SYSC 3303: Real-Time Concurrent Systems
Final Project Report

Author names: Krishang Karir (101080230), Abdelrahman Darwish (101087316), Khalil Aalab (101070879), Jonas Hurlen (101068542), Michael Fairbairn (101071300)
Instructor's name: Shikharesh Majumdar
Due date: April 7th, 2020

Carleton
UNIVERSITY

Table of contents

1.0 RESPONSIBILITIES

The following section consists of a breakdown of responsibilities of each team member for each iteration.

1.1 Iteration 1

- Khalil Aalab: Setting up a work environment, incorporating various members' work, notifying the floor from scheduler via acknowledged linked list.
- Abdelrahman Darwish: Floor subsystem thread, reading input file.
- Michael Fairbairn: JUnit testing, separate packages, general code cleanup.
- Jonas Hurlen: README, scheduler communication & logic, elevator framework.
- Krishang Karir: Constructing class diagram, sequence diagram and verifying them with the requirements.

1.2 Iteration 2

- Khalil Aalab: Implementation of a state machine, organizing files in the project.
- Abdelrahman Darwish: Reading input file.
- Michael Fairbairn: JUnit testing.
- Jonas Hurlen: README, scheduler communication & logic, elevator framework.
- Krishang Karir: Constructing class diagrams, sequence diagrams and state machine diagrams and documentation.

1.3 Iteration 3

- Khalil Aalab: Assisting with reading input, as well as UDP communication, as well as debugging and integrating code.
- Abdelrahman Darwish: Reading input file for elevator subsystem and floor subsystem, UDP communication between subsystems.
- Michael Fairbairn: JUnit testing.
- Jonas Hurlen: README, scheduler communication & logic (algorithm), and elevator framework.
- Krishang Karir: Constructing class diagrams, sequence diagrams and state machine diagrams, Java doc and documentation.

2.0 DIAGRAMS

The following section displays the cumulative diagrams obtained in our latest iteration.

