|  |
| --- |
| **Technical Manual** |
| **Smart Lift** |
| **CSCI 321 GA2** |

Table of Contents

[Objectives 3](#_Toc370980813)

[Core software and technology 3](#_Toc370980814)

[Software: 3](#_Toc370980815)

[Technology: 3](#_Toc370980816)

[Naming rules 3](#_Toc370980817)

[Class name: 3](#_Toc370980818)

[Function name: 3](#_Toc370980819)

[Variable name: 4](#_Toc370980820)

[Data dictionaries 4](#_Toc370980821)

[Data interactions with other programs 6](#_Toc370980822)

[Requirements 7](#_Toc370980823)

[Requirements Summary 7](#_Toc370980824)

[Functional Requirement 8](#_Toc370980825)

[F1 --- Log in 8](#_Toc370980826)

[F2 --- Administrator 8](#_Toc370980827)

[F3 --- Manager 9](#_Toc370980828)

[F4 --- User 9](#_Toc370980829)

[F5 --- Database 9](#_Toc370980830)

[F6 --- AI 9](#_Toc370980831)

[F7 --- Plan 10](#_Toc370980832)

[F8 --- Account 10](#_Toc370980833)

[F9 --- Lift 10](#_Toc370980834)

[F10 --- Simulation 11](#_Toc370980835)

[F11 --- History 11](#_Toc370980836)

[Non-Functional Requirement 12](#_Toc370980837)

[Scenarios 12](#_Toc370980838)

[Some pseudo code (Scenario 1 ~6) 19](#_Toc370980839)

[Class Structure and function command 27](#_Toc370980840)

[Function names and descriptions 30](#_Toc370980841)

# Objectives

The efficiency of lifts within a multi-storey building can be a significant factor in determining the efficiency of a business. Lift labs will help you determine the appropriate number and type of lifts required in your building based upon speed and carrying capacity. The lift management system module allows the setting of default floors and setting of rules governing the flow of lifts. The advanced AI module allows for lift management to be automated and adapt to appropriate conditions.

# Core software and technology

## Software:

Eclipse IDE

It's the standard because it's used by more people. And it benefitted from positive hype early on, while Netbeans suffered from a lingering bad reputation. The differences are less and less these days. Eclipse sure has more plugin.

Netbeans

It is convenient and easy to do the GUI of project.

## Technology:

Java SE

Java Platform, Standard Edition (Java SE) is a specification describing an abstract Java platform. It provides a foundation for building and deploying network-centric enterprise applications ranging from the PC desktop computer to the workgroup server. Java SE is implemented by the Java Software Development Kit (SDK).

Rule Execution Server can execute rule sets with 100% Java SE code. There are many Functions for pure Java SE execution, such as running batches or running rules from a JMS Provider or non-Java EE Enterprise Service Bus (ESB).

# Naming rules

## Class name:

Java file start with capital letter. For example Person, Control, Lift.

GUI file start with gui\_xxx.For example gui\_login, gui\_about.

## Function name:

If it is only one word, first letter is upper case. If more than one word, the first word is lower case, the second word first letter is upper case and the third word names same as second one. For example, Login(), checkInput(), changeUpButton()

## Variable name:

Start with class name’s first character with upper case, then a ‘\_’, and the first letter of first word, and then the second word with first upper case. For example, P\_weight, L\_cFloor. Otherwise, if the name has more than one word, each word is combined by ‘\_’, like stand\_by\_floor.

# Data dictionaries

* **Lift**

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| L\_cFLoor | Array(int) | It is a int array that used to store every lift current floor |
| l\_totalnumber | Int | It is a int variable that used to store lift total number in a building |
| total\_floor | Int | The total number of floor in a building is store here |
| d\_pnumber\_in\_lift | Int | Default max person In the lift |
| c\_pnumber\_in\_lift | Array(Int) | It is a array to stand for current people in every lift |
| l\_counter | Array (int) | This array is used as a counter for every lift, since open door, enter people, close door, lift moving from one floor to another, the time of them are different, so using this variable to count, for example, when the count is reach to 1, the lift is open the door and enter person, when it is 2, close the door |
| up\_target | Array(int) | When a lift get more than one tasks, in order to keep its moving is the most efficiency, lift need to move up to the target floor to do every task in same direction |
| down\_target | Array(int) | When a lift get more than one tasks, in order to keep its moving is the most efficiency, lift need to move down to the target floor to do every task in same direction |
| planlift\_d\_floor | Array(int) | If the lift is plan lift, after all task complete, it will move to the default floor |
| stand\_by\_person | Array(int)(int) | The stand by person in every floor are generated random, the stand\_by\_person[0] is stand for up person, stand\_by\_person[1] is stand for down person, and the second dimensional is stand for person number in every floor |
| copyTo | Array(int)(int) | It is a tem array to copy from stand\_by\_person that used to calculate the person move into to lift |
| floor\_want\_to\_go\_number | Array(int)(int) | When the person move into lift, program will generate every person’s target floor, the first dimensional is stand for lift number |
| tem\_floor\_want\_to\_go\_number | Array(int)(int) | Copy from floor\_want\_to\_go\_number, used in calculate person go out the lift |
| available\_floor | Array(int)(int) | When the lift running as Plan Mode, the first dimensional is stand for lift numer, the second dimensional is component with total floor number, for example, if lift one’s floor three is available, then available\_floor[0][2]=1,else available\_floor[0][2]=0 |
| enter\_lift\_person | Int | The number of person enter into lift |
| L\_direction | Array(char) | Usd to store lift situation,  A: stop first, then go up  B:stop first, then go down  C:open door, then stop  D:go down  E:go up, then stop  F:go down, then stop  H:close door, then go up  K:close door, then go down  S:stop  U:go up  V:go up, then B  W:go down then A  X:go up, then A  Y:go down, then B |
| L\_door | Array(Boolean) | Every lift’s door situation, if door is open, the value is true,else is false |

