



Assignment 3

The purpose of this assignment is to familiarize you with loops and conditions in java, as well as to practice writing an algorithm to solve a problem.

Refer to our Programming [Assignments FAQ](#) for instructions on how to install VSCode, how to use the command line and how to submit your assignments.

- See this [video](#) on how to import the files into VS Code and how to submit into Autolab.

Observe the following rules:

- **DO NOT** add any import statements
- **DO NOT** add the project statement
- **DO NOT** change the class name
- **DO NOT** change the headers of ANY of the given methods
- **DO NOT** add any new class fields
- **DO NOT** use static variables
- **DO NOT** use System.exit()
- **ONLY** print the result as specified by each problem.
Observe the examples' output, display only what the problem is asking for.
- **DO NOT** print other messages, follow the examples for each problem.

1. Floor Is Lava – 20 points

Overview

Imagine that the floor has safe spaces (stone) to step on and dangerous spaces (lava) that if you step on you could get hurt. But this floor is tricky as the safe and dangerous spaces switch (stones become lava and lava becomes stone).

In this assignment, you will write a program that simulates a sequence of safe spaces (stones). Your task is to print:

1. all **safe spaces (even numbers)** in ascending order first.
At this moment lava and stones “switch” (stones become odd, lava become even).
2. then all **safe spaces (odd numbers now)** in decreasing order.

Task

- We provide a zip file (find it in Autolab) that contains **FloorIsLava.java**.
- UPDATE and SUBMIT the corresponding file.

Programming

Write a program, FloorIsLava.java, that reads 1 integer argument from the command line.

- Expect the input to be an integer value greater than or equals 0.
- If the input is 0, nothing is expected to be printed.

The program is expected to print all even numbers up to and including the input value (if even). Once all the even numbers have been printed, it prints all odd numbers in descending order (starting with the input value).

Expected Output:

For FloorIsLava 8:

2 4 6 8 7 5 3 1

For FloorIsLava 13:

2 4 6 8 10 12 13 11 9 7 5 3 1

For FloorIsLava 0:
(no output)

Executing and Debugging

First navigate to FloorIsLava directory/folder

- to compile: `javac FloorIsLava.java`
- to execute: `java FloorIsLava [any number]`

Assignment created by Shane Haughton and Maaz Mansuri

2. Elevator – 20 points

Overview

Imagine that a building with several floors has two elevators servicing floor requests (people pushing the elevator button).

In this assignment, you will write a program that simulates which elevator services each requested floor.

Task

- We provide a zip file (find it in Autolab) that contains **Elevator.java**.
- UPDATE and SUBMIT the corresponding file.

Programming

Write a program, Elevator.java, that reads 4 integer arguments from the command line:

- the number of floors in the building at `args[0]`
- the floor requests at `args[1]`
- number of restricted floors at `args[2]`
- and an optional passcode at `args[3]`

The program is expected to print which elevator services each request and which are the floors at which the elevators stopped at.

- The number of floors (`args[0]`) will always be greater than or equal to any value present in the floor requests (`args[1]`).
- There will always be 2 elevators that start at floor 1.
- The elevator that services a request will be based on which elevator is closer to the next floor in the floor requests.
- The elevators traverse the floors with Elevator 1 taking priority over Elevator 2.
 - If the floor the elevators on are equal to each other, Elevator 1 will move to the requested floor.
- Note that the floor requests has several floors and they are in reversed order. For example, for a building with 5 floors the requests could be 134.
 - The first request is floor 4. Since both elevators start at floor 1, and both elevators are in floor 1, elevator 1 will service this request.
 - The second request is floor 3. At this point elevator 1 is at floor 4 and elevator 2 is at floor 1, so elevator 1 is closer and will service this request.
 - The third request is floor 1. At this point elevator 1 is at floor 3 and elevator 2 is at floor 1, so elevator 2 is closer and will service this request.
 - HINT: note that you can use the modulus (%) operator to extract the rightmost digit, which is the first request. Then remove that request from the number by dividing the floor requests by 10 (it removes the last digit).
- The number of restricted floors, say 'n', represents the idea that the top 'n' floors are restricted access.

The password argument (`args[3]`) is optional, meaning that the argument **is only read** if someone is trying to access a restricted floor. Access to the floor is granted if:

- The password modulus the number of floors is equal to the floor being requested, OR

- The password modulus the number of floors is equal to zero AND requested floor is the topmost floor.

