SYSC3303
Real Time Concurrent Systems

Elevator Control System
Final Project Report

Group B1-G1

Joshua Robson, 101195802
Liam Kavanagh, 101182915
Matthew Huybregts, 101185221
Sean Pruss, 101189970
Abed Qubbaj, 101205030

TA: Igor Bogdanov

Friday April 10th, 2024

Table of Contents

1 - Breakdown of Team Member Contributions

Iteration 1:

- Scheduler subsystem: Liam Kavanagh
- Floor subsystem: Matthew Huybregts
- Elevator subsystem: Sean Pruss
- Test code: Joshua Robson
- UML diagrams and README: Abed Qubbaj

Iteration 2:

- Liam Kavanagh: Scheduler State Diagram and Scheduler Coding
- Matthew Huybregts: Floor refactor and Scheduler Coding
- Sean Pruss: Elevator State Diagram and Elevator Coding
- Joshua Robson: JUnit tests
- Abed Qubbaj: Class diagram, sequence diagram and README.md

Iteration 3:

- Liam Kavanagh: Scheduler algorithm and coding
- Matthew Huybregts: Coding, debugging, refactoring
- Sean Pruss: Debugging, also helped out with JUnit tests
- Joshua Robson: JUnit tests
- Abed Qubbaj: Updated diagrams and README.md

Iteration 4:

- Liam Kavanagh: Implementing Door Faults and coding
- Matthew Huybregts: Implementing Timer Class
- Sean Pruss: Timing Diagrams
- Joshua Robson: JUnit tests, Debugging, Coding
- Abed Qubbaj: Updated diagrams and README.md

Iteration 5:

- Liam Kavanagh: Implementing GUI
- Matthew Huybregts: Implementing Capacity Limits
- Sean Pruss: Implementing GUI
- Joshua Robson: JUnit tests, Debugging, Updated UML Diagrams
- Abed Qubbaj: Updated diagrams and README.md

