< Group2 > Final report

201133208 이유호
201133204 유현우
201233313 송치윤
201233334 조남진

Why did we make the scrooge elevator ?

Scrooge elevator

It decreases the waiting time of user and reduces the efficient movement of elevator like scrooge saves the money by trying to not use the non-productive things.

1. The problem of normal elevator.

- Normally, user enters hoped floor in the elevator not outer place. In this case, elevator is hard to make optimal scheduling.

- Users have a trouble with choosing lowest waiting time in a aspects of selecting elevator.

- In many cases, one user monopolizes the two or three elevators even though he or she choices just one elevator.

2. The things we try to solve.

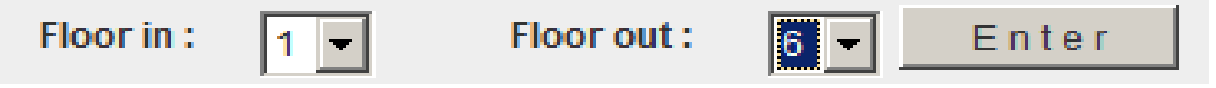
- set the optimal elevator by receiving the information of floor in and out in the inner elevator. (In this case, it prevent monopolization from user.)

- By applying this logic, decrease the movement of elevator and reduce the waiting time of users.

- As a result, changed elevators solve the user's complaint because scheduler set the fair elevator to users.

3. The way to make the scrooge elevator.

(1) Scheduler receives the information of in and out floor.



through the input value, elevators **executes the scheduling.**
(**We made the expression**)

Consideration

1) **Distance** between elevators and people.

2) The **state of elevator.**

3) **The number of waiting people and people in the elevator.**

- **How did we make the expression.**

  - **compare :** | elevator Location – person Location |

  - **usability :** consider the number of waiting people and inner people about each elevator and set usability according to each case (over 12, over 8 …)

  - **'calcul method' :** consider elevator state and person location, destination, then set 'result' according to each case.

```
int compare = Math.abs(gui.floor[input - 1] - gui.obele[i].location);

if (i == 3) {
    if (input < 5 && des < 5)
        compare = compare - 100;

}
if (i == 4) {
    if (input > 4 && des > 4)
        compare = compare - 100;
}
int totalwait = totalhf(gui.obele[i]);
int usability;
if (gui.obele[i].usernum + totalwait > 12)
    usability = 1000;
else if(gui.obele[i].usernum + totalwait > 8)
    usability = 600;
else if (gui.obele[i].usernum + totalwait > 6)
    usability = 300;
else if (gui.obele[i].usernum + totalwait > 4)
    usability = 200;
else if(gui.obele[i].usernum + totalwait > 2)
    usability = 130;
else
    usability =  gui.obele[i].usernum + totalwait;
compare = this.calcul(hereState, wantState, compare, usability, t);
```

```
public int calcul(int here, int want, int Distance, int numOfUser, int eleState) {
    int result;
    int D = Distance;

    if (here == eleState || eleState == 0) {
        if (want == eleState)
            result = 95 + D + numOfUser;
        else {
            result = 100 + D + numOfUser;
        }
    } else
        result = 250 + D + numOfUser;

    return result;

}
```

(2) All elevators are **synchronized to the expected floor** when each elevator's state is idle.

(we use database connection and socket to make that part.)

```
mysql> desc elevator;
+-------+---------------+------+-----+---------+-------+
| Field | Type          | Null | Key | Default | Extra |
+-------+---------------+------+-----+---------+-------+
| name  | int(11)       | NO   | PRI | NULL    |       |
| floor | decimal(8,2)  | YES  |     | NULL    |       |
+-------+---------------+------+-----+---------+-------+
```

We save and update each elevator's expected floor in mysql table.

- **The way to calculate the expected floors.**

Estimated RTT(n)= (1-a) * Estimated RTT(n-1) +a * sampleRTT.

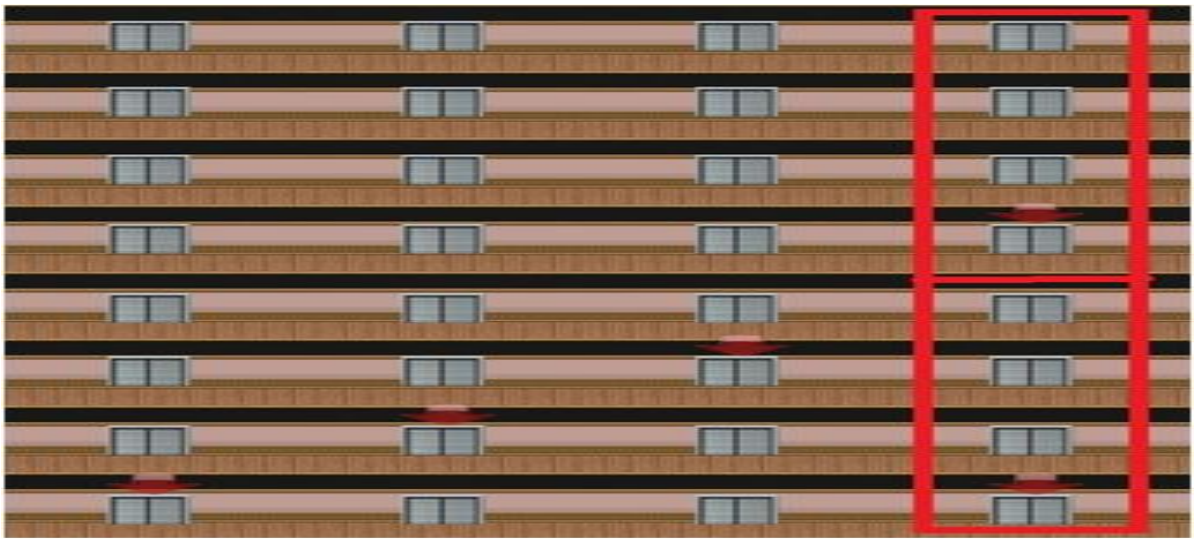
(We referenced the network class.