Assumptions

- There will always be 2 elevators that traverse the floors with Elevator 1 taking priority over Elevator 2.
 - If the floor the elevators on are equal to each other, Elevator 1 will move to the requested floor.
- The number of floors (`args[0]`) will always be less than or equal to any value present in the queue of floor requests (`args[1]`)

Output

The output is expected to be the elevator number (1 or 2) followed by the requested floor number. If there are any restricted floors in the requests, your output must include the status: "Granted" if the passcode is valid, "Denied" otherwise.

Example 1: Executing the command `java Elevator 5 5134 1 25` produces the following output.

```
1 4
1 3
2 1
1 5
Granted
```

Here is a more detailed breakdown of the above example:

```
java Elevator 5 5134 1 25
```

Number of
floors in
Elevators (i.e. 1
through 5)

The top n floors
are restricted
where this param
= n (i.e floor 5)

String of floor
requests where each
digit represents a
floor #

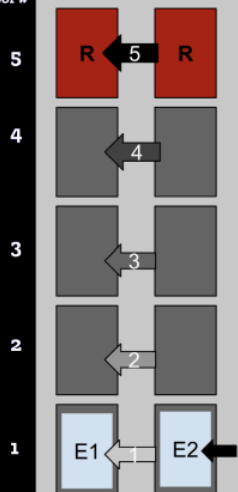
Optional param
password to access a
restricted floor- code
is shown below



**Password % number of
floors must equal the
requested floor number**

Note: you must modify this to
account for the edge case of
accessing the last floor like in
this example.

Floor #



Floor Request Queue

5134



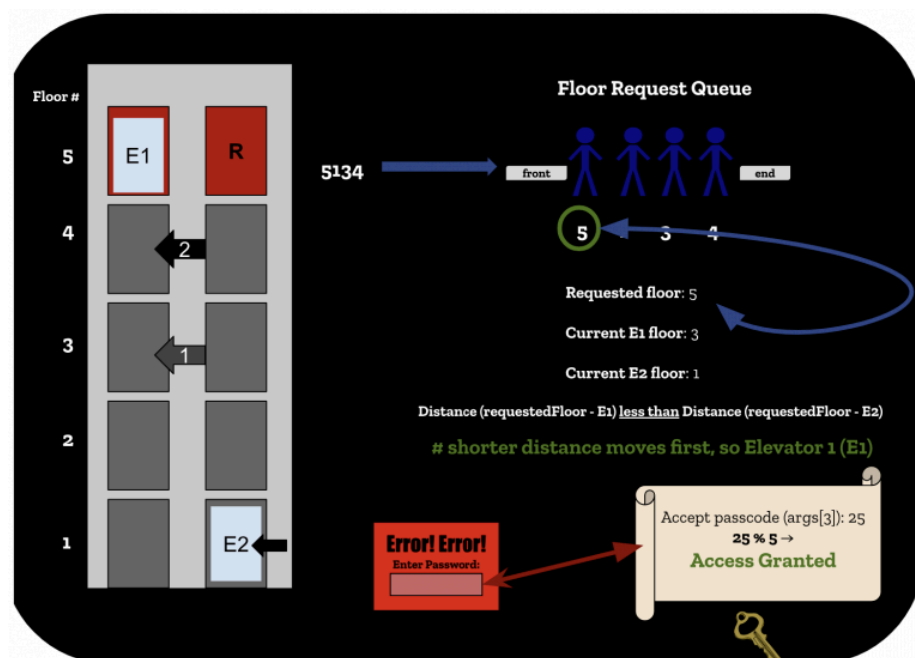
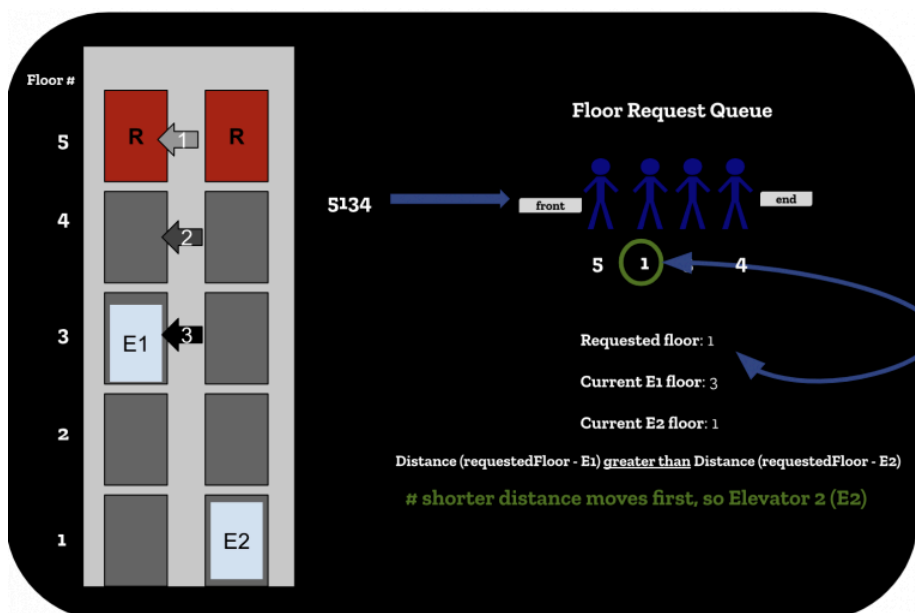
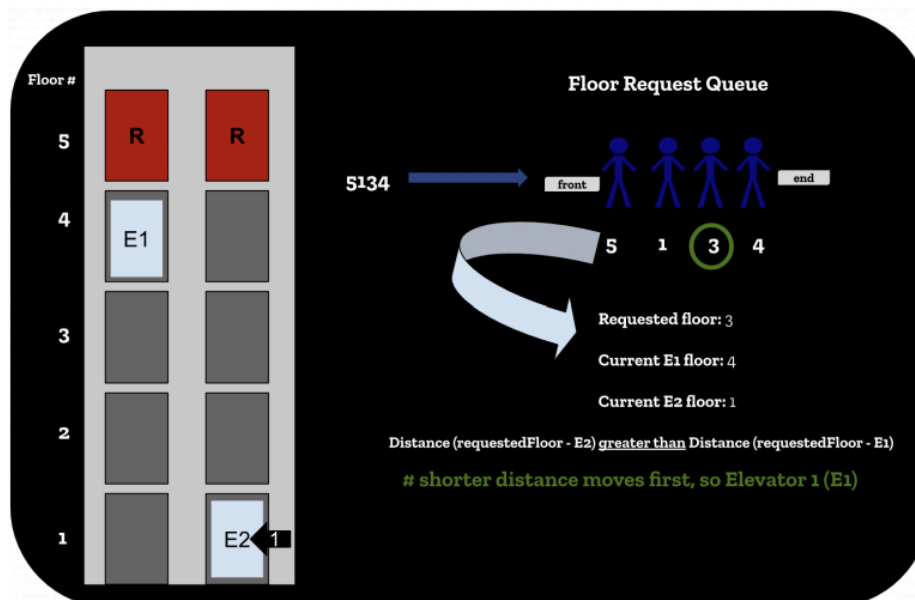
Requested floor: 4

