# Final Report

Group 5:

Arman Kocharyan – 100888381

Katherine Nelson – 101012786

Andrew Dodge - 100938015

Mohamed Gahelrasoul – 101007118

Roman Kishinevsky – 101009733

Dr. Gregory Franks

SYSC 3303A – Real-Time Concurrent Systems

Wednesday, April 10th, 2019

# **Table of Contents**

[Title Page 1](#_Toc5627306)

[Table of Contents 2](#_Toc5627307)

[Breakdown of Responsibilities 3](#_Toc5627308)

Iteration 1……………………………………………………………………………………………3

Iteration 2…………………..………………………………...…………………………..…….……3

Iteration 3…………………..…………………………………...…………………………………...4

Iteration 4…………………..…………………………………..……………………………………4

Iteration 5………..……………………………………………..……………………………………4

[Diagrams 5](#_Toc5627309)

UML Class Diagram – Floor…………………………………………………………………………5

UML Class Diagram – Scheduler……………………………………………………………….……5

UML Class Diagram – Elevator………………………………………………………………….…...6

UML Class Diagram – Whole System..……………………………………………………….……...6

State Machine Diagram – Scheduler………………………………………………………….………7

Sequence Diagram……………………………………………………………………………………7

Timing Diagram – Scheduler…………………………………………………………………………8

[Instructions 9](#_Toc5627310)

Set Up……………………..………………………………………………………………….……….9

Run the GUI…………………………………………………………………………………………..9

Testing……………………..………………………………………………………………..…..…….9

[Measurement Results 10](#_Toc5627311)

[System Analysis 11](#_Toc5627312)

[Design Decisions 12](#_Toc5627313)

[Reflection 13](#_Toc5627314)

# **Breakdown of Responsibilities**

*Iteration 1*

|  |  |
| --- | --- |
| **Group Member** | **Responsibilities** |
| Arman Kocharyan | * Created the elevator subsystem |
| Katie Nelson | * Created the floor subsystem |
| Andrew Dodge | * Created the scheduler subsystem * UML class diagram |
| Mohamed Gahelrasoul | * Created the scheduler subsystem * State machine diagram for scheduler |
| Roman Kishinevsky | * Created the scheduler subsystem * State machine diagram for elevator |

*Iteration 2*

|  |  |
| --- | --- |
| **Group Member** | **Responsibilities** |
| Arman Kocharyan | * Test cases * Timing of elevator carts * Created cucumber tests and the logger * Elevator message (the default message class for all UDP messages) |
| Katie Nelson | * Floor subsystem arrival * Floor controller * Sensors * Implemented pickUpPerson, rideToFloor, and the event listener which opens a socket on a given port and listens until message is received |
| Andrew Dodge | * UML class diagram * Implemented startListen which starts a new thread/daemon that blocks and waits call |
| Mohamed Gahelrasoul | * Documentation * Timing diagrams * Worked on resetting the buttons, and elevator arrival |
| Roman Kishinevsky | * Requesting elevator up and down, with the up down buttons * Small interactions with the elevator |

*Iteration 3*

|  |  |
| --- | --- |
| **Group Member** | **Responsibilities** |
| Arman Kocharyan | * Worked on exceptions * Worked on testing |
| Katie Nelson | * Worked on exceptions * Worked on testing |
| Andrew Dodge | * Worked on timing * Worked on diagrams |
| Mohamed Gahelrasoul | * Worked on timing * Worked on diagrams |
| Roman Kishinevsky | * Worked on timing * Worked on diagrams |

