SYSC 3303 Elevator Control System Final Report

Group: Group 4

Lab Section: Lab 5

Group Members: Akaash Kapoor, Hassan Jallad, Ali Alvi, Areeb Ul Haq, Raj Sandhu

Table of Contents:

Breakdown of Responsibilities

Breakdown of Responsibilities – Iteration 1:

Ali Alvi:

- Developed the Scheduler.java class.
- Took part in code review sessions and approved outstanding pull requests.

Akaash Kapoor:

- Developed the FloorSubsystem.java class.
- Took part in code review sessions and approved outstanding pull requests.

Areeb Ul Haq:

- Developed the ElevatorSubSystem.java class.
- Took part in code review sessions and approved outstanding pull requests.

Hassan Jallad:

- Developed unit tests to show the program reading the input file and passing the data between the different subsystems.
- Took part in code review sessions and approved outstanding pull requests.

Raj Sandhu:

- Developed UML diagram.
- Developed Sequence diagram.
- Wrote the README file.
- Took part in code review sessions and approved outstanding pull requests.

Breakdown of Responsibilities – Iteration 2:

Ali Alvi:

- Extended development for classes within the scheduler folder.
- Developed the scheduler state machine.

- Worked as a primary developer on scheduler and state machine diagram alongside Raj Sandhu.
- Assisting in debugging general issues with the scheduler classes.
- Took part in code review sessions and approved outstanding pull requests.

Akaash Kapoor:

- Redesigned codebase to take advantage of the blocking queue data structure.
- Performed debugging of code issues.
- Assisted Raj with UML class diagram
- Revised unit tests.
- Took part in code review sessions and approved outstanding pull requests.

Areeb Ul Haq:

- Extended unit test development.
- Assisted in general debugging of the program.
- Assisted Raj in designing sequence diagram.
- Took part in code review sessions and approved outstanding pull requests.

Hassan Jallad:

- Extended development for the classes within the elevatorSubsystem folder.
- Developed the elevator state machine.
- Performed debugging of code issues.
- Developed elevator state machine diagram.
- Took part in code review sessions and approved outstanding pull requests.

Raj Sandhu:

- Developed UML diagram.
- Developed Sequence diagram.
- Worked with Ali to develop state machine diagram.
- Wrote the README file.
- Assisted Ali with the classes within the scheduler folder, performing code cleanups and assisting in debugging.
- Took part in code review sessions and approved outstanding pull requests.

Breakdown of Responsibilities – Iteration 3:

Ali Alvi:

- Extended development for classes within the scheduler folder such that the scheduler can now coordinate the movement of cars.
- Assisted in debugging the program such that it worked successfully.
- Revised scheduler state machine diagram to address feedback from iteration 2.
- Took part in code review sessions and approved outstanding pull requests.

Raj Sandhu:

- Developed UML diagram.
- Developed Sequence diagram.
- Wrote README.
- Assisted in debugging the program such that is worked successfully.
- Assisted lead scheduler developer Ali Alvi in code cleanup.

Akaash Kapoor:

- Extended development for the classes pertaining to the floor subsystem.
- Developed the timer class used to measure elevator arrival timings.
- Led in debugging the program such that it worked successfully.
- Took part in code review sessions and approved outstanding pull requests.

Hassan Jallad:

- Extended development for the classes pertaining to the elevator subsystem.
- Led in debugging the program such that it worked successfully.
- Took part in code review sessions and approved outstanding pull requests.
- Revised elevator state machine diagram to address feedback from iteration 2.

Areeb Ul Haq:

- Extended development for the test classes related to the program.
- Assisted in debugging the program such that it worked successfully.
- Took part in code review sessions and approved outstanding pull requests.

Breakdown of Responsibilities - Iteration 4:

Ali Alvi:

- Updated UML diagram to incorporate changes made in iteration 4.
- Updated elevator state machine diagram to incorporate changes made in iteration 4.
- Took part in code review sessions and approved outstanding pull requests.

Raj Sandhu:

- Developed timing diagrams to showcase the various faults considered in iteration 4 (doors stuck open, elevator stuck between floors), as well as a successful run.
- Wrote README.
- Took part in code review sessions and approved outstanding pull requests.

Akaash Kapoor:

- Extended development of the timer class used to measure elevator arrival timings.
- Extended development for the classes pertaining to the elevator subsystem and the scheduler to detect and handle faults.
- Took part in code review sessions and approved outstanding pull requests.

Hassan Jallad:

- Extended development for the classes pertaining to the elevator subsystem and the scheduler to detect and handle faults.
- Took part in code review sessions and approved outstanding pull requests.