Current E1 floor: 1

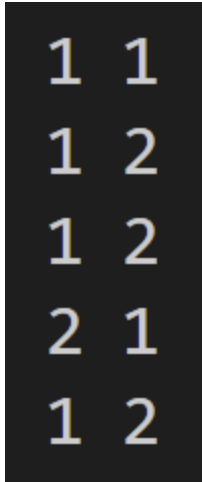
Current E2 floor: 1

Distance (requestedFloor - E1) equals Distance (requestedFloor - E2)

in this case E1 moves first



Example 2: Executing the command `java Elevator 2 21221 0` produces the following output.



1	1
1	2
1	2
2	1
1	2

Executing and Debugging

First navigate to **Elevator** directory/folder

- to compile: `javac Elevator.java`
- to execute: `java Elevator args[0] args[1] args[2] args[3]`
 - NOTE: `args[0]`, `args[1]`, `args[2]`, `args[3]` must be replaced by some number denoting the # of floors, a queue of floor requests, # of restricted floors, and an optional password, respectively.

Assignment created by Vidushi Jindal and Pooja Kedia

VSCode Extensions

You can install VSCode extension packs for Java. Take a look at [this tutorial](#). We suggest:

- [Extension Pack for Java](#)
- [Project Manager for Java](#)
- [Debugger for Java](#)

Importing VSCode Project

1. Download FloorIsLava.zip from [Autolab Attachments](#).
 2. Unzip the file by double clicking.
 3. Open VSCode
- Import the folder to a workspace through File > Open Folder

Before submission

Collaboration policy. Read our collaboration policy [here](#).

Submitting the assignment. Submit SolutionFile.java separately via the web submission system called Autolab. To do this, click the Assignments link from the course website; click the Submit link for that assignment.

Getting help

If anything is unclear, don't hesitate to drop by office hours or post a question on Piazza.

- Find instructors office hours [here](#)
- Find tutors office hours on Canvas -> Tutoring
- Find head TAs office hours [here](#)
- In addition to office hours we have the [Coding and Social Lounge \(CSL\)](#) , a community space staffed with lab assistants which are undergraduate students further along the CS major to answer questions.