2 - Diagrams of the Elevator System

2.1 UML Class Diagrams

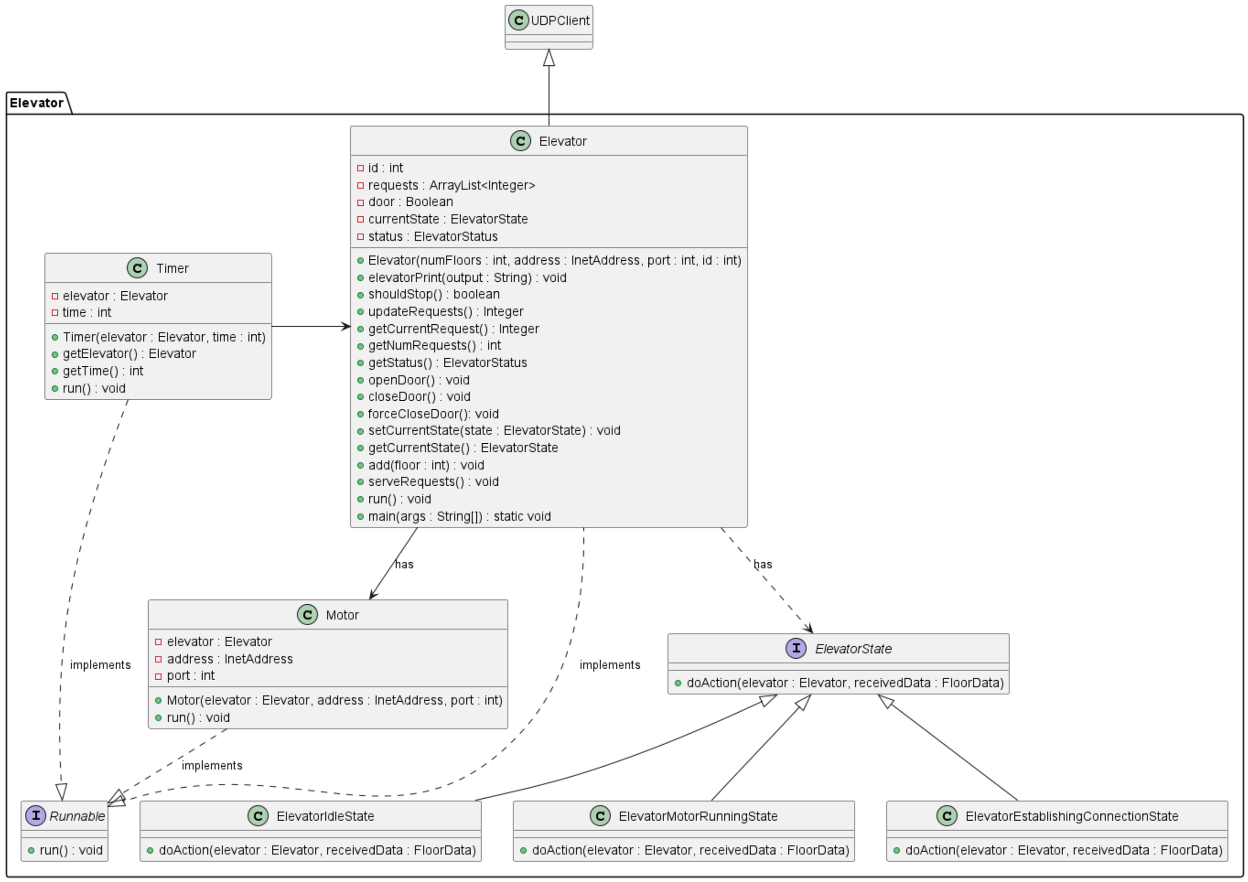

Figure 1: Elevator Class Diagram

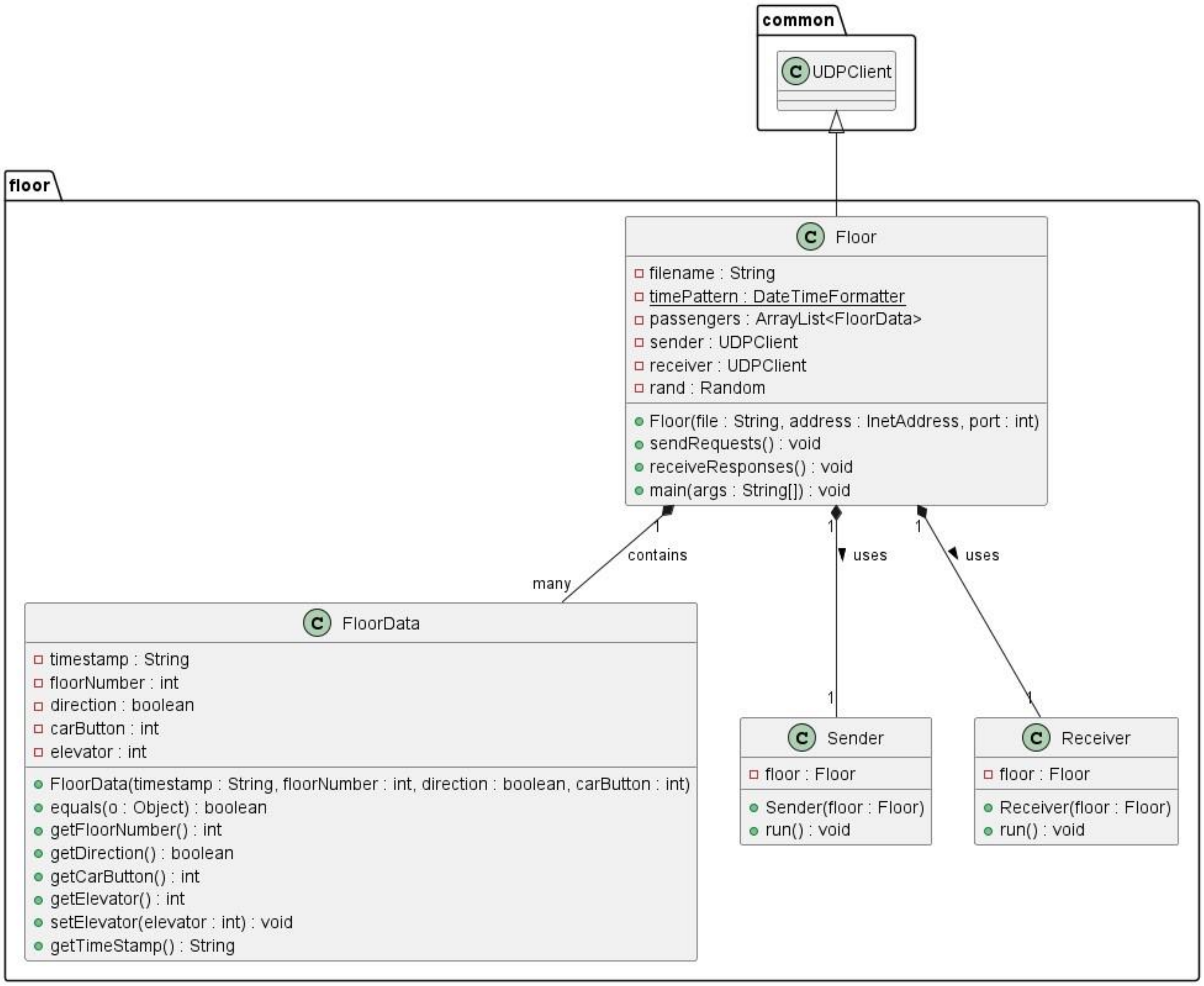

Figure 2: Floor Subsystem Diagram

