SYSC 3303 Real Time Concurrent Systems

Final Report

Project - Elevator Control System and Simulator

Lab Section: A3

Group Number: G9

Due Date

April 7th, 2020

Group Members

Team Member Name	Student Number
Abdul-Rahmaan Rufai	101088271
Ediomoabasi Emah	101082797
Hilaire Djani	101087724
Lois Aleghe	101081652
Mohammed-Yasir Suara	101088810

Table of Contents

1.0 BREAKDOWN OF RESPONSIBILITIES

1.1 Iteration 1

Group Members	Responsibilities
Lois Aleghe	FloorData class, Test Suite (AllTests.java), Test Case (SchedulerTest.java)
Hilaire Djani	Scheduler class, FloorSubsystem class, Test Case (SchedulerTest.java)
Ediomoabasi Emah	ElevatorSubsystem class, Test Case (FloorDataTest.java)
Abdul-Rahmaan Rufai	FloorSubsystem class, Test Case (FloorDataTest.java)
Mohammed-Yasir Suara	README file, UML diagrams, Group Roles, Direction class

1.2 Iteration 2

Group Members	Responsibilities
Lois Aleghe	README file, UML diagrams, Group Roles, ElevatorSubsystem class, Floor class
Hilaire Djani	Elevator class, FloorQueue class, Scheduler class, ElevatorSubsystem class, Test Case (ElevatorTest.java)
Ediomoabasi Emah	Elevator class, FloorQueue class, Test Case (ElevatorTest.java), Test Case (FloorDataTest.java)
Abdul-Rahmaan Rufai	Scheduler class, Test Case (FloorDataTest.java)

Mohammed-Yasir Suara	Elevator class, Direction class, State Machine diagram, Test Case (SchedulerTest.java)

1.3 Iteration 3

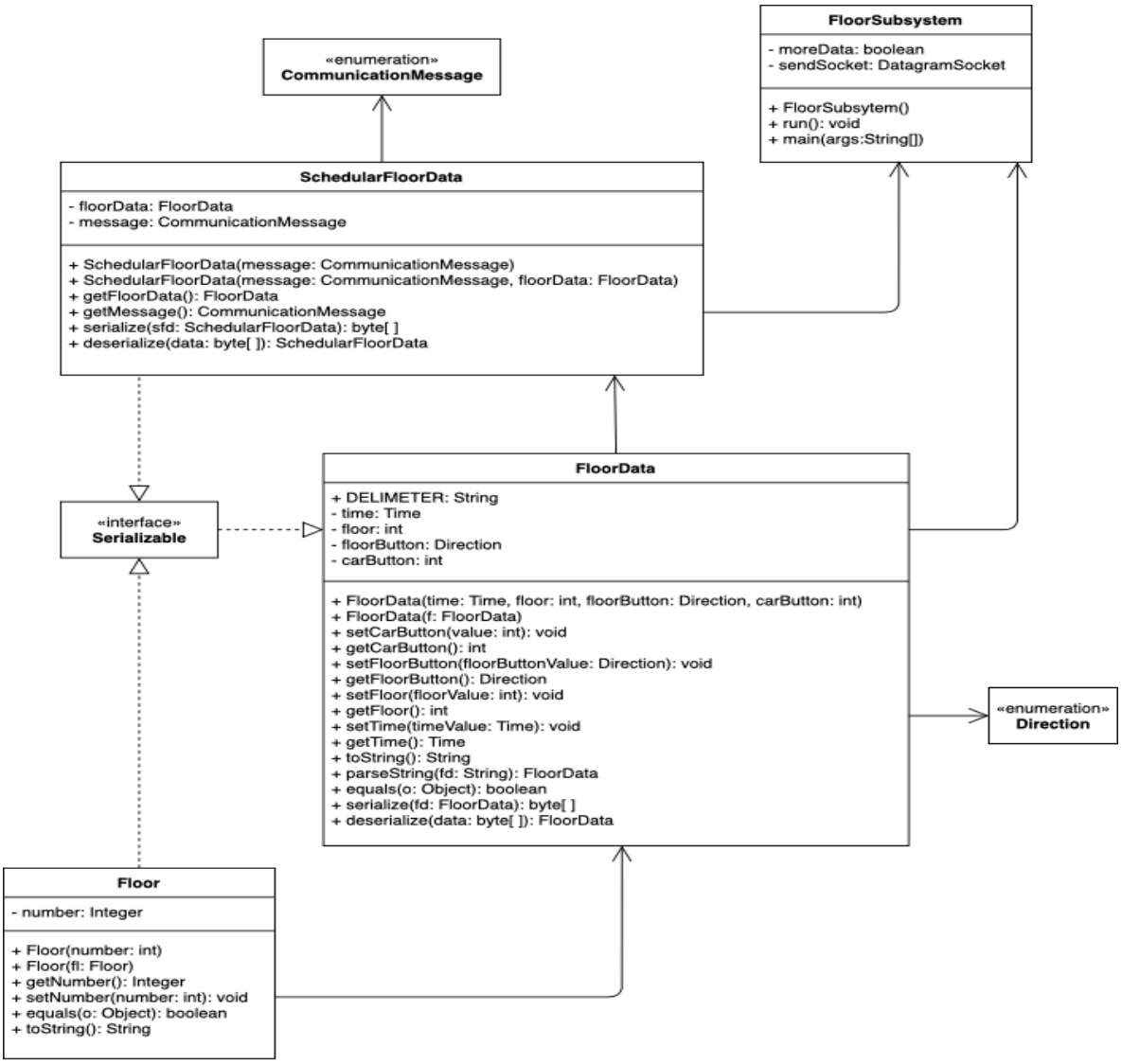
Group Members	**Responsibilities**
Lois Aleghe	Scheduler class, ElevatorData class
Hilaire Djani	Scheduler class, ElevatorSubsystem class, SchedularElevatorData class, SchedularFloorData class, CommunicationMessage class, FloorData class
Ediomoabasi Emah	ElevatorSubsystem class, README file, UML diagrams, Group Roles
Abdul-Rahmaan Rufai	Scheduler class
Mohammed-Yasir Suara	Elevator class, FloorSubsystem class