Areeb Ul Haq:

- Extended development for the test classes related to the program.
- Took part in code review sessions and approved outstanding pull requests.

Breakdown of Responsibilities – Iteration 5:

Ali Alvi:

- Added GUI component to display console output on a jFrame.
- Worked with Raj Sandhu to update the UML diagram.

Raj Sandhu:

- Wrote README.
- Worked with Akaash Kapoor to develop scheduler timing diagrams.
- Worked with Ali Alvi to update the UML diagram.
- Performed statistical analysis of timing.

Akaash Kapoor:

- Worked with Raj Sandhu to develop scheduler timing diagrams.
- Worked with Hassan Jallad to add timing instrumentation to measure how long program takes to run.

Hassan Jallad:

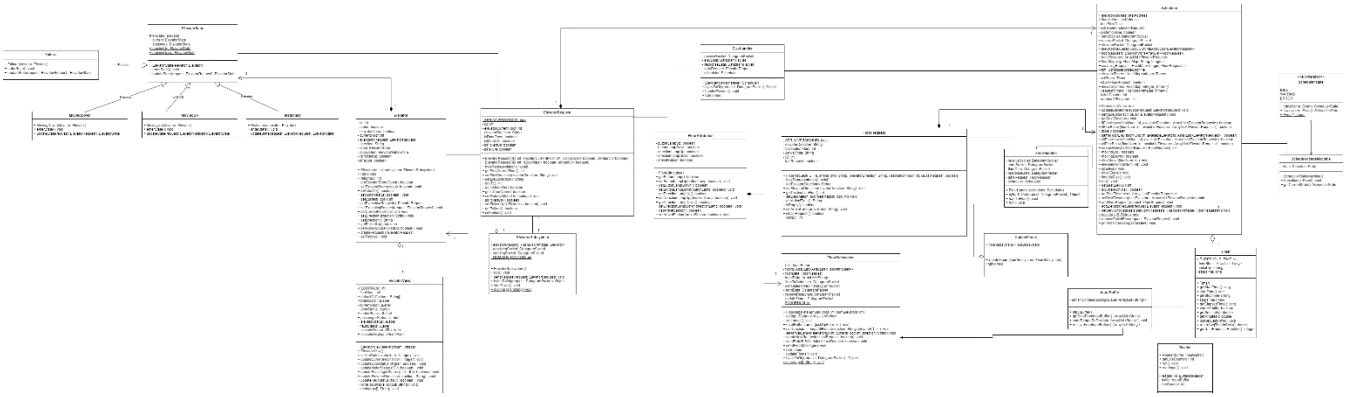
- Worked with Akaash Kapoor to add timing instrumentation to measure how long program takes to run.
- Added new sequence diagram showing error scenarios.

Areeb Ul Haq:

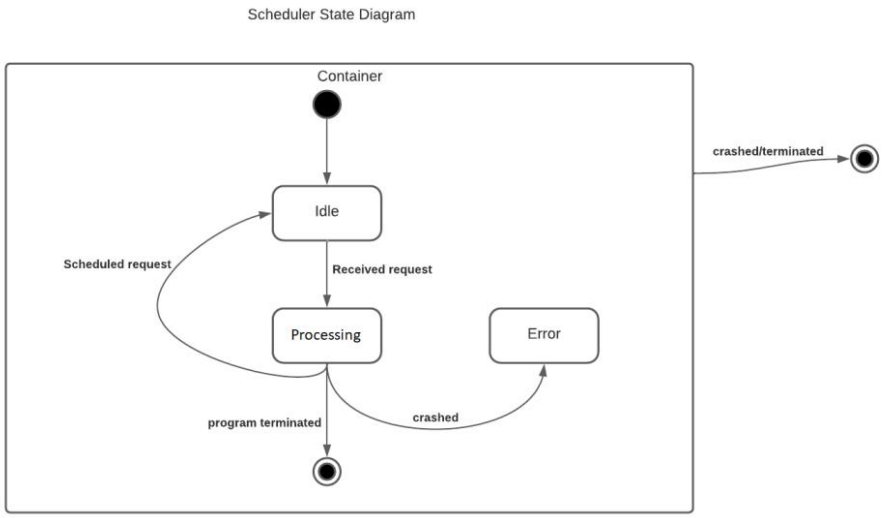
- Refined unit tests to consider program timing analysis.