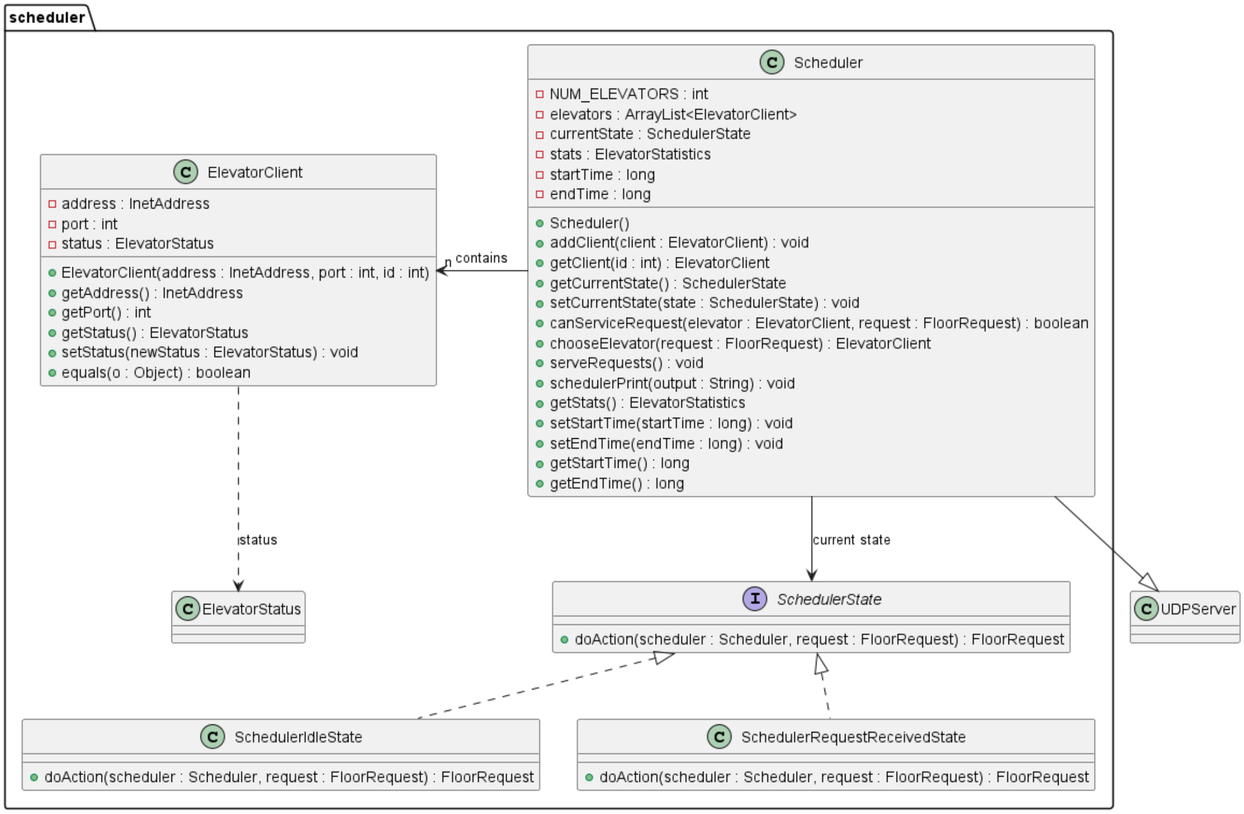

Figure 3: Scheduler Subsystem Diagram

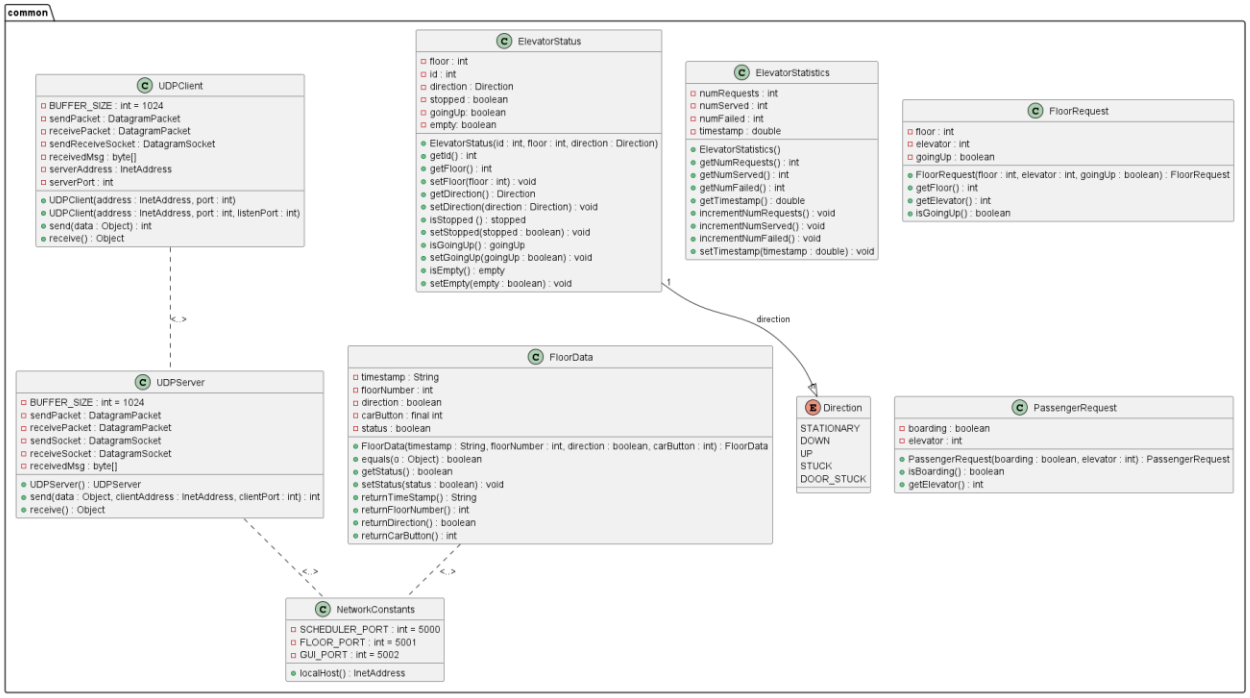

Figure 4: Common Subsystem Diagram

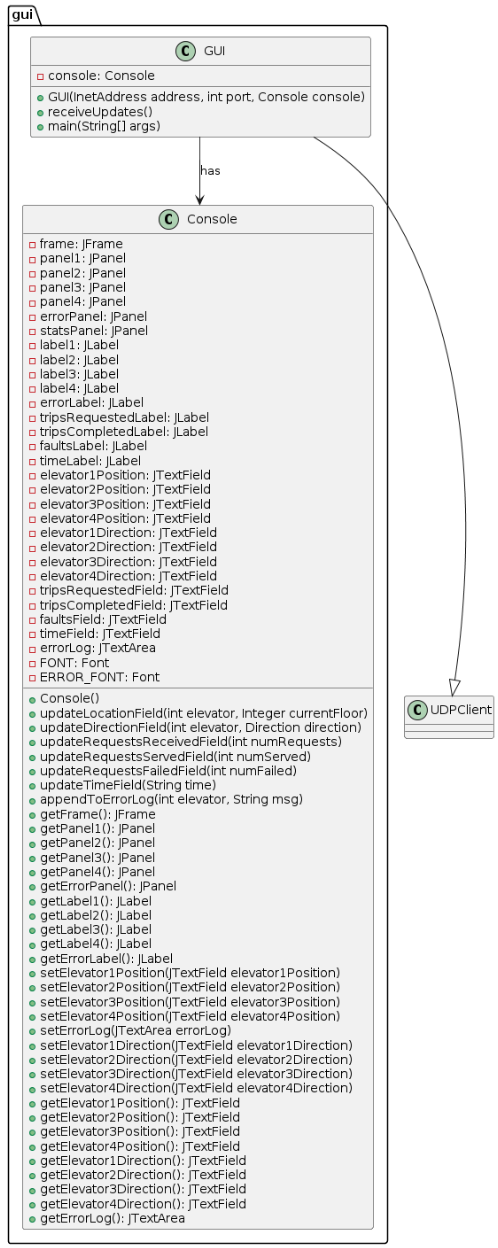