2.0 DIAGRAMS

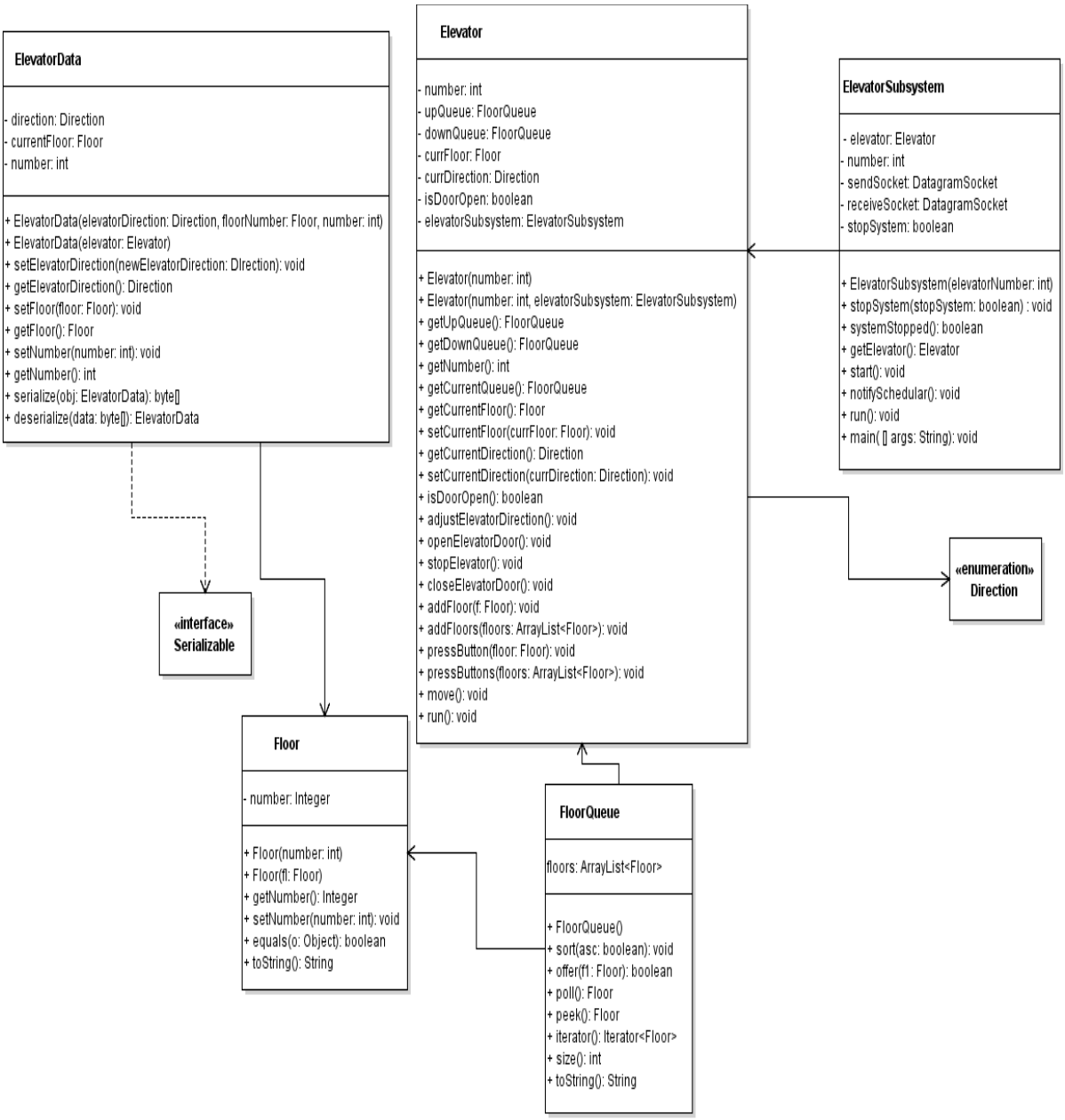
2.1 UML Class Diagram

A UML class diagram is a type of diagram that clearly describes the structure of a system by showing its classes, attributes, methods and relationship among objects.

