Quarter Programming Project

Project Overview

The quarter programming project is to design and implement an object-oriented elevator simulator. This simulator application will model a building, its floors, its elevators, its call boxes, controllers, its people, etc. in order to perform a variety of analyses that will help determine the optimal elevator configuration for a given building. Additionally this application can predict the effects of changing the "default" floor for one or more elevators, and can predict the expected effect of taking an elevator down for repairs on the building's population.

Development Approach

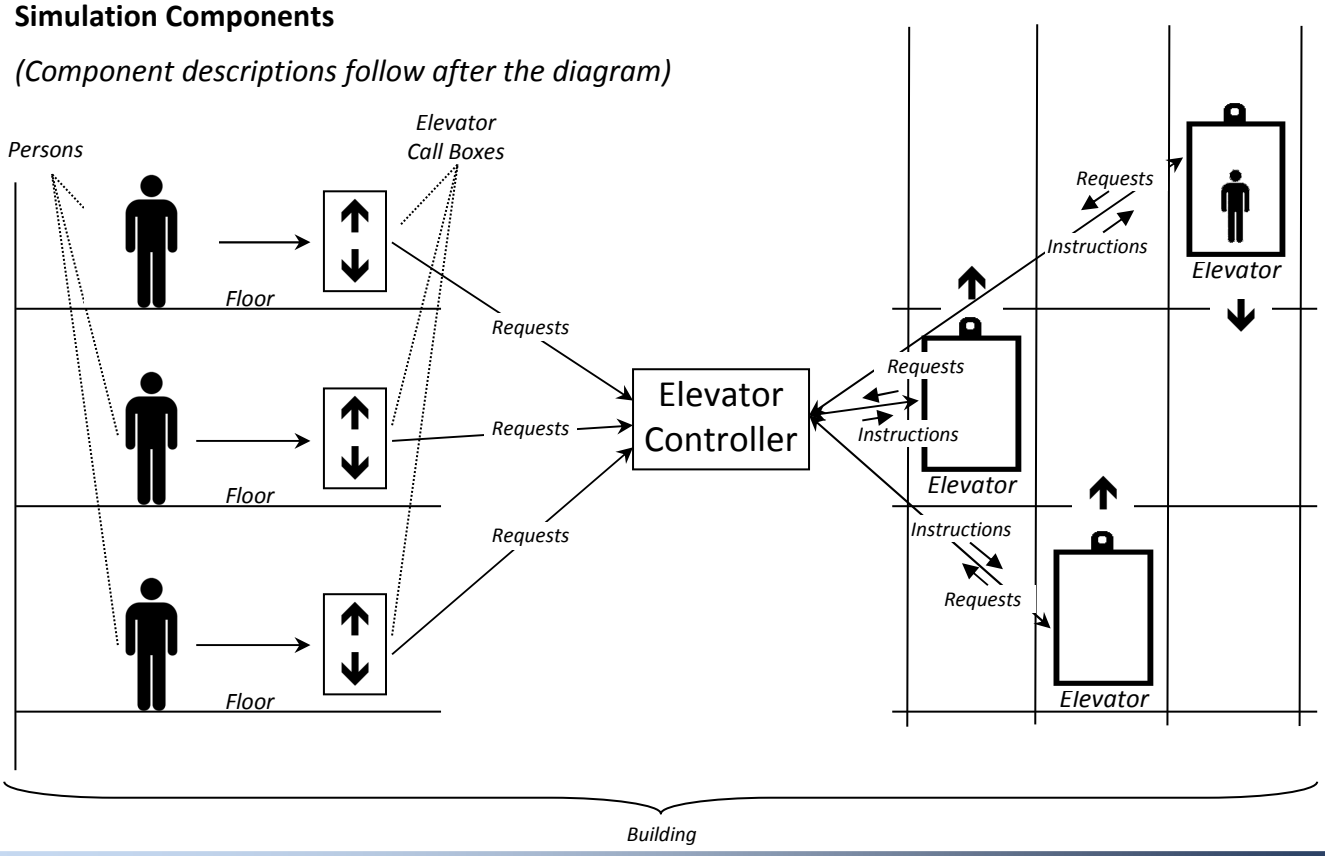
Students may elect to work alone or as a part of a team of 2. If working alone, the student is still responsible for all aspects of the full project. If working as team the work must be divided between the 2 team members as equally as possible and documentation must be provided indicating who did what.

Student's working as a team *must* email me their team member names within the first week of project work (on or before April 17th, 5:45 pm). *Anyone not mentioned in a team-related email by that time will be considered working alone.*

Project work will consist of developing an optimal object oriented design and implementation that satisfies the needs of the application. Certain aspects of the project must be documented using Javadoc, and unit tested using JUnit. Individual deliverables are described later in this document.

Simulation Components

(Component descriptions follow after the diagram)



Simulation Component Descriptions

- Person – Person objects are created and initially placed on a floor of the building. They will call the elevator (by pressing the Up or Down call button), enter the elevator when it arrives, they will select a destination floor, they will ride the elevator to their destination, then exit the elevator at their desired floor.
- Elevator – Elevators travel up and down to the various floors of the building. Elevators have a button panel that allows riders (Persons) to select a destination floor. *Elevators have a maximum passenger count (i.e., 10 people), they have a time in milliseconds per floor value (i.e., the speed of the elevator – for example, 1000 ms per floor), and a door-operation-time in milliseconds (i.e., how long the elevator door takes to open at a floor, remain open for Persons to enter & exit, and to close). Other constants such as these will be needed.*
- Elevator Call Box – Each building floor has an elevator call box – this has an up and down button that is used to call an elevator to their floor.
- Floor – Each floor of the building holds a number of Person objects. Some will be waiting for an elevator (after pressing a button on the elevator call box). Others will remain on that after exiting an elevator. Each floor has an elevator call box where an elevator can be requested (called) to go up or down.
- Building – Each building owns a certain number of floors and a certain number of elevators. A building also owns an elevator controller that coordinates the actions of all elevators.
- Elevator Controller – The elevator controller determines which elevator should respond to a request made at an elevator call box, and when. The elevator controller will instruct individual elevators to go to certain floors to respond to elevator requests made by Person objects.