* **Plan**

|  |  |  |
| --- | --- | --- |
| planLift | Array(Boolean) | Check whether the lift is Plan lift |
| Planarray | ArrayList(plan) | It stores all the plans information |
| listSub | ArrayList(String) | Used to store a plan whole data |
| Name | String | It stores the name of plan |
| Year | String | It stores the year of start time or end time |
| Month | String | It stores the month of start time or end time |
| Day | String | It stores the day of start time or end time |
| Hour | String | It stores the hour of start time or end time |
| Min | String | It stores the min of start time or end time |
| liftnum | String | It stores the number of lift in the building |
| Defloor | String | It stores the default floor of planed lifts, while the lift is free, the lift goes to the default floor |
| date | String | It stores the date of start time or end time, includes year, month, day |
| Time | String | It stores the time of start time or end time, includes hour and min |
| Freq | String | It stores the frequency of plan used. It could be weekday or weekend |
| Afloor | String | It stores the available floors of planed lift, the lift will only go to the available floors |

# Data interactions with other programs

Data interaction in simulation is:

Currently, we use two files which includes building floor number, lift number, building name, plan name, default floor, planed lifts, available floors, start time, end time instead of database.

Every class read these file and set their current value when the file is needed. And update the file when the current status is changed.

# Requirements

## Requirements Summary

|  |  |  |
| --- | --- | --- |
| ID | Requirement | Priority |
| F1.1 | Log in the client | Base |
| F1.2 | Check Login Password | High |
| F2.1 | Create basic settings of building | Base |
| F2.2 | Manage account | High |
| F2.3 | Manage plan | High |
| F2.4 | Check simulation | High |
| F2.5 | Log out the system | Base |
| F2.6 | Check running history | High |
| F3.1 | Manage plan | High |
| F3.2 | Check simulation | High |
| F3.3 | Log out the system | Base |
| F3.4 | Check running history | High |
| F4.1 | Check simulation | High |
| F5.1 | Edit Data | Base |
| F5.2 | Read Data | Base |
| F5.3 | Delete Data | Base |
| F6.1 | Analyse summary data | Medium |
| F6.2 | AI manage plan | Medium |
| F7.1 | Read Plan | Base |
| F7.2 | Edit Plan | Base |
| F7.3 | Delete Plan | Base |
| F8.1 | Check account | Base |
| F8.2 | Create account | Base |
| F8.3 | Delete account | Base |
| F9.1 | Detecting over weight | Base |
| F9.2 | Record database | Base |
| F9.3 | Control the lift running direction | Base |
| F9.4 | Control the lift door state | Base |
| F9.5 | Update the number of people in lift | Base |
| F10.1 | Show lift | Base |
| F10.2 | Show stand by people | Base |
| F10.3 | Show current people in lift | Base |
| F10.4 | Show current lift door state | High |
| F10.5 | Show all stand by people | High |
| F10.6 | Show specific lift state | High |
| F10.7 | Update simulation | High |
| F11.1 | Create history | High |
| F11.2 | Read history | High |
| N1 | Easy to use | Medium |
| N2 | Easy to understand | Medium |
| N3 | **Every window connects well** | Medium |

## Functional Requirement

### F1 --- Log in

|  |  |  |
| --- | --- | --- |
| **ID:** F1.1 | **Requirement:** Log in the client | **Priority:** Base |
| The system will provide manager and administrators with the ability to login to the system via PC by providing an account ID and password.  Upon login, the system would display different menu for different type of users. | | |

|  |  |  |
| --- | --- | --- |
| **ID:** F1.2 | **Requirement:** Check Login Password | **Priority:** High |
| System will check the user ID and password with the user data file, depend on F1.1, F5.2,F8.1 | | |