2.1 UML Class Diagrams

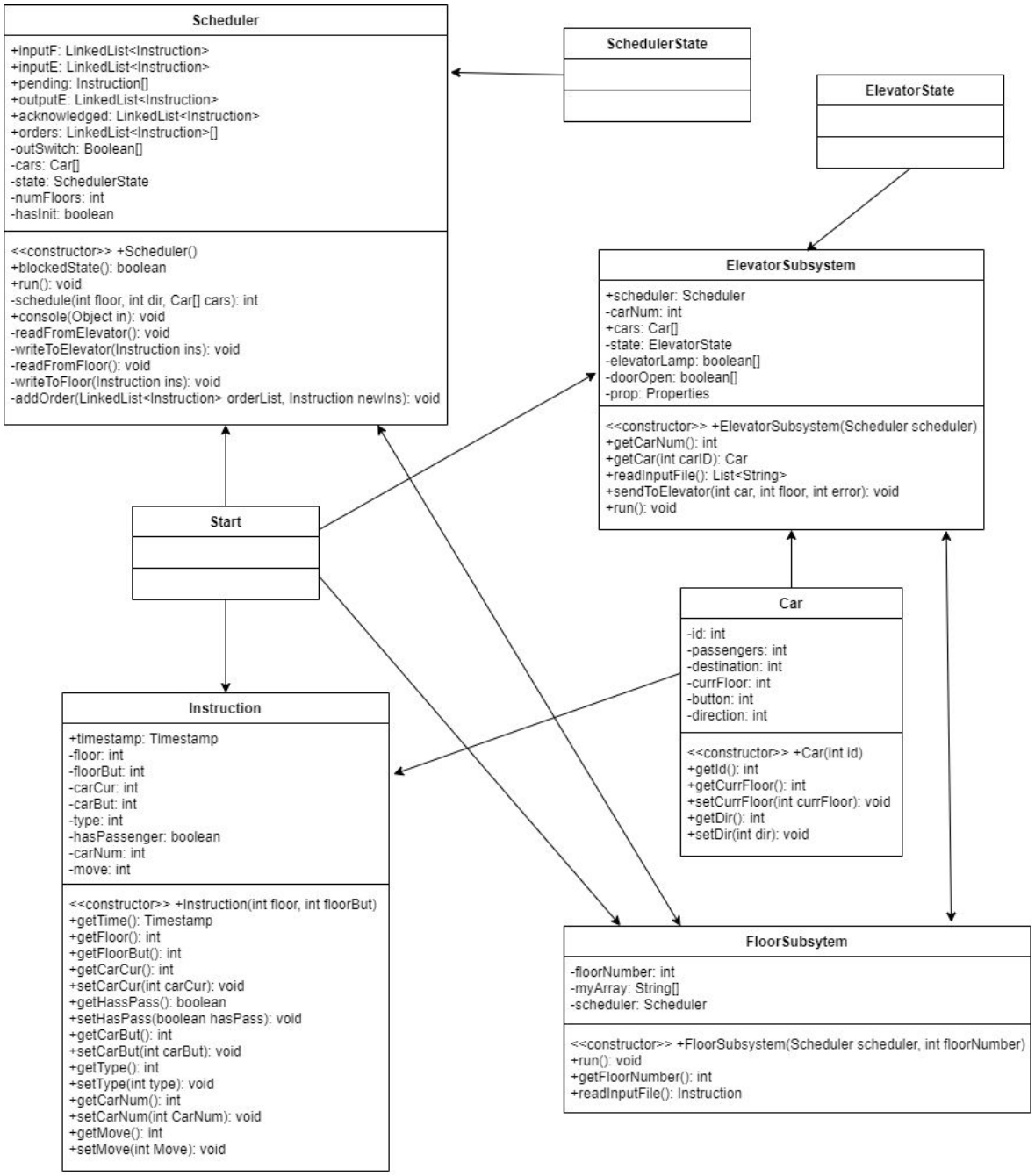

Figure 1: UML Class Diagram

2.2 State Machine Diagrams

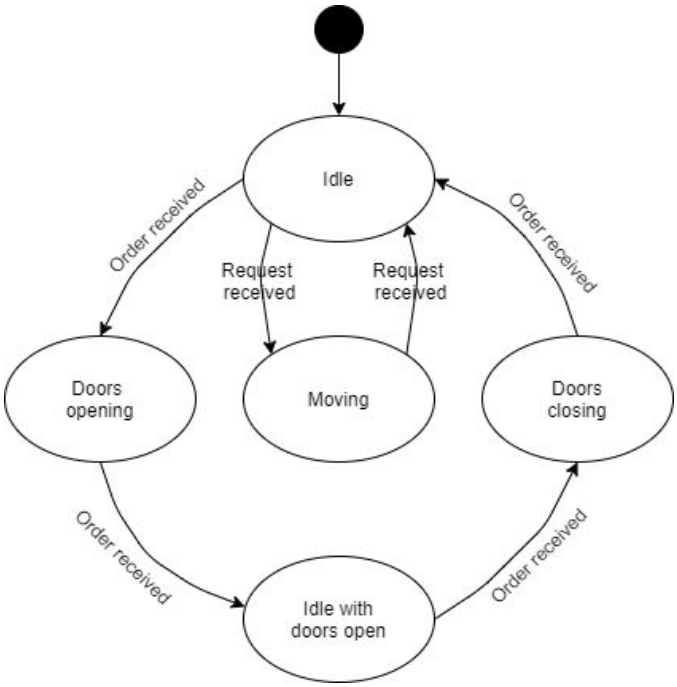

Figure 2: Elevator State Machine

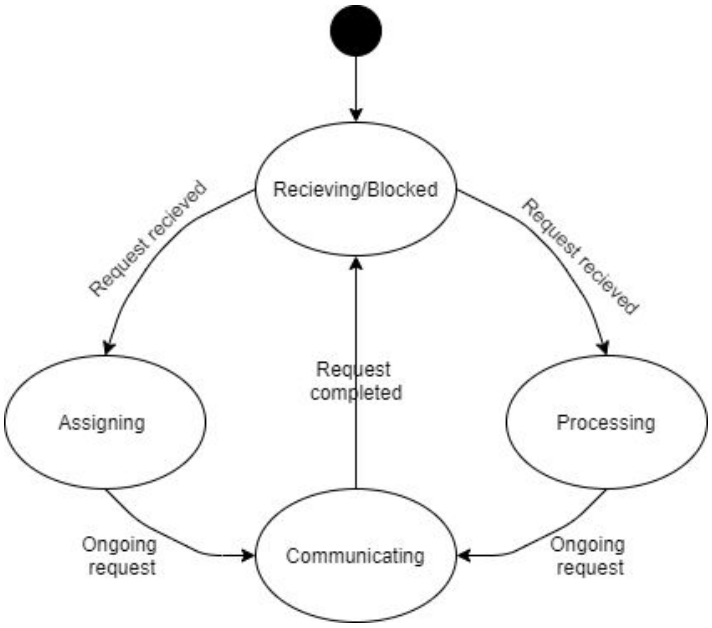

Figure 3: Scheduler State Machine

2.3 Sequence Diagrams

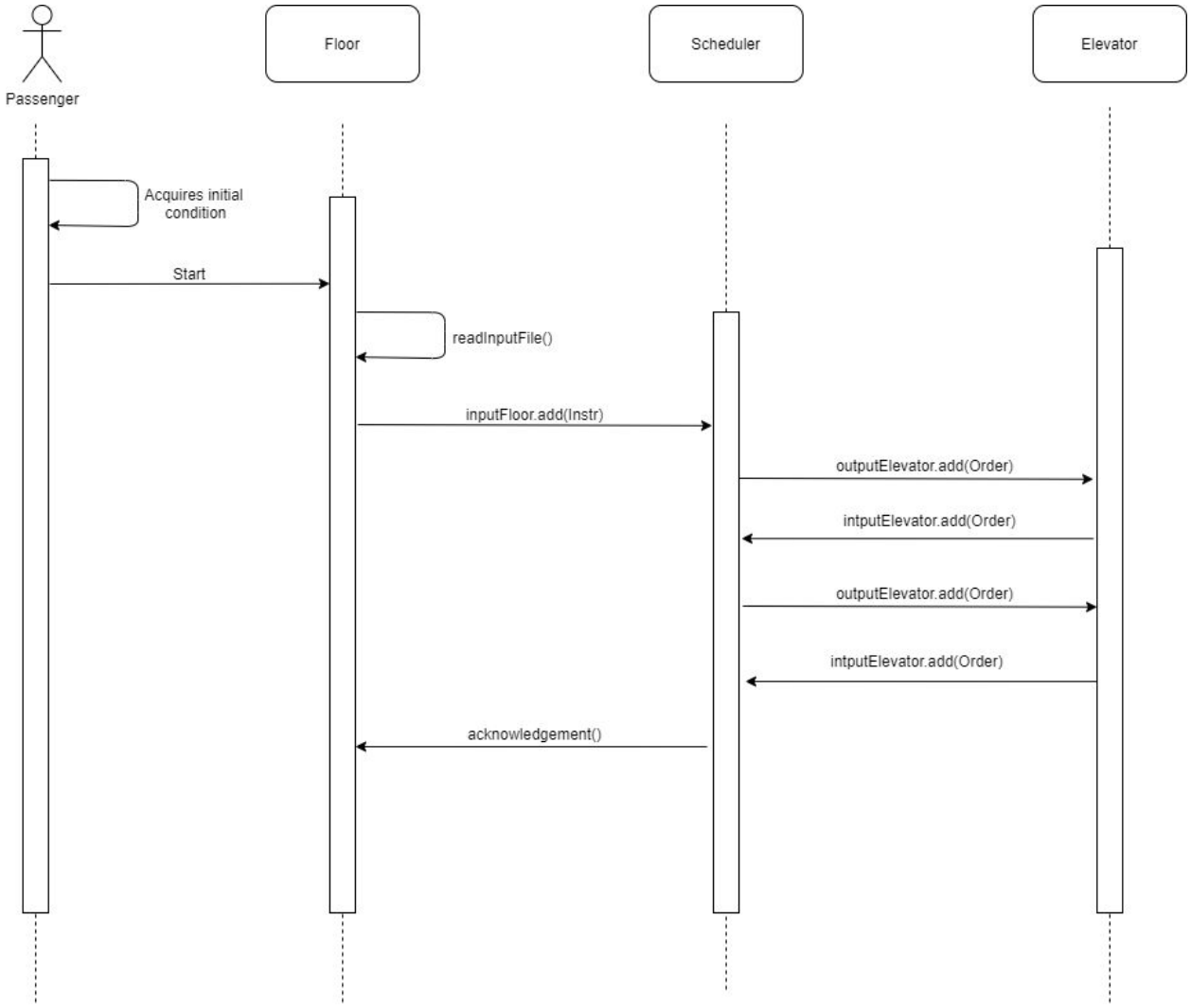

Figure 4: Sequence Diagram

3.0 SETUP & TEST INSTRUCTIONS

This section outlines the instructions in order to run the systems without any complications.

3.1 Installation

All java files as well as the text (.txt) input file must be included in the same project to ensure proper communication and functionality.

3.2 Running

To begin running the program, a predefined running order must be followed. The running order is as follows: Scheduler class, ElevatorSubsystem class, then the FloorSubsystem class. This

order must be followed to ensure correct system behaviour. If desirable, the text (.txt) input file can be filled with additional instructions for the program.

4.0 RESULTS

This section contains the resulting data collected by our team from the Dunton Tower elevators.