Algorithms

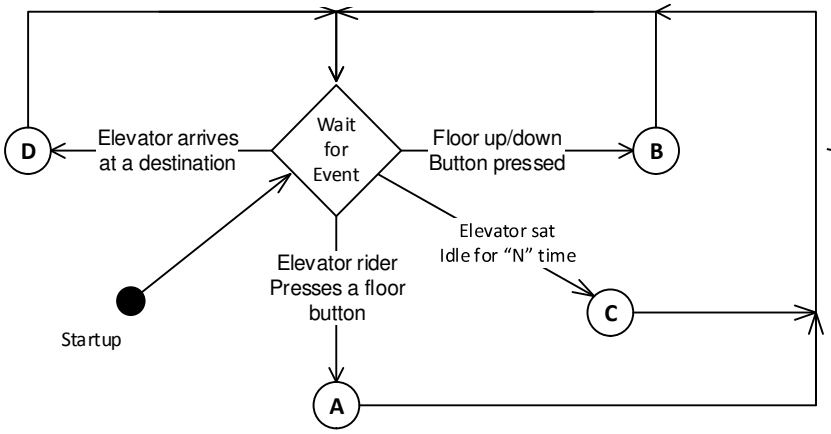
The subsequent pages contain UML activity diagrams that highlight the elevator behaviors (algorithms). These diagrams represent the "default" set of behaviors that *must* be incorporated into all student applications. In object oriented fashion, several algorithms in this simulation should be designed and implemented as changeable (via the Strategy Design Pattern) to allow extendable application behavior changes:

1. Selection of the specific elevator to respond to an Elevator Up/Down Call. Activity Diagram Name: "Elevator Call Button (Up/Down) is Pressed" (B)
2. Selection and assignment of "Pending" elevator requests to elevators. Activity Diagram Name: "Select from Pending Requests" (E)

In addition to implementing the two algorithms listed above as the Activity Diagrams specify, each project must also provide a *second* implementation of the two algorithms that improves either the elevator wait times (without increasing the ride-time), or improves the elevator ride times (without increasing the wait-time). Elevator wait times and ride times are discussed later in this document.

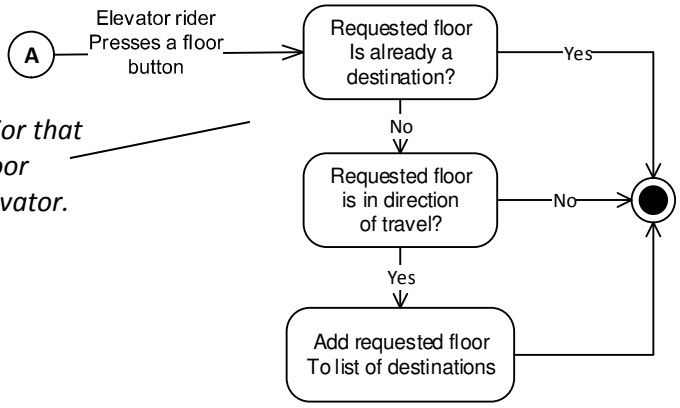
Behavioral Diagrams

High-Level Behavioral Flow



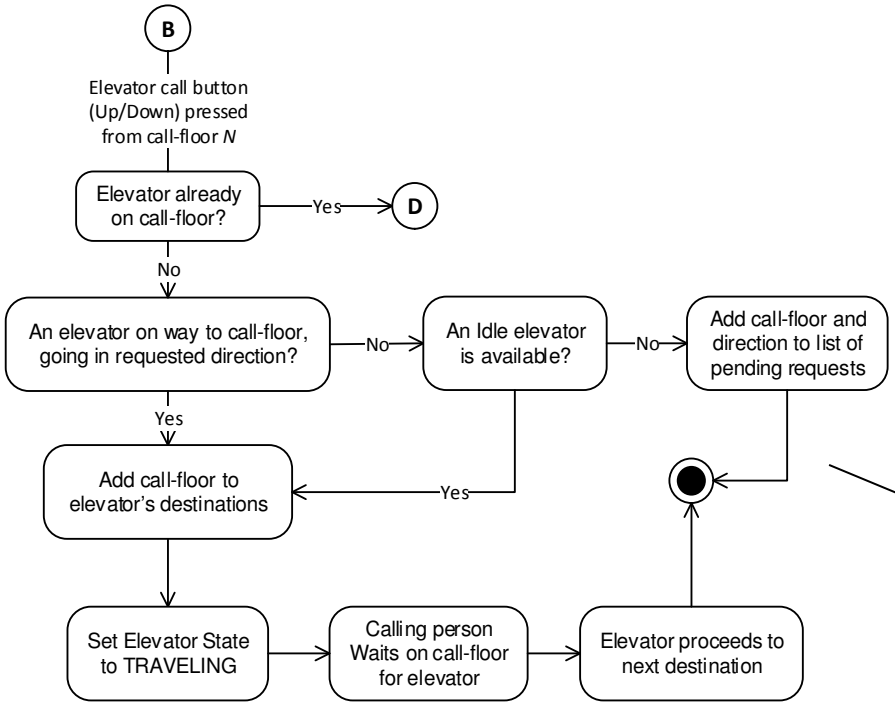
This diagram shows the basic behaviors that the elevator and elevator system can perform.
The key actions are a person pressing an Up/Down elevator call button, a person pressing a floor number button from inside the elevator, the arrival of the elevator at a destination floor, and an idle elevator timing out.