### F2 --- Administrator

|  |  |  |
| --- | --- | --- |
| **ID:** F2.1 | **Requirement:** Create basic settings of building | **Priority:** Base |
| System will allow admin to create initial settings of lift, include quantity of lifts and storey of building, and building name which is used to identify the different building. Depend on F5.1 | | |

|  |  |  |
| --- | --- | --- |
| **ID:** F2.2 | **Requirement:** Manage account | **Priority:** High |
| System will allow admin to create/ delete account id and password. Depend on F5.1, F5.2, F5.3,F8.2, F8.3 | | |

|  |  |  |
| --- | --- | --- |
| **ID:** F2.3 | **Requirement:** Manage plan | **Priority:** High |
| System will allow admin to create/edit/delete plan. Depend on F5.1, F5.2, F5.3, F7.1, F7.2, F7.3 | | |

|  |  |  |
| --- | --- | --- |
| **ID:** F2.4 | **Requirement:** Check simulation | **Priority:** High |
| System will allow admin to check simulation. Depend on F10.1 F10.2 F10.3 F10.4 | | |

|  |  |  |
| --- | --- | --- |
| **ID:** F2.5 | **Requirement:** Log out the system | **Priority:** Base |
| System will allow admin to log out the system. | | |

|  |  |  |
| --- | --- | --- |
| **ID:** F2.6 | **Requirement:** Check running history | **Priority:** Base |
| System will allow admin to check running history of lifts. F11.2 | | |

### F3 --- Manager

|  |  |  |
| --- | --- | --- |
| **ID:** F3.1 | **Requirement:** Manage plan | **Priority:** High |
| System will allow managers to create/edit/delete plan. Depend on F5.1, F5.2, F5.3, F7.1, F7.2, F7.3 | | |

|  |  |  |
| --- | --- | --- |
| **ID:** F3.2 | **Requirement:** Check simulation | **Priority:** High |
| System will allow managers to check simulation. Depend on F10.1 F10.2 F10.3 F10.4 | | |

|  |  |  |
| --- | --- | --- |
| **ID:** F3.3 | **Requirement:** Log out the system | **Priority:** Base |
| System will allow managers to log out the system | | |

|  |  |  |
| --- | --- | --- |
| **ID:** F3.4 | **Requirement:** Check running history | **Priority:** High |
| System will allow managers to check running history of lifts. F11.2 | | |

### F4 --- User

|  |  |  |
| --- | --- | --- |
| ID: F4.1 | Requirement: Check simulation | Priority: High |
| System will allow users to check simulation with default building setting. Depend on F10.1 F10.2 F10.3 F10.4 | | |

### F5 --- Database

|  |  |  |
| --- | --- | --- |
| ID: F5.1 | Requirement: Edit Data | Priority: Base |
| The system must allow the database to be edited by permission users. | | |

|  |  |  |
| --- | --- | --- |
| ID: F5.2 | Requirement: Read Data | Priority: Base |
| The system must allow the permission users to read data from database. Database includes user ID, user password, building name, building floors, number of lifts, plan details. | | |

|  |  |  |
| --- | --- | --- |
| ID: F5.3 | Requirement: Delete Data | Priority: Base |
| The system must allow the database to be deleted by permission users. | | |

### F6 --- AI

|  |  |  |
| --- | --- | --- |
| ID: F6.1 | Requirement: Analyse summary data | Priority: Medium |
| The system will allow AI to read the database, and can edit database. Depend on F5.1, F5.2 | | |

|  |  |  |
| --- | --- | --- |
| ID: F6.2 | Requirement: AI manage plan | Priority: Medium |
| The system will allow AI create/edit/delete plan. Depends on F6.1. | | |

### F7 --- Plan

|  |  |  |
| --- | --- | --- |
| ID: F7.1 | Requirement: Read Plan | Priority: Base |
| The system must can read the plan. Depend on F5.2 | | |

|  |  |  |
| --- | --- | --- |
| ID: F7.2 | Requirement: Edit Plan | Priority: Base |
| The system must can edit the plan. Depend on F5.1 | | |

|  |  |  |
| --- | --- | --- |
| ID: F7.3 | Requirement: Delete Plan | Priority: Base |
| The system must can delete the plan. Depend on F5.3 | | |

### F8 --- Account

|  |  |  |
| --- | --- | --- |
| ID: F8.1 | Requirement: Check account | Priority: Base |
| The system must can read the account database. Depend on F5.2 | | |

|  |  |  |
| --- | --- | --- |
| ID: F8.2 | Requirement: Delete account | Priority: Base |
| The system must can delete account database. Depend on F5.3 | | |

|  |  |  |
| --- | --- | --- |
| ID: F8.3 | Requirement: Create account | Priority: Base |
| The system must can edit account database. Depend on F5.1 | | |

### F9 --- Lift

|  |  |  |
| --- | --- | --- |
| ID: F9.1 | Requirement: Detecting over weight | Priority: Base |
| System should allow the lift to detect whether the current weight is safe. If the weight is over the lift will not allow people get in the lift until the weight is under safe. | | |

|  |  |  |
| --- | --- | --- |
| ID: F9.2 | Requirement: Record database | Priority: Base |
| The system should allow the lift to record the daily running data. Depend on F5.1 | | |

|  |  |  |
| --- | --- | --- |
| ID: F9.3 | Requirement: Control the lift running direction | Priority: Base |
| The system should allow the lift to decide where it should go. There should be a variable to store the destination floor. Depend on F5.1 | | |

|  |  |  |
| --- | --- | --- |
| ID: F9.4 | Requirement: Control the lift door state | Priority: Base |
| The system should allow the lift to open door and close door. Depend on F5.1 | | |

|  |  |  |
| --- | --- | --- |
| ID: F9.5 | Requirement: Update the number of people in lift | Priority: Base |
| The system should allow the number of people in the lift updates. Depend on F5.1 | | |

