# Elevator-SYSC3303

## Group Members:

Kamran Sagheir
Omar Azam
Abdulla Al-wazzan
Tiantian Lin
Ruixuan Ni

## Introduction

This project is a simulation of elevator subsystem

## Instrucitons

TO IMPORT THE PROJECT INTO ECLIPSE

1) Upzip the package
2) Open Eclipse and do following:
   ➔ Import the project into Eclipse (General -> Existing Projects into Workspace).
      ➔ Inside the project:"Elevator-SYSC3303":

      Run Scheduler.java (located src > scheduler > scheduler.java)

      Run ElevatorSubsystem.java (located src > elevator > elevatorSubsystem.java)

      Run FloorSubsystem.java (located src > client > floorSubsystem.java)

**Main .java files:**

Three files are needed to run the elevator system.

- ElevatorSubsystem.java: It will instantiate all elevators in separate threads as defined in the config.xml file. Each elevator thread waits for an event from the Scheduler to trigger an action.
- FloorSubsystem.java: This will instantiate all floors in separate threads as defined in the config.xml file. Then the requests.txt file is parsed, each request defined in this file is sent to the corresponding floor (the main method controls the timing of each request such that each request is sent relative in time to the preceding request). When each floor receives a trip request from the main() method, it sends this to the Scheduler. This simulates a trip request coming from each floor.
- Scheduler.java: When run from main(), this will instantiate the scheduler as defined in the config.xml file. The scheduler will then wait to receive and process requests.

**Tests:**

All the tests are located under the "tests" package

## Diagrams:

The uml, sequence and State diagrams are located under the "docs" package

## Deliverables And Responsibilities

**Scheduler (Kamran Sagheir)**

- MakeTrip.java: deals with the trip requests made, contains current and destination location
- Monitor.java: used to update the current state of elevator and stores list of trips ‘information
- Scheduler.java: accepting requests and sending events and commands(requests) as responds
- Server.java: creates DatagramSocket server for sending and receiving the data and getting the data in the bytes format as well as printing out the details of the data packet received
- MonitoredEventTimer.java: class responsible for the

**Server (Kamran Sagheir)**

- Server.java: uses socket to receive and send packets on specific port

**Info (Kamran Sagheir)**

- Helper.java: transfer between requests and datagram packet
- MutInt.java
- Parser.java: Responsible for the parsing of the datagrampackets.
- Populator.java: This class is responsible for taking the data from the datagrampacket and populating the appropriate fields in the appropriate systems.

**Enums (Sayyid Kamran Sagheir)**

SystemEnumTypes.java: a collection of enums used to define the states of lamps, requests, directions, scheduler events, and door.

**Requests (Kamran Sagheir, Tiantian Lin, Ruixuan Ni)**

- DirectionLampRequest.java
- ElevatorArrivalRequest.java
- ElevatorDestinationRequest.java
- ElevatorDoorRequest.java
- ElevatorLampRequest.java
- ElevatorMotorRequest.java
- ElevatorWaitRequest.java
- FloorButtonRequest.java
- FloorLampRequest.java
- LampRequest.java
- Request.java: This class and the other request classes are used to transmit information between threads

**Elevator (Tiantian Lin)**

- ElevatorEvents.java: an interface for subsystems
- ElevatorState.java: used to store the states of elevator, including moving, direction, floor, door and lamps

- ElevatorSubsystem.java: used to simulate the moving of elevator
- ElevatorSystemConfiguration.java: configure elevator subsystems by reading input xml files

**Floor (Ruixuan Ni)**

- FloorSubsystem.java: responsible for reading input requirements files, sending and receiving requests to/from server

**Unit tests (Ruixuan Ni, Tiantian Lin, Omar Azam and**

**Kamran Sagheir)**

**Sequence Diagram and UML (Tiantian, Abdulla Al-**

**Wazzan and Kamran Sagheir)**

**State Diagram (Ruixuan Ni)**

**Reflection on concurrency control at the scheduler:**

In terms of concurrency control, as compared to the iteration 2 there is not much that needed to be changed as the systems were already designed with the intention of making the systems communicate using the UDP. Hence, all the communication was already happening by means of datagram packets and hence, the system was already capable of coordinating the movements such that the wait time is minimized. Further enhancements were made to the already existing scheduler such that:

If an elevator is going up and a button is pressed, in the upward direction, the elevator will stop to carry the person. The estimated time required to pickup from the current floor is calculated as:

- In case elevator is in STAY state: (number of floors) x (average time between floors).
- In case of an re-route trip: = [(number of floors) x (average time between floors)] + [(number he return is 0 of stops) x (average time per stop)].
- If the elevator can't facilitate the request, it notifies because it will cause a delay to pick up, it notifies the other elevators.

Similarly, for the concurrency control purposes, method are present that are responsible for checking if the elevator has the destination button pressed or not, and which elevator is it for which the button is pressed, what's the highest floor, which elevator is at which floor(managed by the monitor class), what is the lowest floor, addition of the trips to the queue in case elevator already has a job to finish so that it may finish that job first and then move on to the other one (managed by the monitor class), once the elevator has completed a job, assigning the next job form the queue to that elevator, if the elevator trip needs to be rescheduled, doing so on the basis of the above stated formulas, once a destination is received, removal of floor for the destination queue, getting current elevator states and setting of the states etc. Hence, for each elevator, a record is kept and updated regularly by the scheduler including but not limited to, the elevator which is serving the request, the floor of the request, and the destination.