Elevator Rider Presses a Floor Button



This diagram (A) shows the behavior that occurs when a person presses a floor number button from inside the elevator.

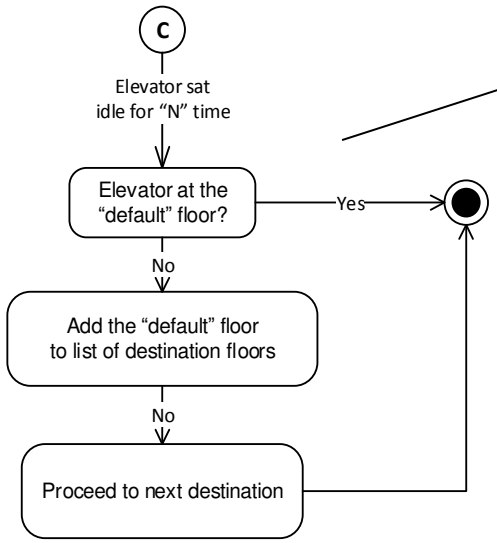
Elevator Call Button (Up/Down) is Pressed



This diagram (B) shows the behavior that occurs when a person presses an Up or Down elevator call button from a buildina floor.

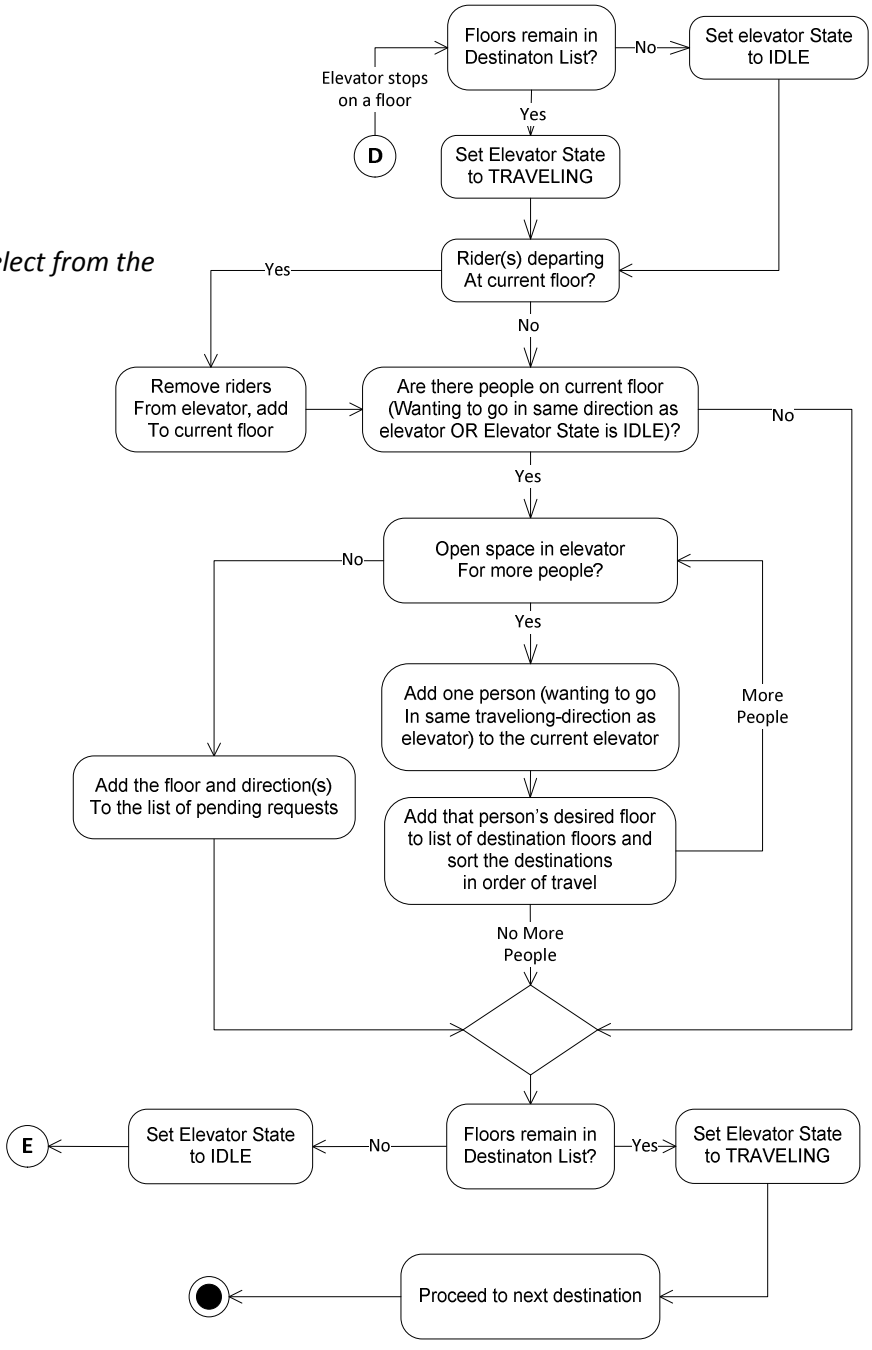
Elevator Remains Idle for *"N"* Time

This diagram (C) shows the behavior that occurs when an elevator remains idle (with no riders and no Up/Down calls pressed) for "N" time – the elevator returns to the "default" floor.