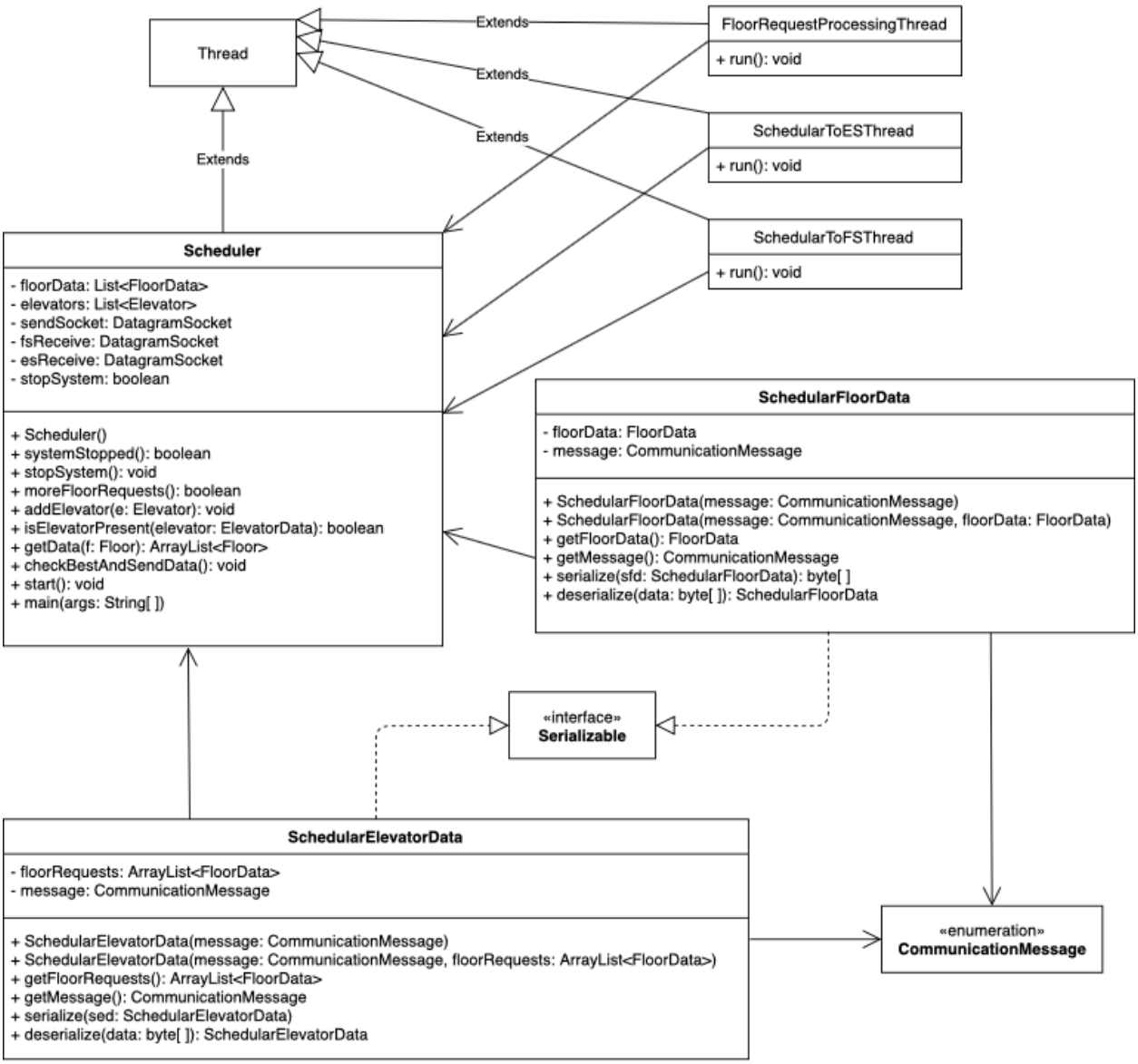
2.1.1 Floor Subsystem



2.1.2 Elevator Subsystem



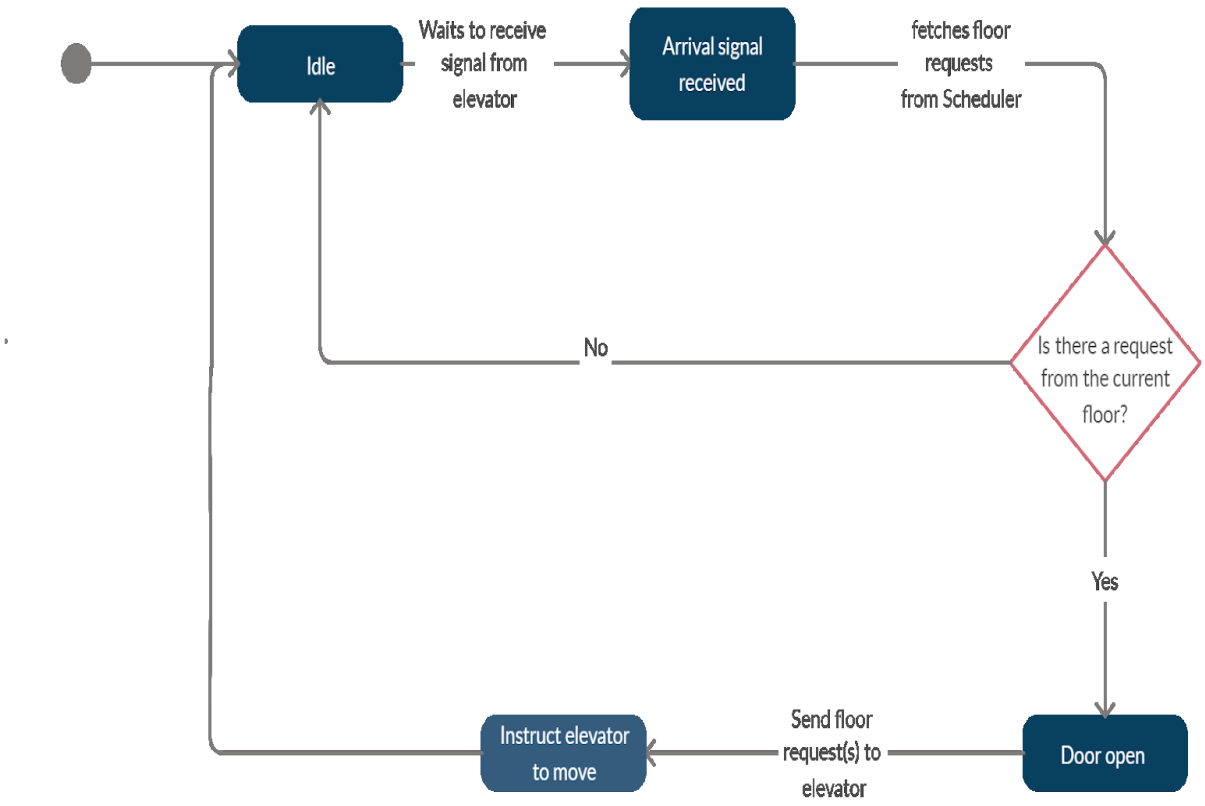