Diagrams:

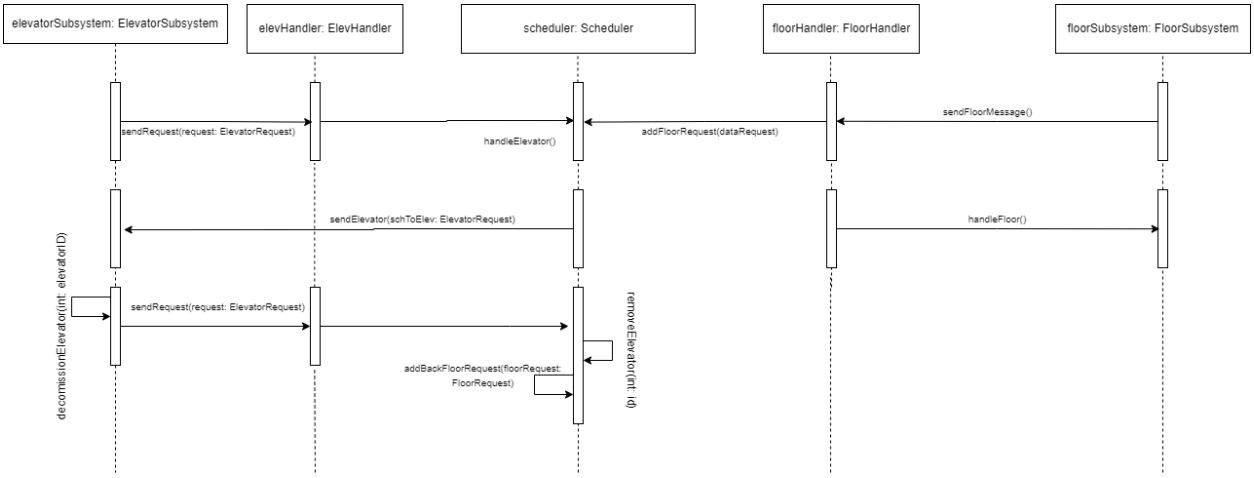
UML Diagram:



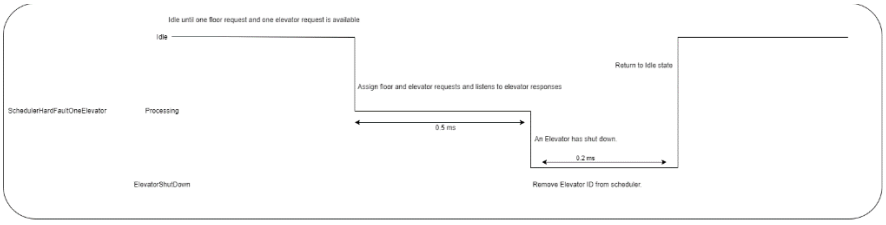
Scheduler State Machine Diagram:

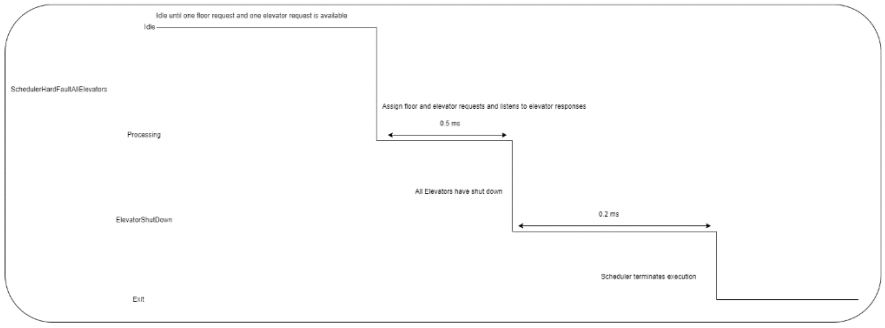


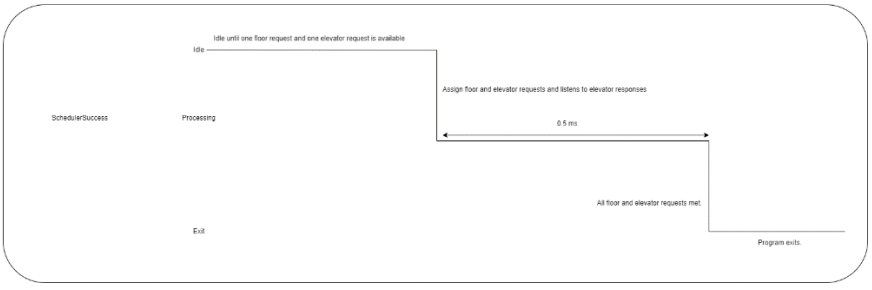
Error Scenarios Sequence Diagram:



Scheduler Timing Diagrams:







Note, for other diagrams not requested with the report, please see the documentation folder in the submission.