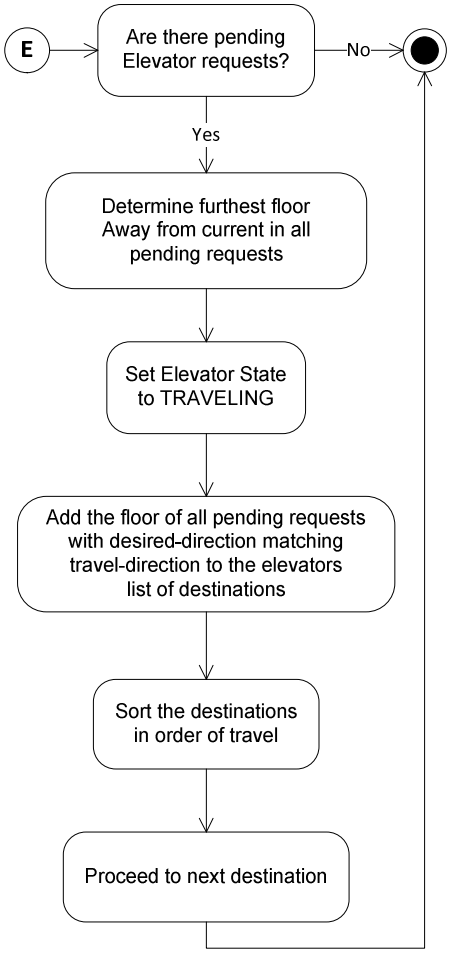
This diagram (D) shows the behavior that occurs when an elevator stops at a destination floor (a floor requested by a rider –or– a floor call made via the Up/Down elevator call buttons.

Elevator Stops at a Floor



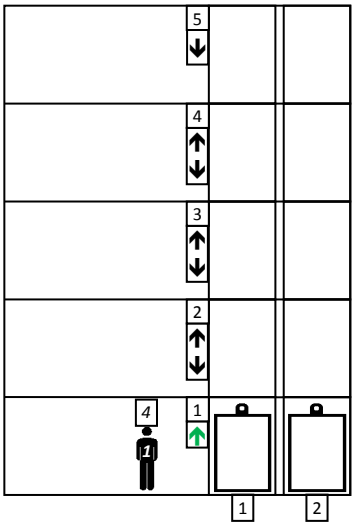
This diagram (E) shows how to select from the set of pending elevator requests

Select from Pending Requests

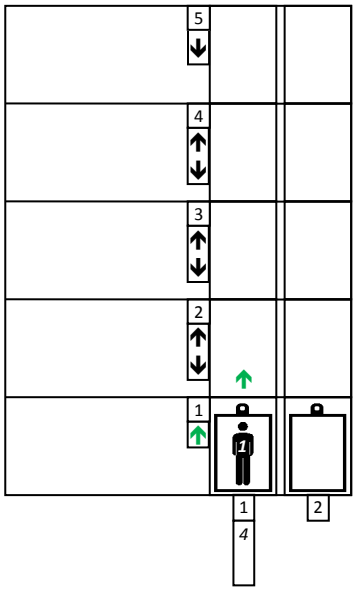


Expected Elevator Behavior – Sample Scenario

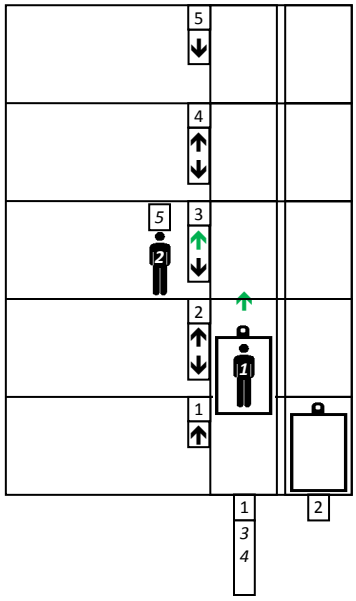
1) Person 1 calls the elevator from Floor 1 to go Up. Person 1 wants to go to Floor 4.



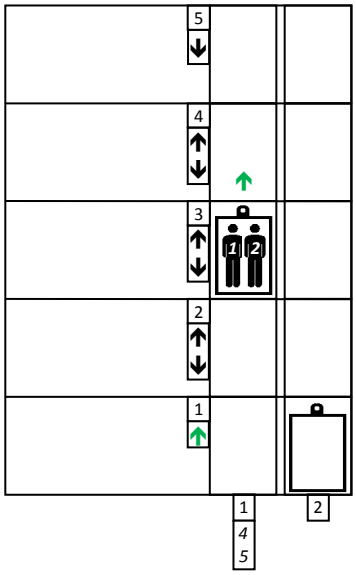
2) All elevators are available and are on Floor 1, so Elevator 1 is randomly selected. The door opens, Person 1 enters. Person 1 presses the Floor 4 button. The door closes. Elevator 1 is going up to Floor 4.