### F10 --- Simulation

|  |  |  |
| --- | --- | --- |
| ID: F10.1 | Requirement: Show lift | Priority: Base |
| System should allow the simulation shows lifts. Depend on F5.2 | | |

|  |  |  |
| --- | --- | --- |
| ID: F10.2 | Requirement: Show stand by people | Priority: Base |
| System should allow the simulation shows at least 5 floors stand by people. Depend on F5.2 | | |

|  |  |  |
| --- | --- | --- |
| ID: F10.3 | Requirement: Show current people in lift | Priority: Base |
| System should allow the simulation shows current people in lift. Depend on F5.2 | | |

|  |  |  |
| --- | --- | --- |
| ID: F10.4 | Requirement: Show current lift door state | Priority: High |
| System should allow the simulation shows current lift door state. Depend on F5.2 | | |

|  |  |  |
| --- | --- | --- |
| ID: F10.5 | Requirement: Show all stand by people | Priority: High |
| System should allow the simulation shows all the stand by people. Depend on F5.2 | | |

|  |  |  |
| --- | --- | --- |
| ID: F10.6 | Requirement: Show specific lift state | Priority: High |
| System should allow the simulation shows specific lift state, include current floor, current people in lift, current lift door state. Depend on F5.2 | | |

|  |  |  |
| --- | --- | --- |
| ID: F10.7 | Requirement: Update simulation | Priority: High |
| System should allow the simulation updates. Depend on F5.2 | | |

### F11 --- History

|  |  |  |
| --- | --- | --- |
| ID: F11.1 | Requirement: Create history | Priority: High |
| System should generate history. Depend on F5.1, F9.2 | | |

|  |  |  |
| --- | --- | --- |
| ID: F11.2 | Requirement: Read history | Priority: High |
| System should allow permission user read history. Depend on F5.2 | | |

## Non-Functional Requirement

|  |  |  |
| --- | --- | --- |
| ID:N1 | Requirement: Easy to use | Priority: Medium |
| The system will be learned within a short time period by a user who often uses PC. | | |

|  |  |  |
| --- | --- | --- |
| ID:N2 | Requirement: Easy to understand | Priority: Medium |
| The system will be understood easily. | | |

|  |  |  |
| --- | --- | --- |
| ID:N3 | Requirement: Every window connects well | Priority: Medium |
| The system will be connected well, all the windows will have back button to previous window or the main window. | | |

# Scenarios

1.The lift is not running right now, and Alice pressed the up button, the lift will stop at the floor, and open the door, Alice get in the lift, the door closed, then, Alice choose the 5th button，the lift going up or down to send him to the 5th floor, the door is open, Alice get off the lift, after few second, and the door close automatically.

Relational functions:

showLiftDirection()

changeUpButton()

changeCurrentFloor()

changeDirection()

changeDoor()

changeFloorButton()

countTime()

checkOverweight()

changeDestination()

2.Alice presses the up button, and then the lift is coming, the door is open, but Alice does not get in the lift, the lift will wait for few second and then close the door, and stop moving.

Relational functions:

changeUpButton(),

showLiftDirection(),

changeCurrentFloor(),

changeDirection(),

changeDoor(),

countTime(),

changeDestination()

3. Alice and Bob in different floors press the up button at different time, the lift will pick the lower floor first. And if Alice and Bob press the down button, the lift will go to the higher level first.

Relational functions:

showLiftDirection(),

changeUpButton(),

changeCurrentFloor(),

changeDirection(),

changeDownButton(),

changeDoor(),

countTime(),

changeDestination()

4. Alice and Bob want go to different trend, the lift will deal with the person who presses the button first.

Relational functions:

showLiftDirection(),

changeCurrentFloor(),

changeUpButton(),

changeDownButton(),

changeDestination()

5. Alice stay at 4th floor and want to going up, but he presses the down button, the lift will stop at the floor, and open the door, Alice maybe get in the lift, then presses the higher floor button, the lift will close the door, and then open the door again, because nobody wants going down, it was a wrong order. And then the door is closed, the lift going up to send Alice.

Relational functions:

changeDownButton(),

changeCurrentFloor(),

showLiftDirection(),

changeDoor(),

changeFloorButton(),

countTime(),

checkOverweight()

6. Alice presses the up button, then the lift is coming, open door, Alice get in the lift, quickly press some floor button, and then get off the lift. So the lift will waiting for few second, and closes the door, at this moment, nobody in the lift, but the lift also needs to finish the mission.