Figure 5: GUI subsystem diagram

2.2 - State Machines

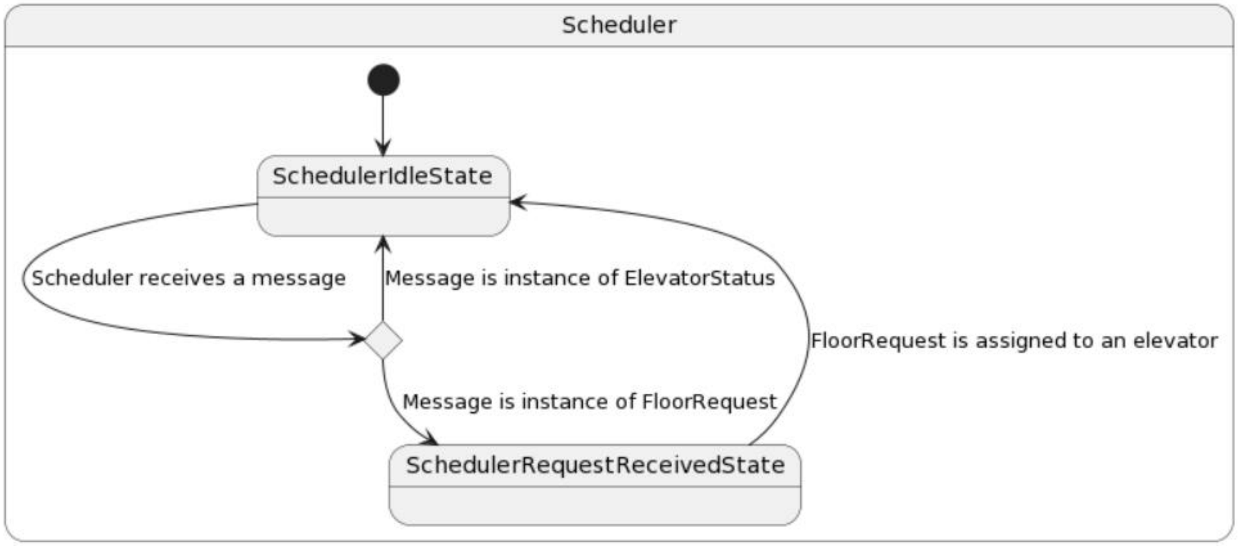

Figure 6: Scheduler State Diagram

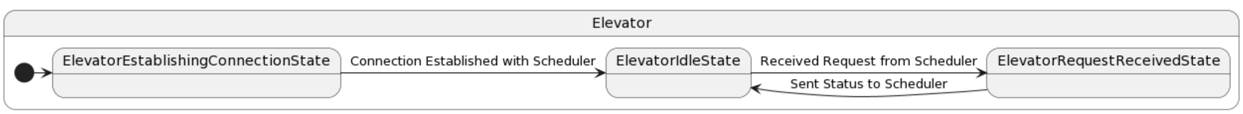

Figure 7: Elevator State Diagram