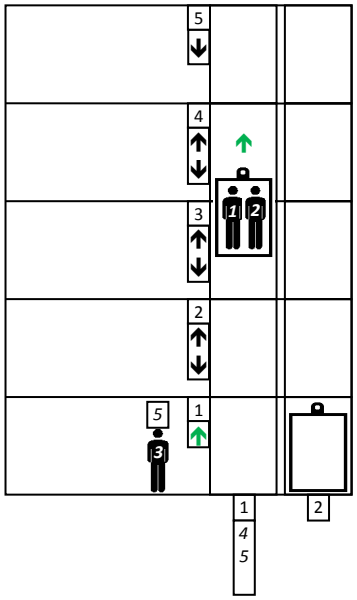
3) While traveling to Floor 4, Person 2 on Floor 3 presses the Up button, wanting to go to Floor 5. Since Floor 3 is in the current direction of travel of Elevator 1, *and* the call button pressed (Up) matches the direction of travel (Up), Floor 3 is added to the list of destinations.



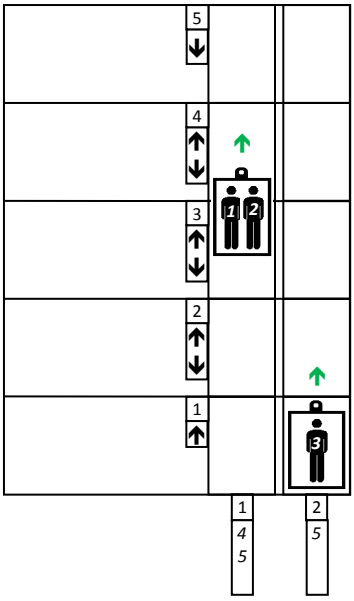
4) Elevator 1 reaches its first destination at Floor 3. The elevator stops and the door opens. Person 2 enters the elevator and presses the Floor 5 button. Floor 5 is added to the list of destinations.



5) Person 3 on Floor 1 presses the Up button, wanting to go to Floor 5.

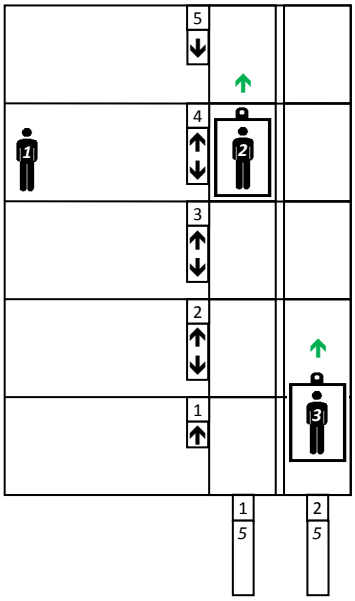


6) Elevator 1 is above Floor 1 heading Up - so that elevator is *not* selected. Elevator 2 is idle at Floor 1 so that elevator *is* selected. The door opens, Person 3 enters. Person 3 presses the Floor 5 button. Elevator 2 is going up to Floor 5.

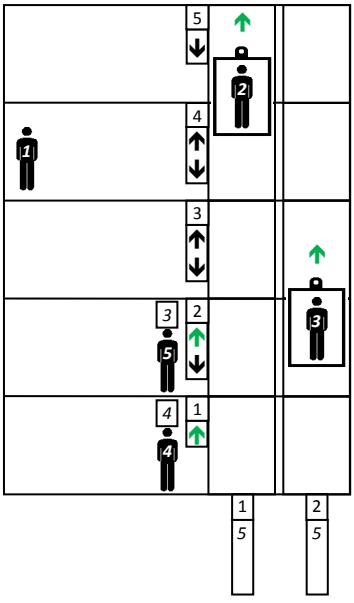


Continued...

7) Meanwhile, elevator 1 arrives at its next destination, Floor 4. The door opens, and Person 1 exits onto Floor 4, and the door closes.

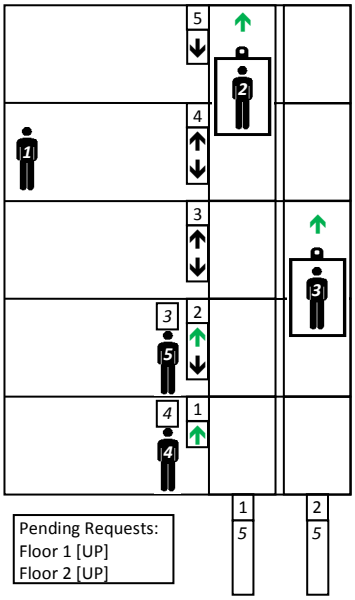


8) Now, Person 4 on Floor 1 presses the Up button, wanting to go to Floor 4. *Additionally*, Person 5 on Floor 2 presses the Up button, wanting to go to Floor 3.

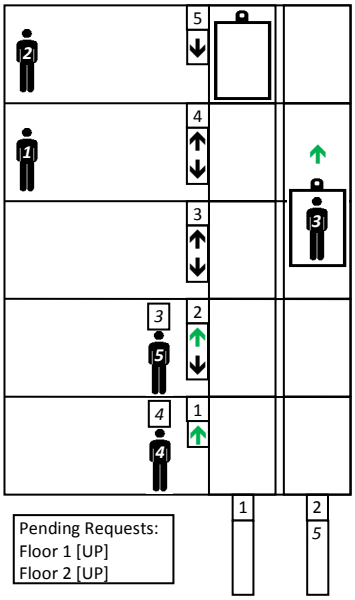


9) Elevator 1 & 2 are both heading up and have gone past Floor 1 and Floor 2, so the elevators both must finish their current travel *before* returning to Floor 1.