Relational functions:

showLiftDirection(),

changeCurrentFloor(),

changeUpButton(),

changeDoor(),

changeFloorButton(),

checkOverweight(),

countTime(),

changeDestination()

7. Alice currently at 3th floor and wants going up, and lift is coming, door open, then Alice get in the lift, pressed 5th floor button, but it was not his destination, so he pressed 7th floor button. If the former floor is lower (2th floor), he has to stop at 2th floor first, and then Alice can reach the correct floor.

Relational functions:

showLiftDirection(),

changeCurrentFloor(),

changeUpButton(),

changeDoor(),

changeFloorButton(),

checkOverweight(),

countTime(),

changeDestination()

8.Alice currently at 3th floor and wants going up, and lift is coming, door open, then Alice get in the lift, pressed 5th floor button, but it was not his destination, so he pressed 7th floor button. But the floor button can be canceled, so Alice does not need to wait at the mistake floor.

Relational functions:

showLiftDirection(),

changeCurrentFloor(),

changeUpButton(),

changeDoor(),

changeFloorButton(),

checkOverweight(),

countTime(),

changeDestination()

9.Alice and Bob at the different floor, when the Alice (at 4th floor ) press the up button, then the lift (at 7th floor) move down to the 4th floor. While it is moving, the Bob (at 5th floor) press the up button, after it, the lift will arrive at the 4th floor first, and Alice comes into lift and choose 9th floor button, so the lift move up to 9th floor, the same steps will continues except Alice choose 6th floor button, after system process, the lift will arrive at 6th floor, then, Alice and Bob get out of lift first, then reach 9th floor after it.

Relational functions:

showLiftDirection(),

changeCurrentFloor(),

changeUpButton(),

changeDoor(),

changeFloorButton(),

checkOverweight(),

countTime(),

changeDestination()

10.Alice and Bob at the different floor, when Alice (at 5th floor) press the up button, then the lift (at 8th floor, M>N1) move down to the N1. While it is moving, Bob (at 6th floor) press the up button, after it , the lift will arrive at the 5th floor first , and Alice comes into lift and choose 7th floor button , so the lift move up to 6th floor, the same steps will continues except the person choose 9th floor button, after system process, the lift will arrive at 7th floor and Alice get out of lift first, then reach 9th floor after it.

Relational functions:

showLiftDirection(),

changeCurrentFloor(),

changeUpButton(),

changeDoor(),

changeFloorButton(),

checkOverweight(),

countTime(),

changeDestination()

11. Alice and Bob at the different floor, when Alice (at 4th floor) press the up button, then the lift (at 7th floor) move down to the . While it is moving, Bob(at 5th floor) press the up button, after it , the lift will arrive at the 4th floor first , and Alice comes into lift and choose 6th floor button, so the lift move up to 6th and Alice get out, next, the lift will get to the 5th floor; and Bob comes in and choose 7th floor button, after it, the lift will arrive at 7th floor and Alice get out of lift.

Relational functions:

showLiftDirection(),

changeCurrentFloor(),

changeUpButton(),

changeDoor(),

changeFloorButton(),

checkOverweight(),

countTime(),

changeDestination()

12. More than three people want to go upstairs, Alice which is the one of them press the up button, and then the rest of people do not need to press it any more. Seconds later, the lift is coming, the door is open, and then Bob and Carol get in the lift. Bob presses the floor button, and Carol presses the other floor button which is closer than the Bob’s, and then, and then, at the same time, the lift will close the door. The lift will go to the nearest floor then, Bob get off the lift, maybe someone press the close door button, the lift close the door, keep going up. If no any upper floors are required, the lift will stop at the last mission floor. It would not work until someone wants to use lift.

Relational functions:

showLiftDirection(),

changeCurrentFloor(),

changeUpButton(),

changeDoor()

,changeFloorButton(),

checkOverweight(),

countTime(),

changeDestination(),

changeCloseButton()

13.The lift is stop at one floor and there are already a lot of people in the lift, and then someone get in the lift, suddenly, the lift alarms, at this moment, the lift is overweight, so someone will get off the lift, usually the last one needs get off the lift. Then the lift can keep working.

Relational functions:

showLiftDirection()

,changeUpButton(),

changeDoor(),

checkOverweight(),

14. The lift is full of people, but the weight still under safety standard. Now, if Bob wants get in the lift, it still ok for weight, but there is no room for the people, when the lift wants to close the door, it will detect something is between the door, it will not close the door, until the “something” is moved away the door. So Bob cannot get in the lift as well.

Relational functions:

showLiftDirection(),

checkOverweight(),

countTime(),

changeDoor()

15. The lift is arrived one floor and Alice at the corner wants get off the lift, but it is too crowded, so it will cost some time to reach the door. At this moment, the lift will detect no one wants get off the lift anymore, so the lift will close the door, fortunately, someone next to the door help the person keep the door open by pressing the open door.

Relational functions:

showLiftDirection(),

changeDoor(),

countTime(),

checkOverweight()

16.The lift is going up to send a lot of people, someone get off at one floor, and at the same time, someone in the floor wants going down, even though the door is open, it is unnecessary to get in the lift. Because the lift will finished going up, then lift will turn back to bring him to down stair.