*Iteration 4*

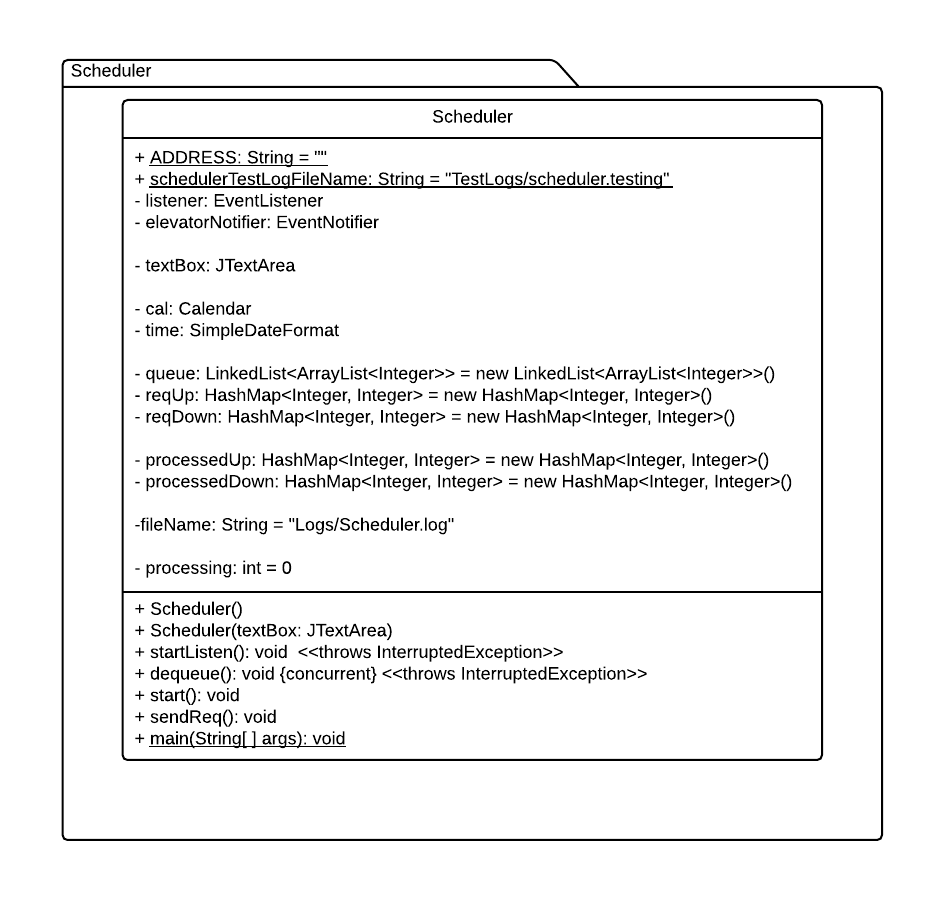
|  |  |
| --- | --- |
| **Group Member** | **Responsibilities** |
| Arman Kocharyan | * Got the scheduler working on a separate computer. * Calculated the mean and variance of the scheduler |
| Katie Nelson | * Worked on exceptions * Worked on testing |
| Andrew Dodge | * Got the scheduler working on a separate computer. * Calculated the mean and variance of the scheduler |
| Mohamed Gahelrasoul | * Worked on timing * Worked on diagrams |
| Roman Kishinevsky | * Worked on timing * Worked on diagrams |

