 8 | 9 | 12 | Randomly generated ArrayList |
| 12 | 9 | 8 | 5 | 3 | Sorted ArrayList |

```
Starting at floor 1
   Going up: now at floor 2
Stopping at floor 2 for 3 seconds → 1, 2, 3

Starting at floor 2
   Going up: now at floor 3
Stopping at floor 3 for 3 seconds → 1, 2, 3
```

```
Starting at floor 3
   Going up: now at floor 4
Stopping at floor 4 for 3 seconds → 1, 2, 3

Starting at floor 4
   Going up: now at floor 5
   Going up: now at floor 6
 Stopping at floor 6 for 3 seconds → 1, 2, 3

 Starting at floor 6
    Going up: now at floor 7
Stopping at floor 7 for 3 seconds → 1, 2, 3

 Starting at floor 7
    Going up: now at floor 8
 Stopping at floor 8 for 3 seconds → 1, 2, 3

 Starting at floor 8
    Going up: now at floor 8
    Going up: now at floor 9
    Going up: now at floor 10
 Stopping at floor 10 for 3 seconds → 1, 2, 3

 Starting at floor 10
    Going up: now at floor 11
 Stopping at floor 11 for 3 seconds → 1, 2, 3

 Starting at floor 11
    Going up: now at floor 12
 Stopping at floor 12 for 3 seconds → 1, 2, 3

 Starting at floor 12
    Going down: now at floor 11
    Going down: now at floor10
    Going down: now at floor 9
 Stopping at floor 9 for 3 seconds → 1, 2, 3

 Starting at floor 9
    Going down: now at floor 8
 Stopping at floor 8 for 3 seconds → 1, 2, 3

 Starting at floor 8
    Going down: now at floor 7
    Going down: now at floor 6
    Going down: now at floor 5
```

```
Stopping at floor 5 for 3 seconds → 1,2, 3

Starting at floor 5
    Going down: now at floor 4
    Going down: now at floor 3
Stopping at floor 3 for 3 seconds → 1,2, 3

Starting at floor 3
    Going down: now at floor 2
    Going down: now at floor 1
```

Do you want to run the elevator again? "***Y or y***" continue. "***N or n***" stop.

8. It must also display the request structures – requests for going up and down as shown above.

9. Request for the elevator must be randomly generated and placed in an up ArrayList and a down ArrayList (**NO DUPLICATE VALUES ALLOWED**). So the ArrayList **size** must be 12. Carry out the generation at the beginning of the program.

10. Every time the elevator stops at a floor, it should check the relevant request ArrayList to see which floor it should stop at next. It should then delete that floor from the request ArrayList when it stops at that floor.

11. Any other functions that you could implement that would make the elevator operation as realistic as possible.

11. Give the user the option to run the program again and again, etc.

## What to turn in:

Turn **all of the .java and .class/.jar files** at the web site iCollege(D2L - https://gastate.view.usg.edu/) in the directory/folder A#2 into your respective CRN section NO LATER THAN **11:00 p.m**. on the due date of **September, 15th , 2017.**
MAKE SURE YOUR NAME IS ON THE FILE.