Relational functions:

showLiftDirection(),

changeDoor(),

changeDestination()

17. There is an emergency happened, like fire, the lift will shut down if nobody is in the lift. If the lift is sending somebody, it will stop at the nearest floor, and the people in the lift must get off the lift, then the lift will shut down.

Relational functions:

showLiftDirection(),

changeDoor(),

countTime(),

changeDestination(),

shutDown(),

checkOverweight()

18.The lift is out of energy, so the lift is stop at somewhere, there are some people still in the lift, the people need to press the call button, and someone in the security room will answer the call, and call the fireman to open the door, and the lift door is not allow to open inside, because it is dangerous to open it inside.

Relational functions:

checkOverweight(),

changeTalkingButton(),

19. The lift is used in the hospital, the speed of lift will well control not too fast, just in case patient would feel uncomfortable. And the lift would keep stable.

# Some pseudo code (Scenario 1 ~6)

**SC1**

c\_weight=0;

l\_current\_speed=0,

l\_current\_floor=3;

p\_current\_floor=G;

p\_position=outside;

p\_button=up;

c\_liftdestination=G;

c\_next\_floor=G;

c\_liftup=false;

c\_liftdown=true;

l\_currentspeed>0;

l\_current\_floor=G;

c\_door=open;

l\_door=open;

p\_position=inside;

l\_door\_button=close;

c\_weight>0;

if( c\_weight>l\_alarmweight){

c\_alarm=true;

}

if(c\_weight<l\_alarmweight){

c\_alarm=false;

}

if(c\_alarm==false){

c\_door=close;

l\_door=close;

l\_floorbutton=3;

c\_liftdestination=3;

c\_next\_floor=3;

c\_liftup=true;

c\_liftdown=false;

l\_current\_speed>0;

l\_current\_floor=3;

p\_current\_floor=3;

l\_current\_speed=0;

c\_door=open;

l\_door=close;

p\_position=outside;

c\_waiting\_time=5sec;

c\_door=close;

l\_door=close;

}

**SC2**

c\_weight=0;

l\_current\_speed=0,

l\_curent\_floor=3;

p1\_current\_floor=G;

p1\_position=outside;

p1\_button=up;

p2\_current\_floor=2;

p2\_position=outside;

p2\_button=up;

c\_liftdestination={G,2};

c\_next\_floor=G;

c\_liftup=false;

c\_liftdown=true;

l\_currentspeed>0;

l\_current\_floor=G;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p1\_position=inside;

c\_weight>0;

if( c\_weight>l\_alarmweight){

c\_alarm=true;

}

if(c\_weight<l\_alarmweight){

c\_alarm=false;

}

if(c\_alarm==false){

l\_floor\_button=3;

c\_liftdestination={2,3};

c\_next\_floor=2;

l\_door\_button=close;

c\_door=close;

l\_door=close;

c\_liftup=true;

c\_liftdown=false;

l\_currentspeed>0;

l\_current\_floor=2;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p2\_position=inside;

c\_weight>0;

if( c\_weight>l\_alarmweight){

c\_alarm=true;

}

if(c\_weight<l\_alarmweight){

c\_alarm=false;

}

if(c\_alarm==false){

l\_floor\_button=3;

c\_liftdestination={3};

c\_next\_floor=3;

l\_door\_button=close;

c\_door=close;

l\_door=close;

c\_liftup=true;

c\_liftdown=false;

l\_currentspeed>0;

l\_current\_floor=2;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p1\_position=outside;

p2\_position=outside;

c\_waiting\_time=5sec;

c\_door=close;

l\_door=close;

}

}

**SC3**

c\_weight=0;

l\_current\_speed=0,

l\_curent\_floor=3;

p1\_current\_floor=G;

p1\_position=outside;

p1\_button=up;

p2\_current\_floor=2;

p2\_position=outside;

p2\_button=down;

if (l\_position- l\_normalsafety>0) {

c\_liftdestination={2,G};

c\_next\_floor=2;

c\_liftup=false;

c\_liftdown=true;

l\_currentspeed>0;

l\_current\_floor=2;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p2\_position=inside;

c\_weight>0;

if( c\_weight>l\_alarmweight){

c\_alarm=true;

}

if(c\_weight<l\_alarmweight){

c\_alarm=false;

}

if(c\_alarm==false){

l\_floor\_button=G;

c\_liftdestination={G};

c\_next\_floor=G;

l\_door\_button=close;

c\_door=close;

l\_door=close;

c\_liftup=false;

c\_liftdown=true;

l\_current\_speed>0;

l\_current\_floor=G;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p2\_position=outside;

p1\_position=inside;

c\_weight>0;

if( c\_weight>l\_alarmweight){

c\_alarm=true;

}

if(c\_weight<l\_alarmweight){

c\_alarm=false;

}

if(c\_alarm==false){

c\_door=close;

l\_door=close;

l\_floorbutton=3;

c\_liftdestination=3;

c\_next\_floor=3;

c\_liftup=true;

c\_liftdown=false;

l\_current\_speed>0;

l\_current\_floor=3;

l\_current\_speed=0;

p\_current\_floor=3;

l\_current\_speed=0;

c\_door=open;

l\_door=close;

p1\_position=outside;

c\_waiting\_time=5sec;

c\_door=close;

l\_door=close;