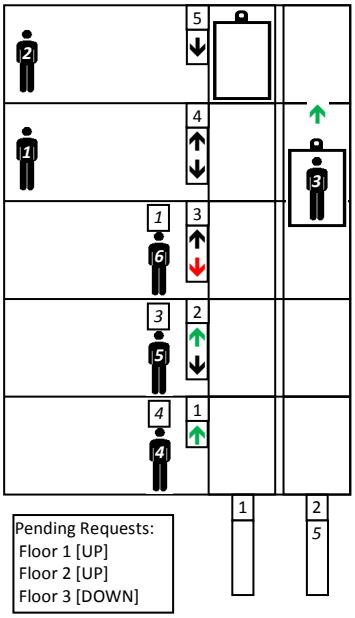
Since there are no other elevators, the requests are considered "pending" and are added to the list of pending floor requests. These will be handled by the next elevator to complete its destination list.



10) Elevator 1 arrives at its last destination, Floor 5. The door opens, and Person 2 exits onto Floor 5, and the door closes.



11) Meanwhile, Person 6 on Floor 3 presses the Down button, wanting to go to Floor 1. There are currently no elevators heading down so this request is considered "pending" and is added to the list of pending floor requests.



12) Elevator 1 now has no more destinations so that can be used to process Pending Requests.

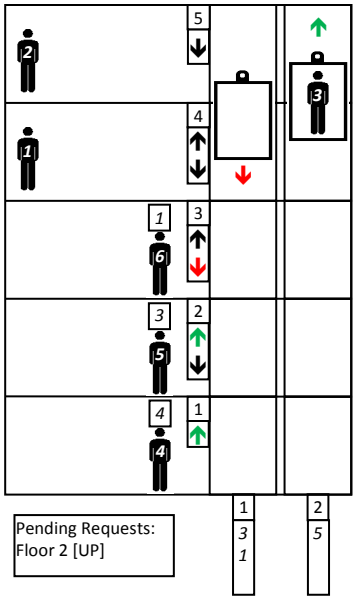
The furthest pending request floor is Floor 1 (UP). Add Floor 1 (to go UP) to Elevator 1's destination list. The direction of travel is Down.

Also add any other requests that can be satisfied going in the direction of travel (down) from Floor 5 to Floor 1. This adds the Pending Request to go from Floor 3 Down.

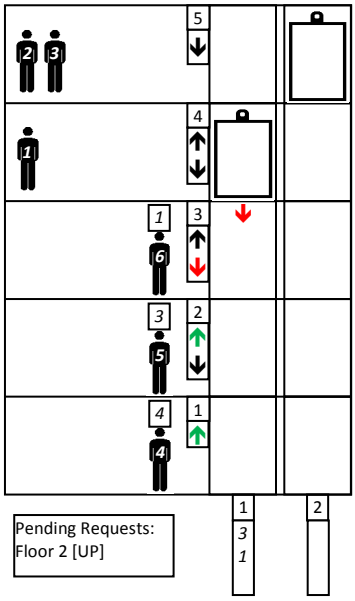
The Pending Request to go from Floor 2 Up remains pending.

Continued…

Results after Step 12:



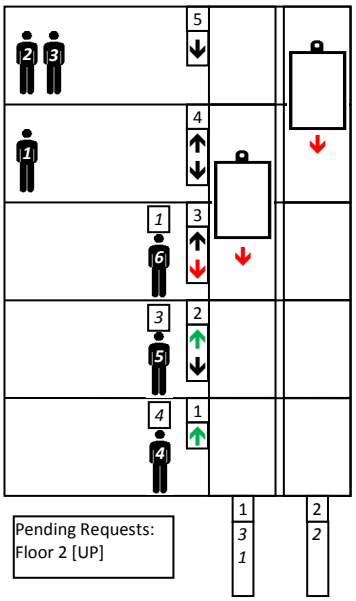
13) Elevator 2 arrives at its last destination, Floor 5. The door opens, and Person 3 exits onto Floor 5, and the door closes.



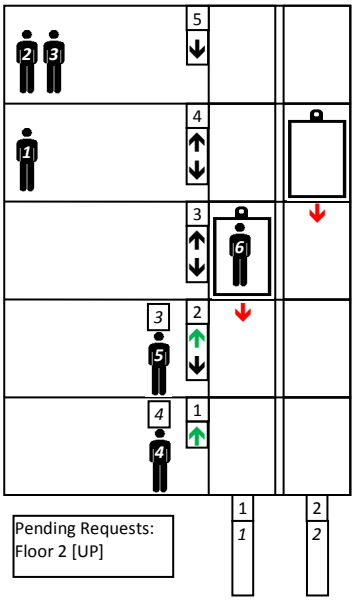
14) Elevator 2 now has no more destinations so that can be used to process Pending Requests.

The furthest pending request floor is Floor 2 (UP). Add Floor 2 (to go UP) to Elevator 1's destination list. The direction of travel is Down.

Add any other requests that can be satisfied going Down from Floor 5 to Floor 1. Since there are no other pending request, there is nothing else to add.

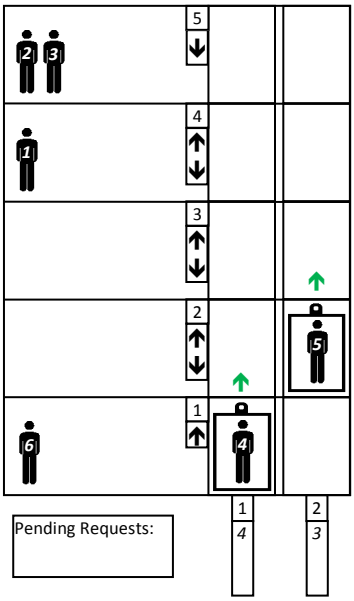


15) Elevator 1 arrives at its next destination, Floor 3. The door opens, Person 6 enters. Person 6 presses the Floor 1 button. Elevator 1 is going to Floor 1.