2.3 - Sequence Diagrams

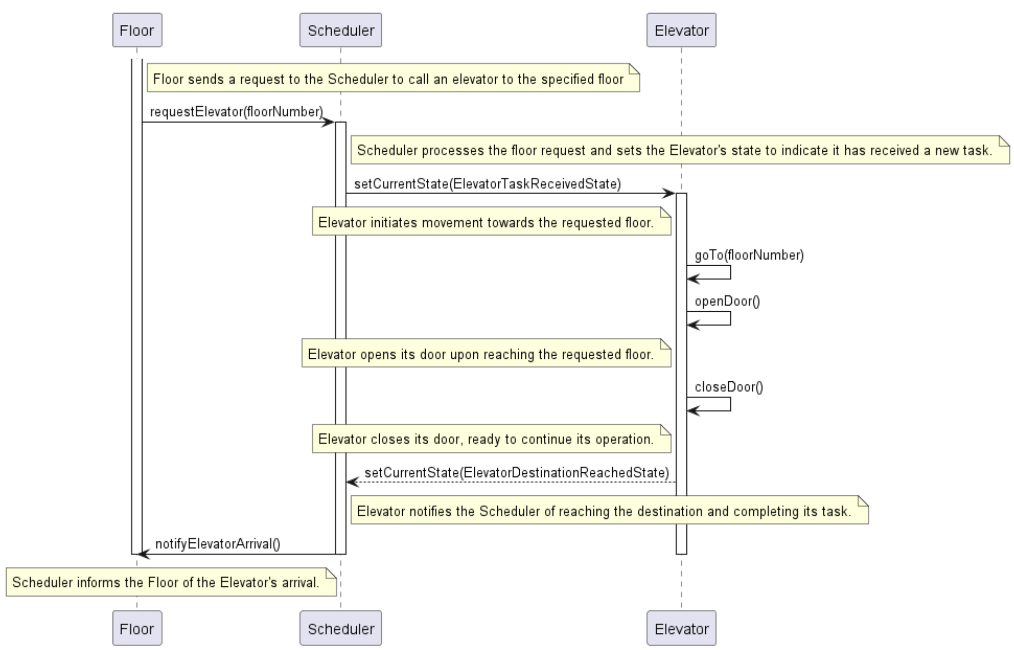

Figure 8: Elevator System Sequence Diagram

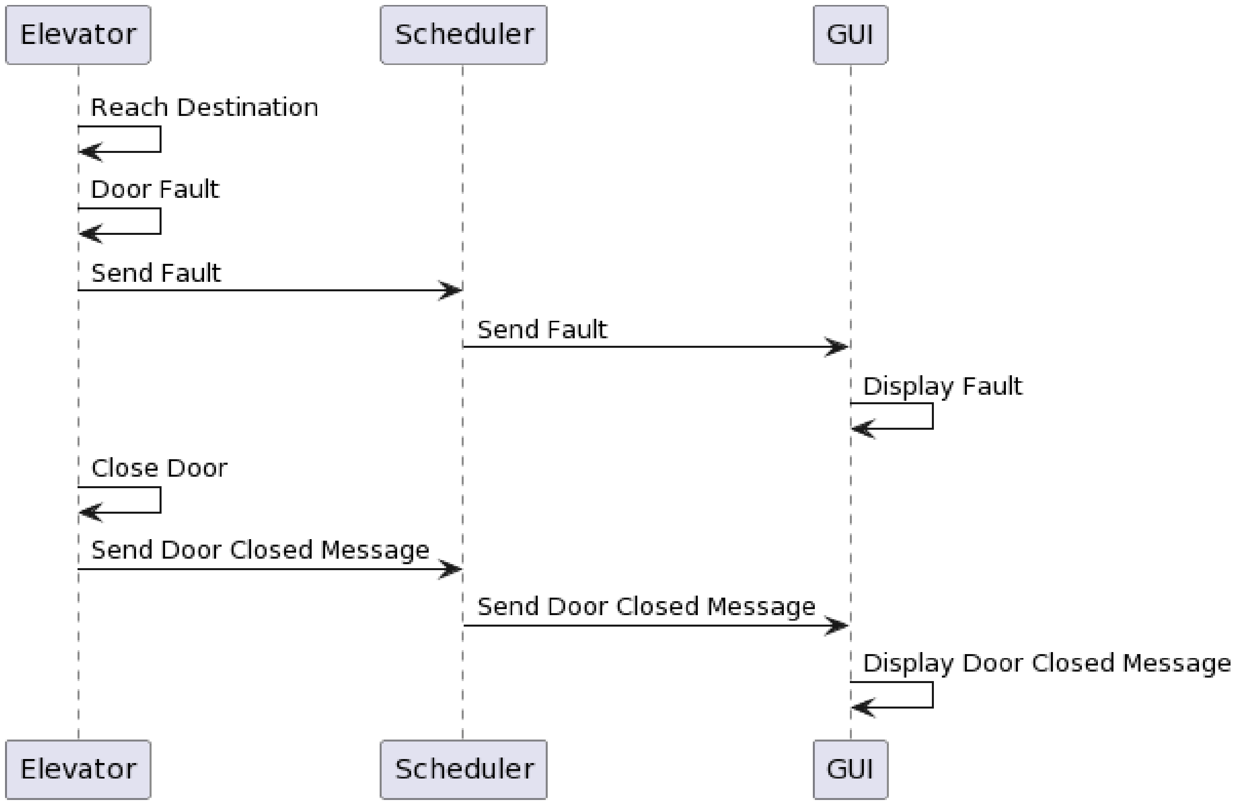

Figure 9: Door Fault Sequence Diagram