}

}

}else{

c\_liftdestination={G,2};

c\_next\_floor=G;

c\_liftup=false;

c\_liftdown=true;

l\_currentspeed>0;

l\_current\_floor=G;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p1\_position=inside;

c\_weight>0;

if( c\_weight>l\_alarmweight){

c\_alarm=true;

}

if(c\_weight<l\_alarmweight){

c\_alarm=false;

}

if(c\_alarm==false){

l\_floor\_button=3;

c\_liftdestination={2,3};

c\_next\_floor=3;

l\_door\_button=close;

c\_door=close;

l\_door=close;

c\_liftup=true;

c\_liftdown=false;

l\_current\_speed>0;

l\_current\_floor=3;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p1\_position=outside;

c\_waiting\_time=5sec;

c\_door=close;

l\_door=close;

c\_liftdestination=2;

c\_next\_floor=2;

c\_liftup=false;

c\_liftdown=true;

l\_current­­\_speed>0;

l\_current\_floor=2;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p2\_position=inside;

c\_weight>0;

if( c\_weight>l\_alarmweight){

c\_alarm=true;

}

if(c\_weight<l\_alarmweight){

c\_alarm=false;

}

if(c\_alarm==false){

l\_floor\_button=G;

c\_liftdestination={G};

c\_next\_floor=G;

l\_door\_button=close;

c\_door=close;

l\_door=close;

c\_liftup=false;

c\_liftdown=true;

l\_currentspeed>0;

l\_current\_floor=G;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p2\_position=outside;

c\_waiting\_time=5sec;

c\_door=close;

l\_door=close;

}

}

}

**SC4**

c\_weight=0;

l\_current\_speed=0,

l\_curent\_floor=G;

p1\_current\_floor=10;

p1\_position=outside;

p1\_button=down;

p2\_current\_floor=5;

p2\_position=outside;

p2\_button=up;

c\_liftdestination={10,5};

c\_next\_floor=10;

c\_liftup=true;

c\_liftdown=false;

l\_current\_speed>0;

l\_current\_floor=10;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p1\_position=inside;

c\_weight>0;

if( c\_weight>l\_alarmweight){

c\_alarm=true;

}

if(c\_weight<l\_alarmweight){

c\_alarm=false;

}

if(c\_alarm==false){

l\_floor\_button=G;

c\_liftdestination={G,5};

c\_next\_floor=G;

l\_door\_button=close;

c\_door=close;

l\_door=close;

c\_liftup=false;

c\_liftdown=true;

l\_current\_speed>0;

l\_current\_floor=10;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p1\_position=outside;

c\_waiting\_time=5sec;

c\_door=close;

l\_door=close;

c\_liftdestination=5;

c\_next\_floor=5;

c\_liftup=true;

c\_liftdown=false;

l\_current\_speed>0;

l\_current\_floor=5;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p2\_position=inside;

c\_weight>0;

if( c\_weight>l\_alarmweight){

c\_alarm=true;

}

if(c\_weight<l\_alarmweight){

c\_alarm=false;

}

if(c\_alarm==false){

l\_floor\_button=10;

c\_liftdestination={10};

c\_next\_floor=10;

l\_door\_button=close;

c\_door=close;

l\_door=close;

c\_liftup=true;

c\_liftdown=false;

l\_currentspeed>0;

l\_current\_floor=10;

l\_current\_speed=0;

c\_door=open;

l\_door=open;

p2\_position=outside;

c\_waiting\_time=5sec;

c\_door=close;

l\_door=close;

}

}

**SC5**

if(c-fire==true&&c\_weight=0)

{

lift=shutdown;

}

if(c\_fire=true&&c\_weight>0)

{

c\_liftup or c\_liftdown=true;

c\_next\_floor=nearestfloor;

l\_current\_speed>0;

c\_current\_floor=nearestfloor;

c\_door=open;

l\_door=open;

p\_position=outside;

c\_waiting\_time=5sec;

c\_door=close;

l\_door=close;

l\_current\_speed=0;

lift=shutdown;

};

**SC6**

c\_lift\_up=true;

c\_lift\_down=false;

c\_weight>0;

c\_speed>0;

c\_lift\_destination=3;

c\_next\_floor=3;

if(c\_speed==0&&l\_current\_position!=c\_next\_floor){

c\_talking;

}

l\_door=open;

p\_position=outside;

l\_door=close;

# Class Structure and function command

* **Simulation part**

Readfile(); // read Lift\_status.txt

Initial(); // initial the data of lift and the user’s current floor (p\_floor)

Show(); // show the outside view of lift at floor “p\_floor”

Begin(); // wait user press button

If outside\_button { // the button inside the lift which can choose the destination floor.

DecideDirection(p\_floor);

UpdateFile();

CheckDestination(p\_floor); // Once the destination changed, lift start to work.

