Quarter Programming Project

2nd Implementation Phase

2nd Implementation Phase Minimum Requirements

In this project phase, the minimum deliverable is your implementation of the complete working code that includes working Person, Elevator, Elevator Call Box, Elevator Controller, Floor & Building code (results reporting not required in this submission).

The behaviors that are required in this phase reflect all the behaviors detailed in the activity diagrams from the original project handout.

More specifically, the working functionality should include:

- The input and use of the inputs listed in the original project handout. These may vary from the handout specification based upon the needs of your application and the data format you choose to use (XML, CSV, etc.).
- The simulation will generate a building with the number of floors and elevators as specified in the input. Elevators will have a max number of persons they can hold. (Inputs #3, #4 & #5 from the original project handout).
- The simulation will generate "N" new Persons per minute on some floor requesting an elevator to another floor (based upon inputs #9 & #10 from the original project handout).
- Elevators will move at the pre-determined rate of one floor per second (#6 from the original project handout).
- The simulation will operate for a specified duration. (#1 from the original project handout). At the end of that duration, no more people should be generated and all people already created will continue to their desired destination. Once they have all arrived at their desired destination, the simulation is over and the outputs can be generated/completed. (Outputs are not needed until the next project phase).
- The simulation can be run at real time or faster/slower than real time (based upon #1 from the original project handout). All real-world times (time per floor, elevator door open/close times, etc.) should be altered based upon the time-scale factor. For example, if the time-scale is 4 to 1 (four simulated minutes for each real-world minutes) then all times will be divided by four making things run four times faster than "normal".
- Remember – use the activity diagrams to determine the basic elevator behavior.

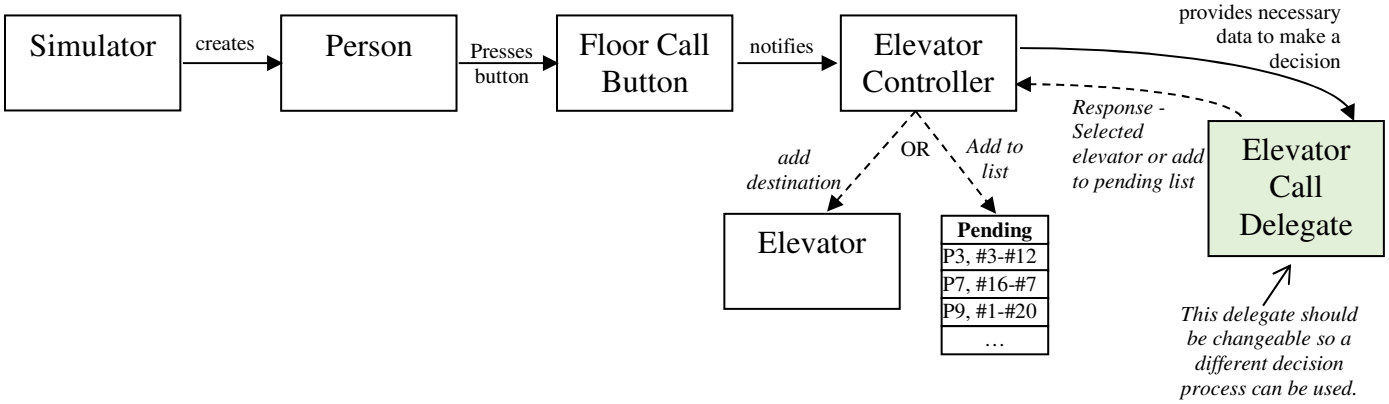
Variable Algorithms

Remember that you must design and code for variability in 2 of the simulation algorithms:

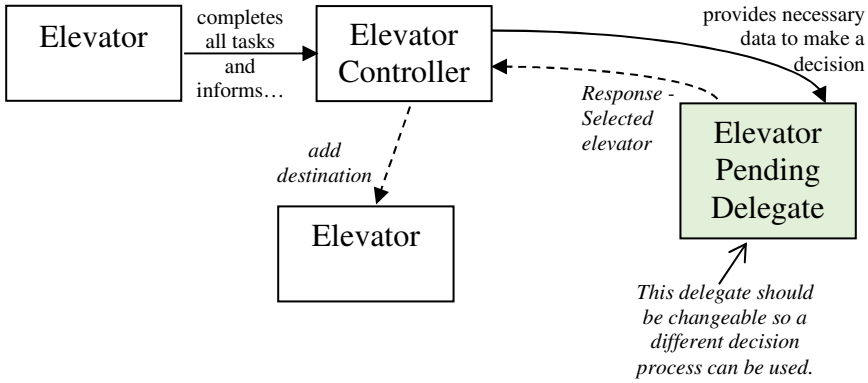
- Selection of the specific elevator to respond to an Elevator Up/Down Call.
- Selection and assignment of "Pending" elevator requests to elevators.

This means that when these situations arise, the elevator controller (assuming the elevator controller performs these tasks) should delegate the decision making process to a delegate:

1) Response to an Elevator Up/Down Call:



2) Selection and assignment of "Pending" elevator requests to elevators.



Basic Flow

- While simulation duration has not expired
  - For "N" people this minute
    - Create a new person
    - Select a start and destination floor
    - Add the person to the start floor
    - Press the appropriate button (up/down) on the call box on that floor

    Automatic processing continued by the other objects in their own threads *
  - End "for"

- End While

* *Person object(s) sit there on their start-floor until the elevator arrives. Once the elevator arrives at that floor, ANY Persons on that floor that want to go in the elevator's direction (Up/Down) should automatically enter. They will ride the elevator until they arrive at their destination floor at which point they will automatically exit.*

Outputs

The outputs in this phase will be the narrative output you should already be generating after completing the Initial Implementation Phase. For example:

```
[…]
10:40:10 Elevator 2 going to Floor 13, Full Destination List: [13, 14, 15]
10:40:10 Elevator 1 passing Floor 9 on the way to 11. Full Destination List: [11].
10:40:10 Elevator 2 passing Floor 7 on the way to 13. Full Destination List: [13, 14, 15].
10:40:11 Elevator 1 passing Floor 10 on the way to 11. Full Destination List: [11].
10:40:11 Elevator 2 passing Floor 8 on the way to 13. Full Destination List: [13, 14, 15].
10:40:12 Elevator 1 arrived at destination Floor 11. Doors open…
10:40:13 Elevator 2 passing Floor 9 on the way to 13. Full Destination List: [13, 14, 15].
10:40:14 Elevator 2 passing Floor 10 on the way to 13. Full Destination List: [13, 14, 15].
10:40:15 Elevator 1 Doors close. No further destinations.
10:40:15 Elevator 2 passing Floor 11 on the way to 13. Full Destination List: [13, 14, 15].
10:40:16 Elevator 2 passing Floor 12 on the way to 13. Full Destination List: [13, 14, 15].
10:40:17 Elevator 2 arrived at destination Floor 13. Doors open…
10:40:20 Elevator 2 Doors close. Continuing to next destination: 14. Full Destination List: [14, 15].
10:40:21 Elevator 2 arrived at destination Floor 14. Doors open…
10:40:24 Elevator 2 Doors close. Continuing to next destination: 15. Full Destination List: [15].
10:40:25 Elevator 2 arrived at destination Floor 15. Doors open…
10:40:28 Elevator 2 Doors close. No further destinations.
[…]
```

Note the above is only an example of output format – this is not expected output.

Development Phases

- *Design & Development Plan (4/10 – **4/24**) [Completed]*
  - *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***) [Completed]
  - 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) [Current]**
  - **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.

NOTE: Before asking for assistance, I expect that you will have gone through the proper steps to debug the issue yourself in order to discover the nature of the problem yourself, which includes verifying that your Unit Test coverage is complete.

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 2nd Implementation Phase 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.