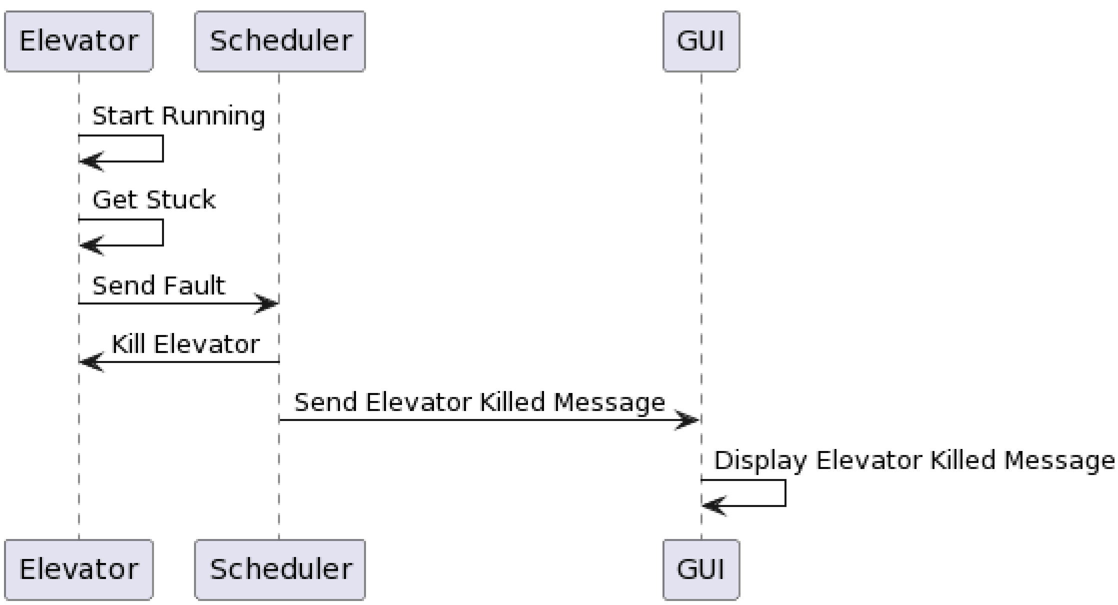

Figure 10: Elevator Stuck Fault Sequence Diagram

2.4 - Timing Diagrams

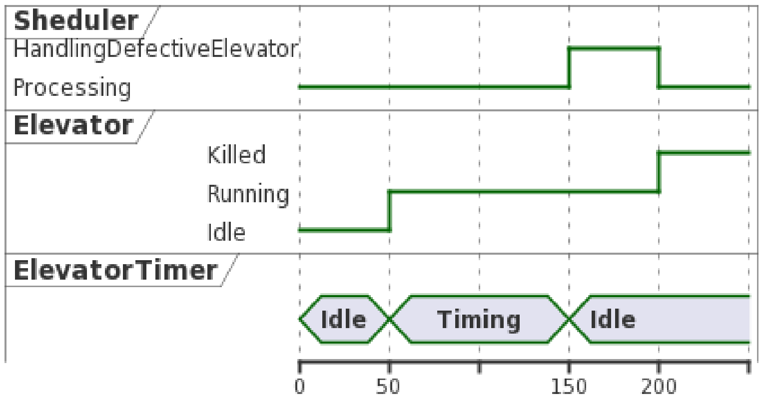

Figure 11: Elevator too slow to reach destination diagram

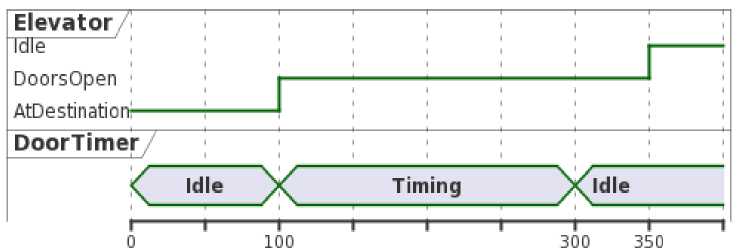

Figure 12: Elevator doors stuck open timing diagram

3 - Detailed Set-Up and Instructions

Requirements

Java SDK 21, and Language Level set to `Default` SDK in Intellij.