16) Then elevator 1 arrives at its last destination, Floor 1. The door opens, Person 6 exits, and then Person 4 enters. Person 4 presses the Floor 4 button. The door closes. Elevator 1 is going up to Floor 4.

At the same time, Elevator 2 arrives at its next destination, Floor 2. The door opens, Person 5 enters. Person 5 presses the Floor 3 button. The door closes. Elevator 2 is going up to Floor 3.

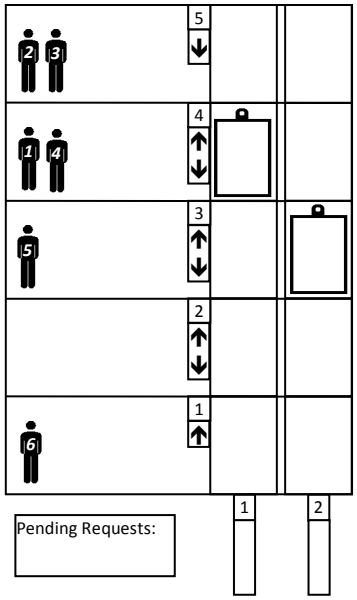


17) Next Elevator 2 arrives at its last destination, Floor 3. The door opens, Person 5 exits, the door closes.

Then Elevator 1 arrives at its last destination, Floor 4. The door opens, Person 4 exits, the door closes.

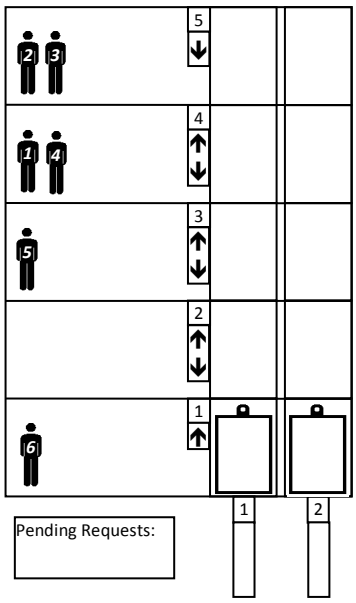
Continued…

Results after Step 17:



18) There are no current requests so both elevators remain idle for a period of time at their current floors.

After a period of time with no requests, the elevators will move to their default destination floors (Floor 1 in this case).



End of Example

Simulation Inputs

The following must be loaded into the simulation as input. The exact format of these specifications is up to you but it should be a well-known format such as XML.

1. Simulation duration time in minutes (15 minutes, 45 minutes, 90 minutes, etc.).
2. Time-scale factor (1:1 = one simulation second is one real second, 5:1 = one simulation second is five real seconds, 10:1 = one simulation second is ten real seconds, etc.)
3. Number of Floors in Building- Example: 24, 5, 7, etc.
4. Number of Elevators in Building – Example: 4, 8, 1, etc.
5. Max Persons Per Elevator – Example 12, 8, 9, etc.
6. Time (in milliseconds) per floor value which represents the speed of the elevator – Example: 1000 ms per floor.
7. Door-operation-time (in milliseconds) which is how long the elevator door takes to open at a floor, remain open for Persons to enter & exit, and to close – Example: 2200 ms, 3000 ms, etc.
8. Default Elevator Floor:

- For All Elevators – Example: All elevators default to floor 1, or floor 12, etc.

– OR –

- For Each Elevator – Example: Elevator 1 Default: Floor 1, Elevator 2 Default: Floor 14, etc.

9. Number of Persons Requesting an Elevator, by Time:
   - Number of Persons requesting an elevator per time (min) – Example: 2 per min, 8 per min, etc.

10. Statistical specification of Floors for Start and Destination Selection:
    - Start and Destination Floor Selection Percentages – Example Data:
      - Floor 1: 50%
      - Floor 2: 5%
      - Floor 3: 8%
      - …
      - Floor N: 14%
      - Total: 100%
    - Start and Destination Floor Selection Percentages – Example usage:
      - Example Specified Percentages:
        - Floor 1: 50% 50%
        - Floor 2: 10% 60%
        - Floor 3: 15% 75%
        - Floor 4: 8% 83%
        - Floor 5: 17% 100%
      - Random selection for **Start Floor** = 0.42 = 42% = **Floor 1**
      - Remaining SCALED Percentages (with Start floor removed):
        - *Floor 1: 0% (Selected Start Floor)*
        - Floor 2: 20% 20%
        - Floor 3: 30% 50%
        - Floor 4: 16% 66%
        - Floor 5: 34% 100%
      - Random selection for **Destination Floor** = 0.63 = 63% = **Floor 4**

Simulation Outputs

1. Timed "Narrative" Description of Activity

(Example)