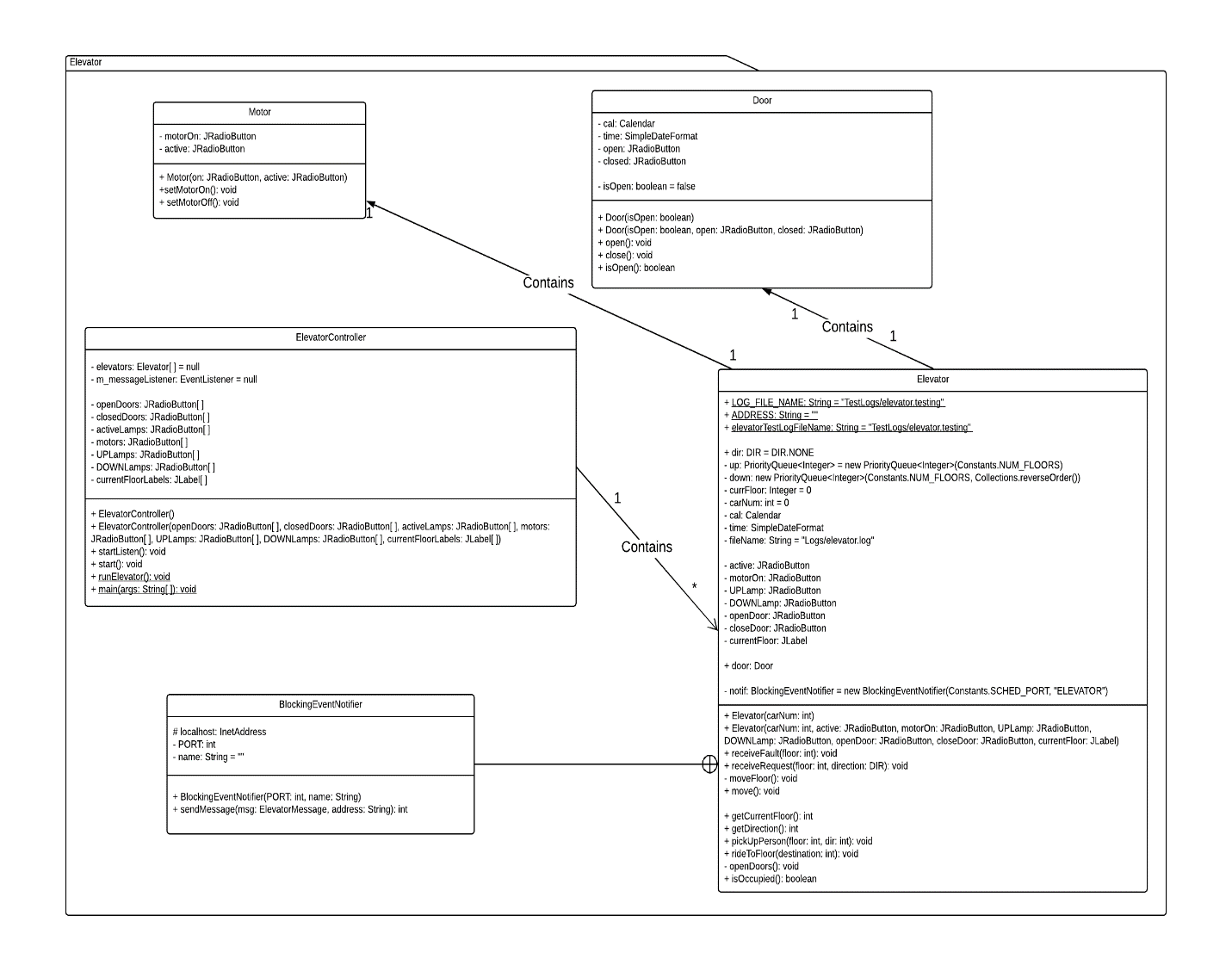
*Iteration 5*

|  |  |
| --- | --- |
| **Group Member** | **Responsibilities** |
| Arman Kocharyan | * GUI * Refactored code |
| Katie Nelson | * GUI * Refactoring code to accept multiple people |
| Andrew Dodge | * GUI * Created, wrote, and formatted the report * Updated UML diagrams |
| Mohamed Gahelrasoul | * GUI * Created sequence diagram * Timing diagram |
| Roman Kishinevsky | * GUI * State machine diagram |

# **Diagrams**

*UML Class Diagram – Floor*

*UML Class Diagram – Scheduler*

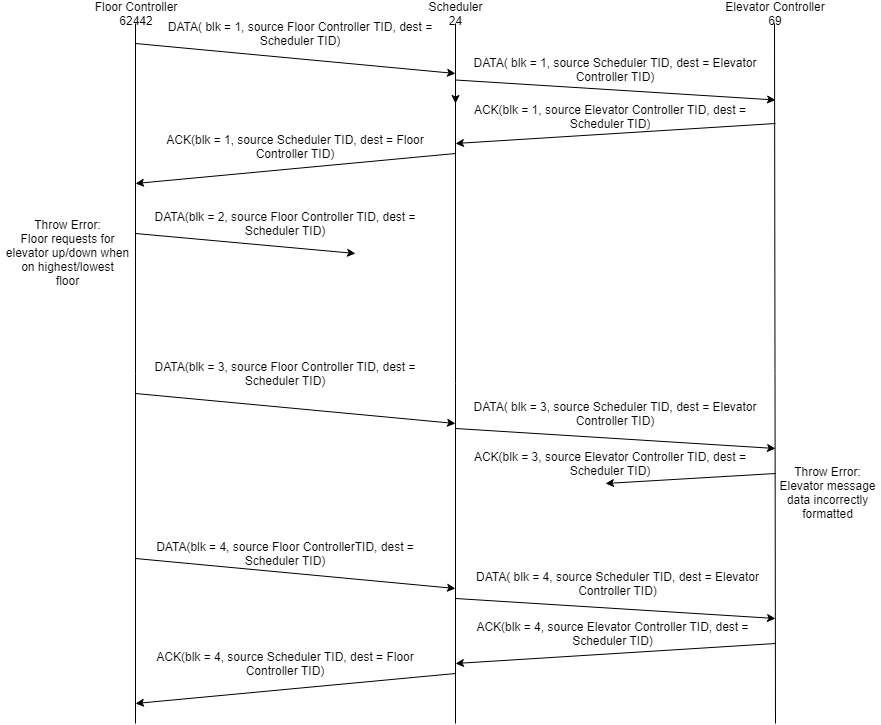
*UML Class Diagram – Elevator*

*UML Class Diagram – Whole System*

* Image was too large to properly display it in this report, it has been included as a separate file with the submission, titled ‘Elevator Project UML.png’

