Individual Contributions

Iteration 1:

Andre Hazim: Elevator and Scheduler subsystem + testing
- ElevatorSimulator: Elevator.java, Scheduler.java, ArrivedElevatorMessage.java
- ElevatorSimulatorTest: SimulatorTest.java

Bardia Parmoun: Scheduler subsystem + testing
- ElevatorSimulator: Buffer.java, Scheduler.java, Message.java
- ElevatorSimulatorTest: ElevatorTest.java, FloorTest.java, MockScheduler.java, SchedulerTest.java, SimulatorTest.java

Guy Morgenshtern: Floor subsystem + testing
- ElevatorSimulator: Floor.java, Simulator.java, Message.java, KillMessage.java, RequestElevatorMessage.java
- ElevatorSimulatorTest: BufferTest.java

Kyra Lothrop: Elevator subsystem + testing
- ElevatorSimulator: Buffer.java, Elevator.java, ArrivedElevatorMessage.java, KillMessage.java
- ElevatorSimulatorTest: BufferTest.java

Sarah Chow: Floor subsystem + testing
- ElevatorSimulator: Floor.java, Simulator.java, RequestElevatorMessage.java
- ElevatorSimulatorTest: ElevatorTest.java, FloorTest.java, MockScheduler.java, SchedulerTest.java

Setup Instructions

1. Unzip the submission file.
2. Navigate the eclipse IDE.
3. Navigate the File menu.
4. Open the "Open Project from File System.." option.
5. Select the root folder for the project "ElevatorSimulator".
6. Click the "Finish button".
7. Select the project folder.
8. Once the project folder is open navigate to the "./src/ElevatorSimulator/Simulator.java" file to run the main file and start the program.

Test Instructions

The ElevatorSimulator.Test package has been dedicated to testing. The testing framework that was used in this project is JUnit 5.

For each subsystem there is a dedicated test class to test them separately.
- FloorTest: testing the floor subsystem alone.
- ElevatorTest: testing the elevator subsystem alone.
- SchedulerTest: testing the scheduler subsystem alone.
- BufferTest: testing the blocking message queue.
- SimulatorTest: testing the simulator system.

To help isolate the floor and elevator subsystems from each other, the scheduler subsystem was mocked and labeled as "MockScheduler".
This allows the scheduler to behave synchronously and avoid having the any of the tests be blocked waiting for the response of the other subsystem.

Navigate to any of these class and run them as a JUnit test to confirm that the system is working as expected.