Table 1: Results from data collected as well as measurements

Run	Floor Started	Floor End	Floor Difference	Load time	Move Time	Move Time per F	Time Taken (s)	Distance (m)	erage Velocity (m	celeration (m/s^2)
1	2	7	5	10.36	15.24	3.048	25.6	20	0.78125	0.06103515625
2	7	22	15	8.46	27.6	1.84	36.06	60	1.663893511	0.09228472051
3	22	2	20	12.44	35.25	1.7625	47.69	80	1.677500524	0.07035020022
4	2	22	20	11.27	36.53	1.8265	47.8	80	1.673640167	0.07002678525
6	2	3	1	11.11	10.3	10.3	21.41	4	0.1868285848	0.01745246004
7	3	4	1	10.62	8.63	8.63	19.25	4	0.2077922078	0.02158880081
8	4	5	1	10.21	8.25	8.25	18.46	4	0.2166847237	0.02347613475
9	5	6	1	10.31	8.93	8.93	19.24	4	0.2079002079	0.02161124822
10	6	7	1	10.58	9.43	9.43	20.01	4	0.19990005	0.01998001499
11	7	8	1	10.89	8.28	8.28	19.17	4	0.2086593636	0.02176936501
12	8	9	1	10.17	8.98	8.98	19.15	4	0.2088772846	0.02181486001
13	9	10	1	10.67	8.86	8.86	19.53	4	0.204813108	0.02097420461
14	10	11	1	10.94	8.01	8.01	18.95	4	0.2110817942	0.02227776192
15	11	12	1	10.86	8.74	8.74	19.6	4	0.2040816327	0.02082465639
16	12	13	1	10.01	8.81	8.81	18.82	4	0.2125398512	0.02258659418
17	13	14	1	11.28	9.22	9.22	20.5	4	0.1951219512	0.01903628792
18	14	15	1	12.98	7.95	7.95	20.93	4	0.1911132346	0.01826213422
19	15	16	1	11.15	6.26	6.26	17.41	4	0.2297530155	0.02639322407
20	16	17	1	11.3	8.1	8.1	19.4	4	0.206185567	0.02125624402
21	17	18	1	11.39	7.77	7.77	19.16	4	0.2087682672	0.0217920947
22	18	19	1	11.28	8.03	8.03	19.31	4	0.2071465562	0.02145484787
23	19	20	1	12.21	8.05	8.05	20.26	4	0.1974333662	0.01948996705
24	20	21	1	10.03	8.16	8.16	18.19	4	0.2199010445	0.02417823469
25	21	22	1	12.24	7.21	7.21	19.45	4	0.205655527	0.02114709789
26	22	17	5	11.02	14.28	2.856	25.3	20	0.790513834	0.06249121217
27	17	15	2	13.59	10.16	5.08	23.75	8	0.3368421053	0.02836565097
28	15	12	3	10.7	11.38	3.793333333	22.08	12	0.5434782609	0.04922810334
29	12	7	5	10.65	14.82	2.964	25.47	20	0.7852375344	0.06165979854
30	7	2	5	9.88	16.29	3.258	26.17	20	0.7642338556	0.0584053386
Average				10.9862069	12.05241379	6.703390805	23.03862069		0.4533388666	0.0338349379
Std Dev				1.012568918	7.802248145	2.726449065	7.788185065		0.46763585	0.02108714385
Mean w Confidance				10.98 +/- 0.368	12.05 +/- 2.84	6.70 +/- 0.99	23.04 +/- 2.84		0.45 +/- 0.171	
Max				27	4	17	30		29	

5.0 REFLECTION

This section explains both advantages of our design of the project, which we liked, and areas for improvement in the future.

5.1 Advantages of our design

Our design allows for easily inserting new instructions for the elevator by simply following the format of the input file and typing the instructions in. Furthermore, our configuration file allows for easily modifying the number of elevators in the system, as well as the number of floors that are being modelled. Our design supports high cohesion and low coupling, because we have fully separated the elevator, floor, and scheduler components of the system.

Yet another of the advantages of our design is the inclusion of separate classes which are threads in order to check whether a message has been sent to the "master" class to which they belong. The advantage is that these threads themselves can block, instead of having the master class itself block. As an example, having the Scheduler class block in receiving communication would cause chaos in the system, so instead, the subordinate thread that we have will check whether the message has arrived, and the subordinate thread blocks. This way, it is easy enough for the master class to subsequently unblock the thread.

5.2 Areas for improvement

One area that our design could improve in is utilizing the configuration file more effectively. For instance, currently, the number of lines (i.e. the number of requests for all elevators) is something which must be hard-coded in our sources code, but we could have included that in our configuration file. Furthermore, the information for port numbers and IP addresses of all subsystems when using UDP is something that is also currently hard-coded, but in the future, it would be possible to easily incorporate these values in the configuration file for easily running the program in different environments. In addition, currently, our design does not account for the time it takes to service a request, and as an improvement we could have alongside the currently present print statements to have print statements showing what time an order request comes in and having a small delay for the time it takes to actually service the request.

Additionally, our current system design does not account for the environmental factors in the problem domain, such as the behavioural aspects of the lights and sensors of the elevators and floors. Another way to improve our design is to make our system prepared to handle any system crashes or failures such as the elevator door not opening or closing completely, or any loss of UDP packets amongst the three subsystems.