2.1.3 Scheduler Subsystem



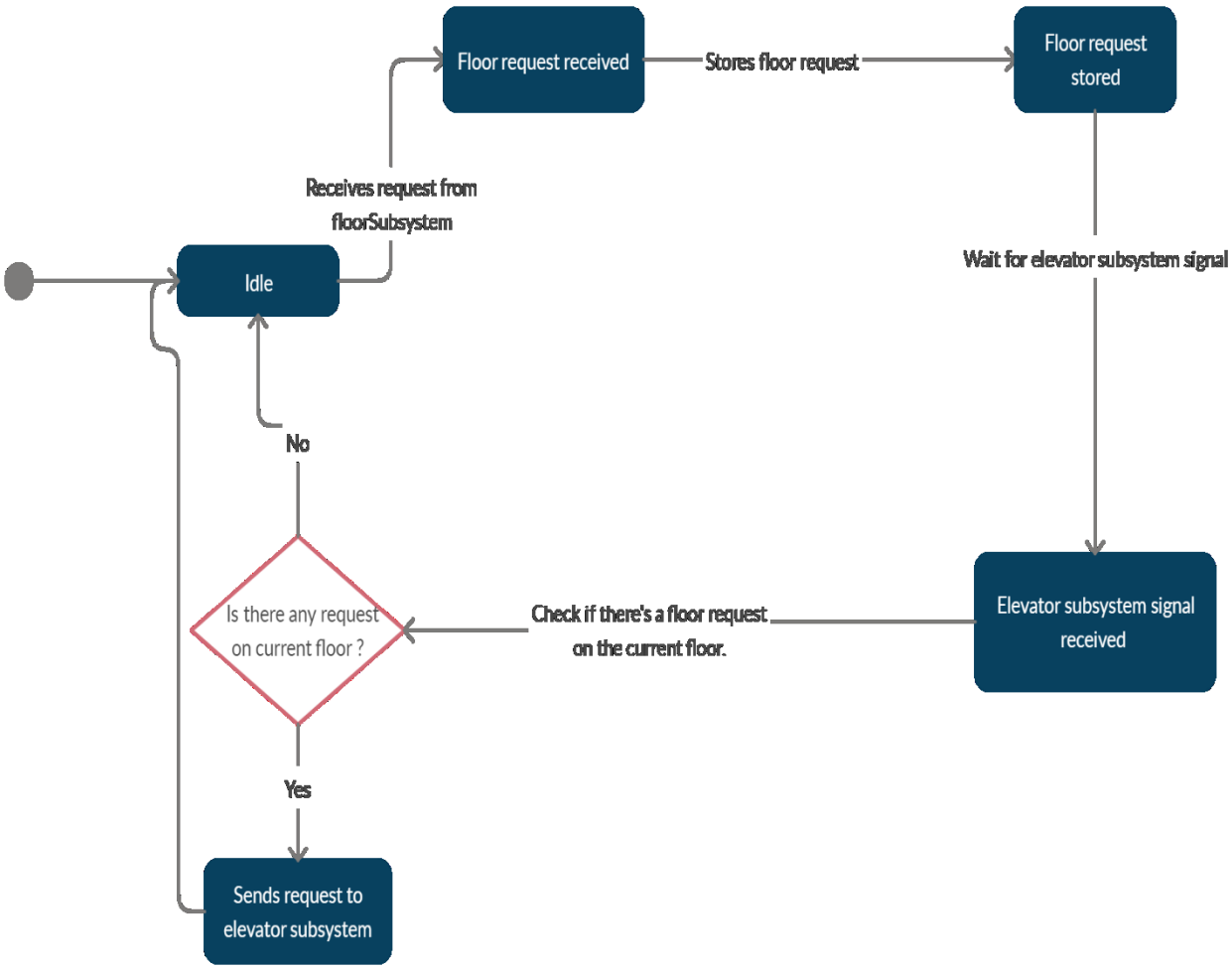
2.2 State Machine Diagram

A state machine diagram is a type of diagram used to model the behavior of a system through state transitions.