*State Machine Diagram – Scheduler*

*Sequence Diagram*

*Timing Diagram – Scheduler*

# **Instructions**

*Set Up*

To run the code:

* Open eclipse ide and import project as a maven project
* Open the “gui directory

1) Right click “MainView.java” file and click “run”

To run on multiple computers:

* Open eclipse ide and import project as a maven project on both machines
* Open the subsystems directory

1. In the core package run the IPGetter.java file on both machines
2. In the scheduler package open the Scheduler.java file on machine 1, set the public static final String ADDRESS = ""; to the IP address of machine 2, add it inside the ""
3. open the Floor.java file from floor package and Elevator.java file from the elevator package, and add machine 1's IP address to the public static final String ADDRESS = ""; variable in those files
4. Run Scheduler.java on machine 1
5. Run ElevatorController.java followed by FloorController.java on machine 2

To run the GUI:

* Open eclipse ide and import project as a maven project
* Open the subsystems directory

1. In the gui package run the ‘MainFrame.java’ file as a Java Application
2. Click start on the GUI window that pops up
3. Watch the program run.

*Testing*

* Open eclipse ide and import project as a maven project
* Open the subsystems directory

1. Run CucmonberRunner.java as a junit test

# **Measurement Results**

|  |  |  |  |
| --- | --- | --- | --- |
| Time Elevator Takes to Complete Trip in Seconds  (System.nanoTime was used to measure these values then was converted to seconds) | | | |
|  | Iteration 1 | Iteration 2 | Iteration 3 |
| Floor 2 to 3 | 3.018 | 3.016 | 3.020 |
| Floor 0 to 4 | 12.016 | 12.019 | 12.021 |
| Floor 3 to 2 | 3.017 | 3.016 | 3.018 |
| Floor 1 to 7 | 18.018 | 18.018 | 18.023 |

|  |  |  |
| --- | --- | --- |
| Time Elevator Takes to Move One Floor in Seconds | | |
| Iteration 1 | Iteration 2 | Iteration 3 |
| 3.018 | 3.016 | 3.020 |
| 3.004 | 3.004 | 3.005 |
| 3.017 | 3.016 | 3.018 |
| 3.003 | 3.003 | 3.003 |

*Total Mean:* 3.011s

*Total Variance*: 0.055s

# **System Analysis**

# 

# Our system has three processes that run concurrently: Scheduler, Elevator Controller, and Floor Controller. The system is based on a User Data Protocol communication protocol where the processes interact with each other via datagram sockets and packets. The role of the scheduler is to schedule elevators for floor requests. Where as the roles of the Elevator and Floor controller processes is to create and hold instances of respective elevator and floor objects as well as listen for any messages coming from the scheduler.

# In order to analyze our system, we tried to measure the execution times of our three processes, while running each process on a separate computer. Unfortunately, since our processes were running on latest intel processors our execution time was very predictable as they came up exactly as we timed them in our program (please check measurement result of the report). Therefore, we decided to generate our own data to demonstrate an elevator system that won’t miss any deadlines.

# We know that our system has three processes therefore:

= 0.779763

# Now we must choose the execution times and the period for our three processes. We know that the scheduler has the highest priority since it is responsible for the inter-communication therefore it must have the lowest period. We will select the elevator controller to have the second highest priority and the floor controller to have third. We know that the floor controller would have the highest execution time since it has over 20 floors. Elevator controller would have the second since it has 4 or more elevators and scheduler would have 1, since there is only one instance of a scheduler. There fore we came up with the following table:

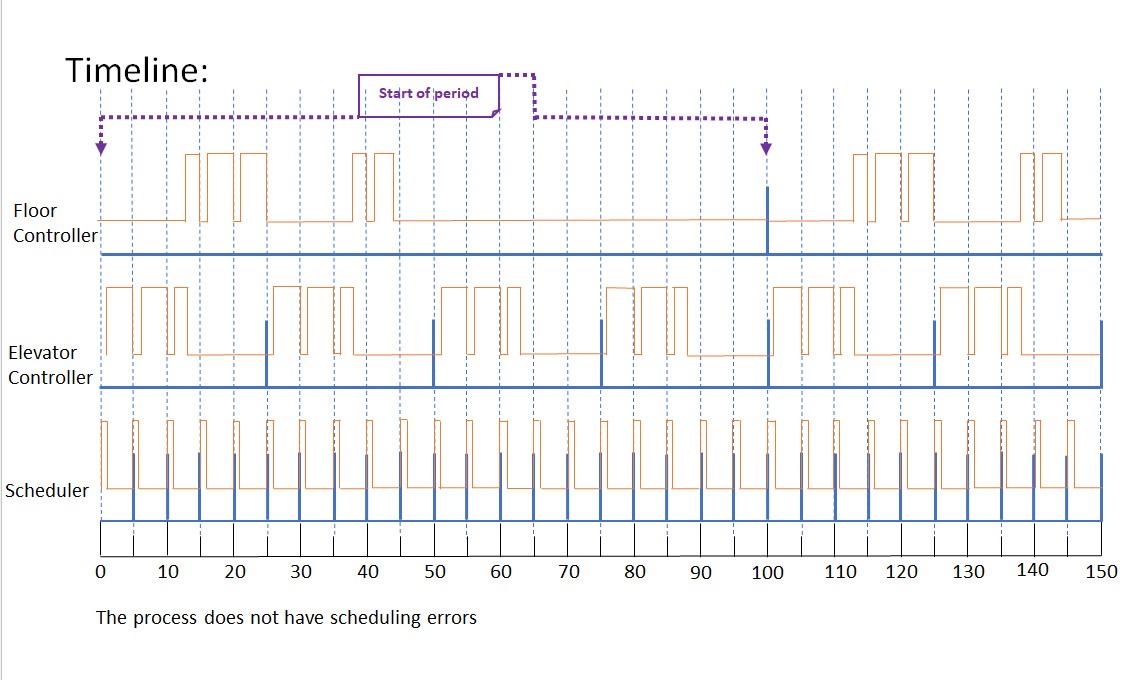
|  |  |  |
| --- | --- | --- |
| Process | Period | Execution Time (C) |
| Floor-Controller | 100 | 15 |
| Elevator-Controller | 25 | 10 |
| Scheduler | 5 | 1 |

Rate Monotonic Analysis:

LCM (Least Common Multiple):   
GCD (Greatest Common Denominator): 5

→ → →

# **0.75 < 0779763 Therefore RMA won’t miss the deadline**



# **Design Decisions**

# Part of our system design was to allow for our schedular to dynamically remove elevator requests from our queue of floors awaiting a ride. As soon as a car reaches a floor, they will stop, and for any passengers awaiting a ride in the same direction, the floor will notify the elevator, at which point the passenger will board the elevator and the request will be removed from the queue. This allows us to easily incorporate multiple ride requests travelling in the same direction, reducing the time and energy the system takes to fulfill ride requests. Another component allowed for each elevator to have a queue of floors to visit, given to it by the scheduler. On top of that, if one car picks up a passenger that another car had pending in its queue, then the request will be dynamically removed from that car so as not to create any disparities in the system.

# **Reflection**

Our group quite enjoyed working on the project, we found it was an interesting experience. It allowed us to program and design a system using technologies that we had never had the skills to use before. As well as gave us the knowledge to attempt project like this in the future. When it comes to elements of the design that we enjoyed it mainly had to do with the code base as opposed to the GUI. We enjoyed how we were able to create a system that was compact and visually appealing for the grandeur of its features. Furthermore, we found that we created a system with code that was laid out nicely and was rather simple to follow if you knew base computer science knowledge. All in all, we were content with the code base however, it was the GUI and small extra elements that our group had an issue with in the end. The parts we believed should be redone if we had the time mostly relate to the GUI, there are some elements that do not stem from the graphics component of the project though they are minimal. Speaking of these minimal changes first, the main complaint we found came from us barely refactoring our code base. While it is true that it is an easy to follow code base, and simple if you understand that language we found that there are a number of redundant code elements within the system. With proper refactoring we would have been able to remove these code snippets and create a system we could be fully proud of. Furthermore, our only other code complaint stems from the fact that we barely documented any of our code, which is why it was stated that the code was easy to read if you had prior knowledge. The lack of comments makes it confusing for someone who is reading the information for the first time and given more time we would have liked to fully document the entirety of the codebase. Finally, our main complaint with the system, and the element we wish we could re do the most is the GUI. Due to outside factors we left the design of this portion of the system very near to the submission deadline. We still managed to finish everything on time, but it was not as polished as we would have liked it to be. We would have wanted to make it more fleshed out and showing the entire span of options that have to do with an elevator, as well as potentially show some amusing interactions with the passengers. To end, we enjoyed working on the project and creating this elevator system. We are quite proud of the code we have designed as a group; however, we all have decided that our visual skills need a serious overhaul.