Move(p\_floor); //pass p\_floor as an parameter

OpenDoor();

CoutdownDoorclosetime;

CheckWeight();

CloseDoor();

}

Else if inside\_button{ // the up and down button outside the lift

DecideDirection(destination); // make decision of lift’s direction

UpdateFile();

// check whether there is a destination before this destination

CheckDestination(destination);

Move(destination);

OpenDoor();

CountdownDoorclosetime;

CheckWeight();

CloseDoor();

}

DecideDirection(destination){ //decide the direction of lift

If (current\_floor > destination) //current floor was set when read liftstatus.txt

SetDirection=Down;

Else if (current\_floor < destination)

SetDirection=Up;

Else

Opendoor();

UpdateFile(); // set the lift direction and update to the file

}

CheckDestination(destination); // Once the destination changed, lift start to work.

Move(p\_floor){ //pass p\_floor as an parameter

If (Direction == D){

Dvalue= current\_floor – p\_floor;

For(i = 0; i< Dvalue; i++){

CheckDestination(p\_floor);

Current\_floor --;

UpdateFile();

}

}

else if (Direction == U){

Dvalue = p\_floor – current\_floor;

for (i = 0; i< Dvalue; i++){

CheckDestination(p\_floor);

Current\_floor ++;

UpdateFile();

}

}

Else

Break;

}

# Function names and descriptions

|  |  |
| --- | --- |
| Function | changeDoor |
| Primary Actor | Person |
| Secondary Actors |  |
| Preconditions | User wants to use simulation to control lift door open and down in the simulation |
| Success End Condition | The door status is changed |
| Failed End Condition |  |
| Basic Flow | 1. user selects changeDoor 2. user selects the lift whether open/close door. 3. The door will open/close in the simulation |

|  |  |
| --- | --- |
| Function | ChangeFloorButton |
| Primary Actor | Person |
| Secondary Actors |  |
| Preconditions | User want to use the lift go a specified floor in the simulation |
| Success End Condition | Floor button is lit |
| Failed End Condition |  |
| Basic Flow | 1. user selects ChangeFloorButton 2. user selects a floor number 3. the specified floor button is lit in the simulation |

|  |  |
| --- | --- |
| Function | changeCloseButton |
| Primary Actor | Person |
| Secondary Actors |  |
| Preconditions | User wants to close lift door in the lift in the simulation |
| Success End Condition | The door is close in the simulation |
| Failed End Condition |  |
| Basic Flow | 1. user selects changeCloseButton 2. lift door will close in the simulation |

|  |  |
| --- | --- |
| Function | changeCurrentFloor |
| Primary Actor | Person |
| Secondary Actors |  |
| Preconditions | Display the number of current floor |
| Success End Condition | The lift arrived at the correct floor |
| Failed End Condition |  |
| Basic Flow | 1. user selects changeCurrentFloor 2. user selects a floor he/she wants to go 3. the value of liftCurrentFloor will change from initial floor to target floor in every floor |

|  |  |
| --- | --- |
| Function | changeOpenButton |
| Primary Actor | Person |
| Secondary Actors |  |
| Preconditions | User wants to open lift door in the lift in the simulation |
| Success End Condition | The door is open in the simulation |
| Failed End Condition |  |
| Basic Flow | 1. user selects changeOpenButton 2. lift door will open in the simulation |

|  |  |
| --- | --- |
| Function | changeDownButton |
| Primary Actor | Person |
| Secondary Actors |  |
| Preconditions | User wants to move down through using lift in the simulation |
| Success End Condition | The down button is lit outside the lift in the simulation |
| Failed End Condition |  |
| Basic Flow | 1. user selects changeDownButton 2. The down button is lit outside the lift in the simulation |

|  |  |
| --- | --- |
| Function | changeUpButton |
| Primary Actor | Person |
| Secondary Actors |  |
| Preconditions | User wants to move up through using lift in the simulation |
| Success End Condition | The up button is lit outside the lift in the simulation |
| Failed End Condition |  |
| Basic Flow | 1. user selects changeDownButton 2. The up button is lit outside the lift in the simulation |

|  |  |
| --- | --- |
| Function | changeTalkingButton |
| Primary Actor | Person |
| Secondary Actors |  |
| Preconditions | User wants to connect to the manager in the simulation |
| Success End Condition | The emergency call button is lit inside the lift in the simulation |
| Failed End Condition |  |
| Basic Flow | 1. user selects changeTalkingButton 2. The emergency call button is lit inside the lift in the simulation 3. User can connect to the manager in the simulation |

|  |  |
| --- | --- |
| Function | changeDestination |
| Primary Actor | Person |
| Secondary Actors |  |
| Preconditions | User wants to view the lift moving |
| Success End Condition | Lift will move to correct floor |
| Failed End Condition |  |
| Basic Flow | 1. user selects changeDestination 2. user selects floor number 3. the specified floor button is lit in the simulation 4. the selected floor values will add to the queue of ChangeDestination 5. lift will move to these floors according to the order in the ChangeDestination queue. |