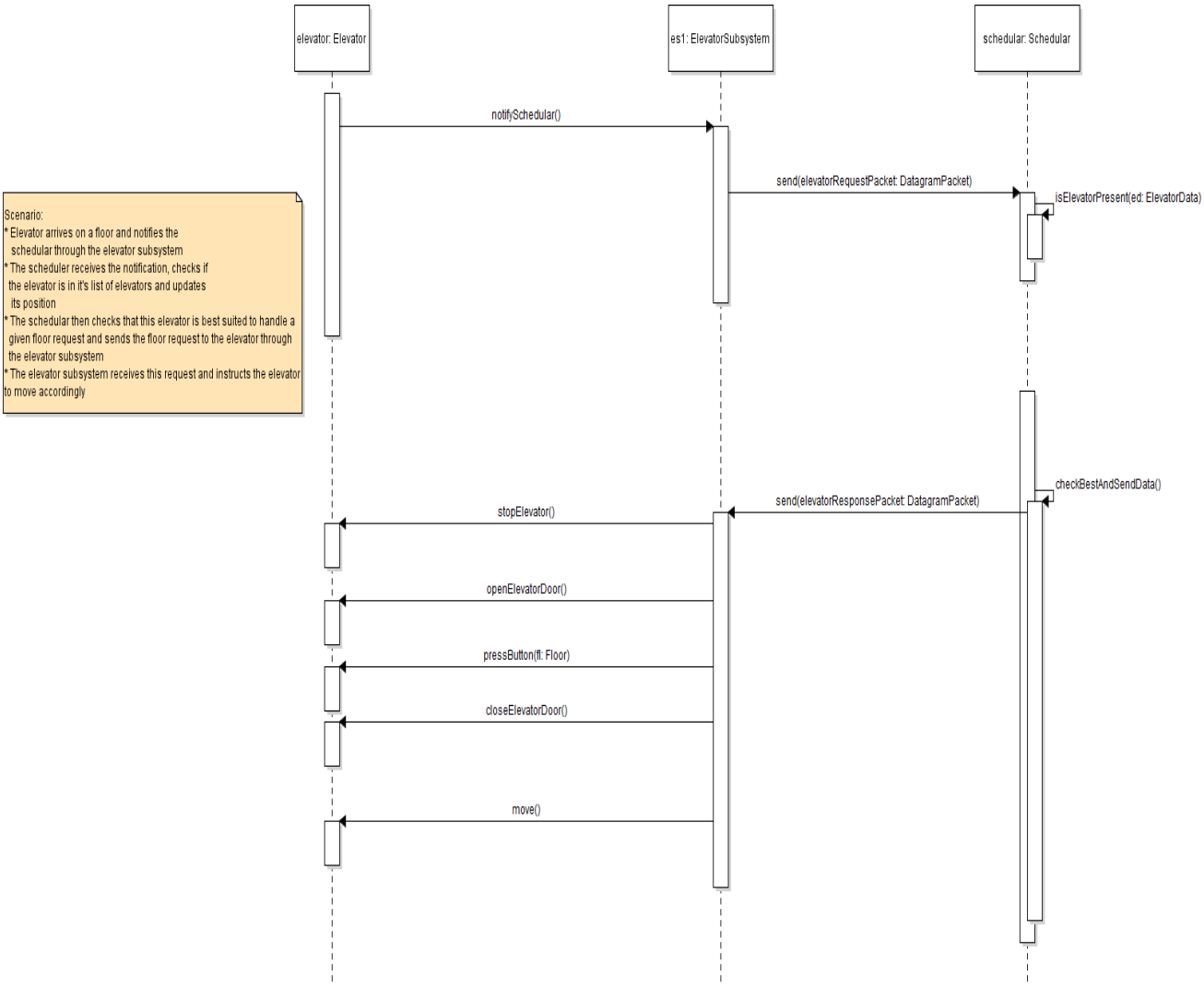
2.2.1 Elevator Subsystem



2.2.2 Scheduler Subsystem



2.3 Sequence Diagram



3.0 SETUP & TEST INSTRUCTIONS

3.1 Setup Instructions

In order to run the system on one computer:

- Click on the arrow beside the green “Run” button
- Select “Run Configurations...”
- Click on “Launch group” to create a new launch group
- Add the “Scheduler”, “FloorSubsystem” and “ElevatorSubsystem” to your launch group in that order
- Click “Apply”
- Finally, click “Run” to run the application

If you have difficulties following these steps, refer to the first part of the demonstration video. (Video link in appendix A).

3.2 Test Instructions

In order to run the test cases in Eclipse:

- Open the “elevatorSimulatorTests” package
- Open the “AllTests” class
- Click on the arrow beside the green “Run” button
- Select “Run as”, then “JUnit Test”

All tests must pass.

4.0 DESIGN REFLECTION

We are pleased with our system design in how the scheduler class receives, sorts and then satisfies requests from both the elevator and floor subsystems. The scheduler addresses elevator subsystem requests by choosing an elevator closest to the point of origin of the request and if it is going in the direction of that request. The was designed to consist of two threads to enforce separation of concerns. One thread created to deal with the floor subsystem requests and another to deal with the elevator subsystem requests. The elevator subsystem is responsible for controlling the behavior of the elevator, i.e. making it move in a specific direction or stopping it at specific floors. We designed our system such that for every instance of an elevator an instance of an elevator subsystem is created to control that elevator making it a one-to-one relationship. We approached it this way rather than having one elevator subsystem for all elevators in order to improve efficiency and increase concurrency in our system.

If given more time we would revise the logic for our scheduler choosing a best-fit elevator to satisfy a request by taking more things into consideration such as the state of an elevator, if it was idle or currently in motion. Unfortunately, due to limitations and restrictions that arose with the outbreak of COVID-19 pandemic, we could not implement a GUI nor provide fault handling for our system.

APPENDIX A

1. GitHub repository link: https://github.com/loisaleghe/elevatorSimulator
2. Demonstration video link:

   https://mediaspace.carleton.ca/media/Kaltura+Capture+recording+-+April+6th+2020%2C+3A03A50+pm/1_dt0b2hwl
3. Team Members

Team Member Name	**Student Number**
Abdul-Rahmaan Rufai	101088271
Ediomoabasi Emah	101082797
Hilaire Djani	101087724
Lois Aleghe	101081652
Mohammed-Yasir Suara	101088810