14:00:00	Person 1 Presses Up Button on Floor 1
14:00:01	Elevator 1 is already on Floor 1 – Doors Open
14:00:03	Person 1 enters Elevator 1 (Elevator 1: Person 1)
14:00:04	Person 1 presses Floor 8 button
14:00:05	Elevator 1 Doors Close
14:00:06	Elevator 1 begins travel to Floor 8 from Floor 1
14:00:09	Person 2 presses Down button on Floor 5
14:00:09	Elevator 2 begins travel to Floor 5 from Floor 7
14:00:11	Person 3 presses Up button on Floor 7
14:00:12	Elevator 1 stops at Floor 7 – Doors Open
14:00:12	Elevator 2 arrives at Floor 5
14:00:13	Elevator 2 Doors Open
14:00:14	Person 3 enters Elevator 1 (Elevator 1: Person 1, Person 3)
14:00:14	Person 2 enters Elevator 2 (Elevator 2: Person 2)

14:00:15 Person 3 presses Floor 8 button
14:00:15 Person 2 presses Floor 1 button
14:00:17 Elevator 1 Doors Close
14:00:17 Elevator 2 Doors Close
14:00:18 Elevator 1 begins travel to Floor 8
14:00:18 Elevator 2 begins travel to Floor 1
14:01:20 Elevator 1 arrives at Floor 8
14:01:22 Elevator 1 Doors Open
14:01:24 Person 1 and Person 3 exit Elevator 1 (Elevator 1: Empty)
Etc…..

2. Time Tables

a) Wait Time for Elevator by Floor (in seconds), default floor is Floor 1

Floor	Average Wait Time	Min Wait Time	Max Wait Time
Floor 1	12 seconds	4 seconds	48 seconds
Floor 2	18 seconds	7 seconds	38 seconds
Floor 3	16 seconds	3 seconds	22 seconds
…			
Floor N	24 seconds	1 second	61 seconds

b) Floor to Floor Average Ride Time (in seconds), default floor is Floor 1

		To Floor (seconds)							
From Floor (seconds)	**Floor**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
	1	X	2	4	7	12	14	16	24
	2	2	X	2	5	6	15	16	19
	3	4	2	X	2	6	7	14	17
	4	8	5	2	X	2	5	8	13
	5	10	9	5	2	X	2	7	9
	6	13	12	10	6	2	X	2	5
	7	15	15	11	9	5	2	X	2
	8	18	16	14	13	8	6	2	x

c) Wait Time by Person (in seconds), default floor is Floor 1

Person	**Wait Time**	**Start Floor**	**Destination Floor**	**Ride Time**
Person 1	12 seconds	1	4	14 seconds
Person 2	23 seconds	8	2	36 seconds
Person 3	5 seconds	3	4	8 seconds
…	…	…	…	…
Person N	17 seconds	6	1	19 seconds

Simulation Driver Process

- While duration has not expired:
  1. Generate "n" Persons per minute according to input specifications
  2. For that Person, select the start and destination floor according to input specifications
  3. Add the Person to the Start Floor, select Up/Down on that floor's Elevator Call Box depending upon the direction to the destination
  4. Continue "While" loop

Exceptions & Exception Handling

It is important to note that any method that accepts parameters should validate the incoming parameters before using them. If bad or unusable parameter data is detected, an appropriate exception should be thrown. You should create your own exception classes to handle all such cases. Remember to only "catch" an exception where the problem encountered can be "fixed".

Remember that if you decide to propagate any exceptions from methods that are declared in an interface, the interface declaration of that method must also indicate that the same exceptions are thrown.

Development Phases

- Design & Development Plan (4/10 – **4/24**)
  - Submit a UML Class diagram showing the classes, interfaces & relationships that you have designed and plan to implement. Classes & interfaces should show all relevant domain data and behaviors where applicable.
  - Submit a development schedule indicating what you plan to do/submit by when based upon the milestone dates below.
  - Formats: Visio, PDF

- Initial Implementation (Elevator functionality) (4/24 – **5/8**)
  - Submit implementation of working Elevator code (standalone). This will consist of an initial implementation containing elevators that take floor number requests and respond to "commands" from the elevator controller. Further requirement and submission details will be posted closer to the start date of 4/24.
  - Code must be Javadoc'd and JUnit tested.

- 2nd Implementation Phase (5/8 – **5/22**)
  - Submit implementation of complete working code that includes working Person, Elevator, Elevator Call Box, Elevator Controller, Floor & Building code (results reporting not

required in this submission). Further requirement and submission details will be posted closer to the start date 5/8.
    - Code must be Javadoc'd and JUnit tested.

- Final Submission (5/22 – **6/5**)
    - Submit final version of Elevator Simulation application. Including all outputs. Further requirement and submission details will be posted closer to the start date 5/22.
    - Code must be Javadoc'd and JUnit tested.

Project Assistance

If you are stuck on some code-related problem that you have exhaustively debugged yourself, you can email me a ZIP file of your entire project so that I can examine the problem. All emailed assistance requests must include a detailed description of the problem, and the details of what steps you have already taken in trying to determine the source of the problem.

Submissions & Grading

All submissions must include all code necessary to compile and run your application. Submitted code must be in the required package-folder form. ZIP'd project folder submissions are usually the best option (with compiled .class & JAR files removed before ZIP'ing). Only one team member needs to submit – one score will be generated for both team members.

The following are the key points that will be examined in your Part 1 classes when graded:

- Good Object Oriented Design & Implementation
- Properly formatted, useful Javadoc documentation
- Properly written, JUnit tests with good code coverage
- Proper Application Execution

When submitting, you should submit a ZIP file of your entire project so that I can compile and execute it on my end.

Late submissions will be penalized by 10% per week late. (i.e., one second late to 1 week late: 10% penalty, one week plus one second late to 2 weeks late: 20% penalty, etc.).

If you do not understand anything in this handout, please ask. Otherwise the assumption is that you understand the content.