Detailed Set Up and Test Instructions:

Set Up Instructions:

- Download the zip file submission from Brightspace to a location of your choice.
- Open up the Eclipse IDE.
- Click File, Import, General, Projects from Folder or Archive, Next, then click Archive in the top right of the screen, find where you stored the zip file submission, select it, then hit finish.
- Ensure Eclipse is correctly configured to run multiple main methods.
- Run the Scheduler.java main method in its own console window.
- Pin the current console window.
- Create a new console window.
- Run the ElevatorSubsystem.java main method.
- Pin the second console window.
- Create a new console window.
- Run the FloorSubsystem.java main method.
- Pin the third console window.
- Open the GUI dialog window for scheduler and choose values.
- Open GUI dialog window for floor and choose values.
- Open the elevator GUI and there will be 4 GUI windows on top of each other.
- Separate each GUI window to its own space to view all 4 elevators at the same time.

Notes:

- Values for each GUI dialog window have to be exactly the same for the program to work properly.

Testing Instructions:

- Download the zip file submission from Brightspace to a location of your choice.
- Open up the Eclipse IDE.
- Click File, Import, General, Projects from Folder or Archive, Next, then click Archive in the top right of the screen, find where you stored the zip file submission, select it, then hit finish.
- Navigate to the InputTests.java file (src/tests/InputTests.java) and hit run.
- Navigate to the StateTests.java file (src/tests/StateTests.java) and hit run.
- Navigate to the SchedulerTests.java file (src/tests/SchedulerTests.java) and hit run.
- Navigate to the TimerAndFailureTests.java file (src/tests/TimerAndFailureTests.java) and hit run.
- Navigate to the IntegrationTests.java file (src/tests/IntegrationTests.java) and hit run.

Results From Measurements:

To conduct a timing statistical analysis, the program run executed a total of 30 times, in order to collect 30 samples. This is because, when calculating confidence intervals using the Z Score table, a minimum of 30 samples must be available. From here, a summary table of the computed measurements shall be provided. For all analysis see Statistics for Elevator Timings.csv.

Mean of Scheduler Time (s)	78.969
Mean Time of Elevator 1 (s)	7.617333333
95% Confidence Interval of Elevator 1 (s)	0.985399916
Mean Time of Elevator 2 (s)	8.571
95% Confidence Interval of Elevator 2 (s)	1.11855694
Mean Time of Elevator 3 (s)	8.708666667
95% Confidence Interval of Elevator 3 (s)	0.866058036
Mean Time of Elevator 4 (s)	8.708666667
95% Confidence Interval of Elevator 4 (s)	1.068710028

Based on these computed values, it is shown that all of the elevators have a very similar completion time which indicates that the elevators were treated fairly and requests were attended to fairly, thus reducing any potential starvation that may occur.

Another calculation that was performed was the mean time across all elevators which indicates the average time for all the elevator to complete. This was computed through the use of the following formula.

$$\frac{\sum_{i=1}^{4} numberRequests(Elevator_i) \times executionTime(Elevator_i)}{TotalRequests}$$

Here is a sample calculation for the first run of the program.

$$\frac{(6.05)(3) + (6.81)(1) + (9.33)(1) + (12.6)(2)}{7}$$

$$= 8.498571429\ seconds.$$

This calculation was then computed for all runs, and its computed mean is 8.090333333 seconds, and its associated 95% confidence interval is 0.245681823 seconds.

Reflection on Design:

We really liked the design of the scheduler because we were able to generate a scheduling algorithm using multiple threads and thread-safe libraries. By implementing CopyOnWriteArrayList and HashMap libraries, the scheduler would be able to keep track of both floor and elevator requests both in real time. The floor subsystem was also designed to send requests in real time according to the timing between requests and the elevators were able to operate based on scheduled requests from the scheduler. Both the elevator and scheduler were able to handle the error scenarios accordingly. If there was one thing that we could change about the design would be to incorporate boundary, controller, and entity specific classes to the design to breakdown the system more into smaller components that having one subsystem class have most of the operations. Some of these classes includes a motor controller, abstract socket handling class, floor lamp controller, floor, and elevator request controller classes, etc. The design of the GUI could have also been improved upon. Since the last iteration was under a very limited time constraint, the easiest option was to simply report the elevator's positions through the console rather than visually display elevators going up and down.