**<Elevator\_ Algorithm Team Project Final Report>**

-4조) 유영준, 조경철, 한주연-

**Our Elevator Structure**

ElevatorNO\_2

ElevatorNO\_3

ElevatorNO\_1

ElevatorDistrubution3

ElevatorDistrubution2

ElevatorDistrubution1

ElevatorNo\_3Moving

ElevatorNo\_2Moving

ElevatorNo\_1Moving

ElevatorMovement2

ElevatorMovement1

ElevatorMovement3

**TotalElevator**

**Using M C F**:

Only our team‘s Algorithm, M.C.F Algorithm was made for goal to put the elevator on the floor that used to the most to destination floor. When you get in elevator, case to get the least waiting time is that elevator is on my floor. This time, waiting time is 0. So our team concentrates this case and for making this situation frequently, our result is the most common floor. Reason that can make the least waiting time means that the floor has the most many people. So this means that frequency rate of using elevator is the highest rating. So using M.C.F, average waiting time can be reduced.

**Instruction Manual**

1. Click the left door & Walk Minions
2. Click the up & down button
3. Waiting Elevator
4. As many as the number of minions, Click the inner button

**class TotalElevator**

This class is for deciding which elevator to get on by considering the state of all elevators. First, consider about the weight of all elevators. If there is overweighed elevator, class will exclude the elevator. If all the elevators are in stop state, the nearest elevator will come.

If top button is pressed, consider about only the statement with 0, 1. If statement is 2, elevator is going up, but waiting time will be increased because elevator will be going down. If statement is 3 or 4, elevator is going down, so we don’t need to consider about it because of the same reason. If statement is 1 and current location is higher than location of pressed button, waiting time will be increased. . Find the elevators which statement is 1 and its’ current location is lower than location of pressed button, and choose one of them to get on.

If bottom button is pressed, consider about only the statement with 3. If statement is 1 or 2, elevator is going up, so waiting time will be increased. In this case, we don’t need to consider about it. If statement is 4, elevator is going down but will be going up soon, it doesn’t need to consider with the same reason. Find the elevators which statement is 3 and its’ current location is higher than location of pressed button, and choose one of them to get on.

**class ElevatorNo\_1,2,3**

It is divided into five states 0, 1, 2, 3, 4.

* Statement 0 : Stop state. After a certain period of time Elevator1 will move the most common floor and Elevator 2 will move second common floor.
* Statement 1 : Rising state. If pressed inner top button or top button on the outside.
* Statement 2 : Rising state. If pressed bottom button outside it will go down.
* Statement 3 : Down state. If pressed bottom button and inner bottom button.
* Statement 4 : Down state. If pressed bottom button or top button on the outside it will go up.

**class Elevatordistribution1,2,3**

Check the current status to decide statement.

* If elevator is going up, search pressed button from current floor to top floor.
* If up button or inner button were pressed, decide statement to 1.
* If bottom button was pressed, decide statement to 2.
* If elevator is going down, search pressed button from bottom floor to current floor.
* If bottom button or inner button were pressed, decide statement to 3.
* If up button was pressed, decide statement to 4.

**class** **ElevatorMovement1,2,3**

This class mainly has role of determining the elevator’s direction where it should go and what’s the goal floor. Also In current location, this class can judge what’s floor to press the up or down button. This class gets the class “ElevatorDistrubution” object and through this class, it can control the elevator’s movement.

First, in ‘if’ sentence, this class tells which floor to press the up buttons and the bottom buttons. And if the elevators are arrived at the goal location, the ‘for’ sentence breaks other ‘for’ sentence would be implemented. In this ‘for’ sentence, from current floor to over floor, it checks whether it pressed up button or down button and return value to be appropriate. And in third for loop, from current floor to under floor, it checks whether it pressed up button or down button. After all thing checked, call Elevatordistrubution.move() method.

**private void init():**

This class manages basic GUI form. Add panels to the base panel (centralPanel), and set the coordinate of all panels. Put the buttons on each designated panel and set the coordinate, too. By using for loops and button arrays, calculates each coordinate and set characteristic efficiently.

**class MyAction implements ActionListener:**

MyAction receives all button actions which are on shown panels. When it takes ele1[] button pressed, set innerButton1[] to true to notice innerButton inside elevator1 is pressed. The other buttons, ele2[] and ele2[] operates same as ele1[]. If ele1[] is pressed, it sets the destination of all minions on that floor which can get on elevator. Also when there is no minion on the elevator through variable countMinion, deactivates the innerButtons.

When top[] is pressed, change allTop[] to true to notice some topButton is pressed, and set icon of topButton to pressed shape. If bottom[] is pressed, it operates same as top[], just means bottom, not top.

If add[] is pressed, it takes floor where it pressed and use floor for parameter of AddMinion class. After setting new minion, run timer for moving minion in front of elevator by crossing the path. Timer set the coordinate by adding values periodically, so it looks like minions are running to the elevator. Every time which timer set the coordinate, it sets each minion’s coordinate, too. When minion moves 450 pixels, timer stops.

**class Minion:**

This class has many getters and setters to set each minion’s properties. When it is called, constructor set weight to 0, start floor and destination floor to -1 that means it hasn’t set yet, and number of elevator which it gets on to 0, and remove which means it is removed or not to 0. There is setter for determining image of minion randomly and getter for return that image. There is setter and getter for button of minion. Also, there are setters and getters for setting and returning the start floor and destination floor, and coordinate of minion. setLinkedIndex() is for individual index, and getLinkedIndex() is for returning individual index. class Minioncan know if this minion is removed or not by setting and getting value of remove.

**class ElevatorNo\_1, 2, 3Moving**

This class manages moving elevator to show through GUI. First it calls moveElevator1 to show each elevator’s movement. If it is matched, set innerButton[] to false and check statement which it is going up or down to set topButton or bottomButton is deactivated. Then changes icons of inner buttons, top and bottom buttons in normal mode, and shows open button is activated. After that, expresses elevator door is opened by change image icon of button and implements for loop for number of minions in order to count minions which get on or out of the elevator.

While counting minions which get on the elevator, inner buttons are activated for number of counted minions. When counting minions who get out of the elevator, run the timer for showing them move to the end of the path and disappear from the panel.

All minions who got on the elevator get out from the elevator, class changes image icon of the buttons to show that elevator door is closed. class ElevatorNo\_2Moving,and classElevatorNo\_3Movingoperates same as class ElevatorNo\_1Moving.