<calculate estimated round trip time.>)

- **The accessibility of database**

Through socket program, client request each elevator's expected floor to server which have database of elevator.

(3) One line is divided to two contents.



- In case of user wants to move the close floor, **divided elevators decrease the inefficient movement and people's waiting time**

- Our algorithm mostly assigns divided elevator to person who wants to go close floor.

4. NORMAL elevator vs SCROOGE elevator

	Normal		Scrooge	
Input	Time	Movement	Time	Movement
FileInput 1	25.18	3730	21.5	3216
FileInput 2	25.32	3752	20.25	3007
FileInput 3	18.36	2719	15.67	2362
FileInput 4	40	5925	34.15	5058
FileInput 5	27.5	4073	23.48	3477
FileInput 6	24.09	3568	20.56	3045
FileInput 7	29.72	4402	25.37	3757
FileInput 8	24.93	3692	21.28	3151
FileInput 9	132.3	19598	112.9	16724
FileInput 10	102.2	15139	87.26	12925

- We test different sample 10 times in normal and Scrooge elevator.

We found that Scooge elevator reduce time and all elevator's movement.

5. The way to execute our elevator program

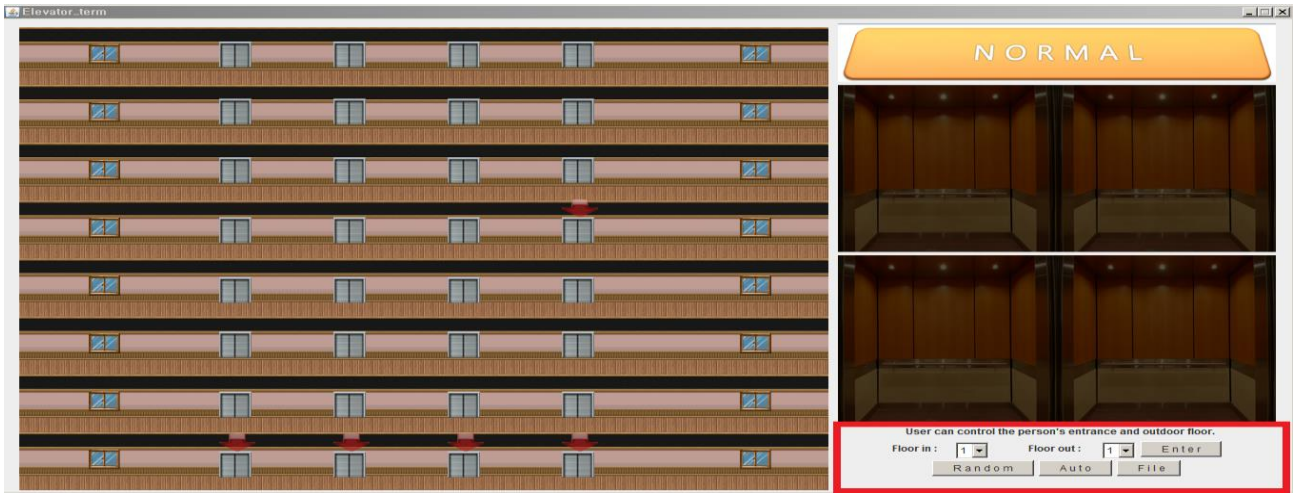
1) import mysql file and type your 'mysql' password in Setting.java

(line 27, 'input password')

2) execute Server.java

3) execute scheduler.java (connect server to use socket)

6. The way to control elevator



1) direct input

Set Floor in, Floor out and click Enter, then a person appears and get in selected elevator

2) Random input

If Random button is clicked then, a person appears on random floor and get in selected elevator

3) auto input and File input

auto input : random input are inserted continuously about 10 seconds
File input : read text file and execute

4) Remote Controller (use Socket program)

If you set IP(or 'localhost') in 'remoteControl.java' in other host, then we can control elevators to use this controller.