Usage

[IMPORTANT] *Run the files in this order for the system to initialize properly*

1. Run `scheduler.Scheduler.java`
2. Run `gui.GUI.java`
3. Run `elevator.Elevator.java`
4. Run `floor.Floor.java`

Running Test Cases: *Run them in any order and anytime as they are independent of each other and the system files*

Input File Format

The floor.Floor subsystem reads input from a file, and each line in the file represents an event. The format is as follows:

[timestamp] [floor_number] [direction (up/down)] [requested_floor]

4 - Result from the Measurements

	A	B	C	D
1	**Split**	**Total Time (s)**	**Loading Time (note we were only able to get samples for 2 people)**	**Time Between Floors (Column B - C)**
2	1 (total and loading time is junk data)	4.85	0	4.85
3	2	19.82	12.29	7.53
4	3	17.37	11.67	5.7
5	4	18.17	10.99	7.18
6	5	18.2	13.1	5.1
7	6	19.74	12.33	7.41
8	7	19.27	12	7.27
9	8	19.46	12.65	6.81
10	9	18.31	12.34	5.97
11	10	18.58	12.87	5.71
12	11	18.16	12.45	5.71
13	12	18.3	12.09	6.21
14	13	18.24	12.53	5.71
15	14	18.54	13.02	5.52
16	15	18.32	12.39	5.93
17	16	18.76	12.78	5.98
18	17	17.98	11.97	6.01
19	18	18.51	12.22	6.29
20	19	18.45	12.91	5.54
21	20	20.62	12.42	8.2
22	21	18.16	12.56	5.6
23				
24	**Mean**	18.99	12.425	5.225
25	**Standard Deviation**	1.173797257	0.190918831	0.530330086
26	**Confidence Interval (95% confidence)**	18.475, 19.504	12.34 , 12.509	4.999, 5.452
27				
28	**Split**	**Time (not including loading time)**		
29	1	34.05		
30	2	34.69		
31				
32	**Mean**	34.37		
33	**Standard Deviation**	0.45254834		
34	**Confidence Interval (95% Confidence)**	33.743, 34.997		
35				

Figure 13: Excel sheet of the Data collection

Assumptions:

- There is a distance of 4 meters between floors
- The rate of acceleration is the same as that of the deceleration
- Assume the rate of acceleration is the same no matter the number of floors
- For an adjacent floor, the elevator never maintains a constant speed. It speeds up and slows again

1. **Average Speed (Adjacent Floors)**

$v = \Delta d / \Delta t$
$v = 4m / 5.225s$
$v = 0.766 m/s$

2. **Average Speed (Multiple Floors)**

   $v = \Delta d/\Delta t$
   $v = 84m/34.37s$
   $v = 2.44m/s$

3. **Acceleration**

   Since we're assuming that the elevator speeds up and then immediately slows down for a single floor, we will assume the maximum velocity is 1.532 m/s (2x the average).

   $a = v/(t/2)$
   $a = 1.532m/s/(5.225/2)s$
   $a = 0.586m/s^2$

4. **Max Speed (Multiple Floors)**

   We estimated it takes around 4.73 seconds for the elevator to reach its top speed based on how it felt (may or may not be accurate)

   $v = at$
   $v = (0.586m/s^2)(4.73s)$
   $v = 2.77m/s$

5 - Reflection of our Design

5.1 Aspects of the Design We Liked

Our design allowed us to easily implement additional features incrementally. When we approached our design, we wanted to reduce the amount of possible refactoring as much as possible so we could focus on new features. Our project is also well structured, with directories for documentation, source code, and library files which are split into subdirectories such as common, elevator, scheduler, floor, and GUI for the src directory. Our GUI was a separate subsystem of the project, which made it a lot easier to implement the scheduler sending updates to it.

5.2 What Can Be Improved Upon

One improvement we can add is to make the Scheduler and GUI multithreaded. With a multithreaded Scheduler and GUI, we can process requests from multiple elevators and display them at once. This would increase the throughput through increased parallelism. We should also implement a way to deal with the case in which there is no elevator currently available for a passenger. This could be done by notifying the floor when there are no available elevators and having the floor resend the request after a set time and a maximum number of retries for these requests.