PROYEK AKHIR

MATA KULIAH PERMODELAN BERORIENTASI OBJEK

SEMESTER GANJIL 2013-2014

ELEVATOR SIMULATION NEW PTIIK BUILDING PROGRAM

Disusun oleh :

Gagah Istaid Billah (125150200111100)

Ikhlasul A F (125150207111105)

M Arief Aminulloh (125150205111001)

DosenPengajar:WayanFirdausMahmudy, Ph.D.

PROGRAM STUDI INFORMATIKA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di era globalisasi ini, segela aktifitas kita tidak terlepas dari yang namanya teknologi komunikasi dan informasi. Baik disaat kuliah ataupun bekerja. Dengan kemajuan ilmu pengetahuan yang sangat pesat, persaingan antara customer atau pengguna gadget dan produsen selaku pembuatnya semakin sengit. Dan pembuatan gadget dan software baru-pun semakin banyak bermunculan akibat tuntutan zaman yang semakin maju.

Dalam pembuatan software, para programmer harus bisa merancang dengan baik program yang dibuatnya, untuk itu perlu bekal ilmu dengan mempelajari Pemodelan Berorientasi Object. Di dalam disiplin ilmu ini, para pembelajar ditekankan pada proses perancangan program. Semakin bagus rancangan yang dibuat, maka program yang dibuat akan semakin bagus.

Di dalam makalah ini, kelompok kami membuat program simulasi lift/elevator yang menampilkan secara visual bagaimana sebuah lift itu bekerja dalam kehidupan nyata serta rancangan-rancangan yang digunakan dalam pembuatan program ini. Rancangan yang kami buat mengacu pada beberapa jenis diagram UML.

1.2 Rumusan Masalah

1) Bagaimana proses atau state berjalannya program simulator ini?
2) Bagaimana rancangannya menggunakan diagram UML?
3) Apa saja media yang digunakan dalam pembuatan program ini?
4) Hal apa sajakah yang perlu diperhatikan dalam pembuatan program ini agar dapat berjalan dengan baik?

1.3 Batasan Masalah

Jadi karena banyak orang yang tidak mengerti bagaimana cara kerja sebuah elevator, maka kami disini akan menjelaskan bagaimana cara kerja serta implementasi elevator dari segi teknis dan mekanis yang sudah ada saat ini.

1.4 Tujuan

1) Untuk mempelajari bagaimana simulator lift bekerja.
2) Untuk lebih memahami perancangan diagram UML ke program.

1.5 Manfaat

Dengan adanya program ini, kita dapat memberikan pemahaman mengenai bagaimana mekanisme elevator kepada pengguna yang ingin mempelajari.

1.6 Sistematika Penulisan

Sistematika penulisan makalah ini menerangkan tentang cara pembuatan program elevator yang terdiri dari beberapa bab yaitu :

BAB I PENDAHULUAN
Berisi latar belakang, Rumusan masalah, Batasan masalah, Tujuan, Manfaat.

BAB II DASAR TEORI
Menjelaskan tentang Permodelan bororientasi objek dan Unified Modeling Language.

BAB III PEMBAHASAN
Berisi tentang penjelasan UML, Class Diagram, Diagram Use Case, State Diagram, Sequence Diagram.

BAB IV PENUTUP
Berisi kesimoulan dan saran untuk keperluan penerapan maupun pengembangan selanjutnya.

BAB II
DASAR TEORI

2.1 Pemodelan Berorientasi Objek

Pemodelan Berorientsi Objek menangkap struktur statis dari sistem dengan menggambarkan objek dalam sistem, hubungan antara objek, serta atribut dan operasi yang merupakan karakteristik setiap kelas dan objek. Model objek adalah hal yang paling penting dari ketiga model. Model berorientasi objek lebih mendekati keadaan nyata, dan dilengkapi dengan penyajian grafik dari sistem yang sangat bermanfaat untuk komunikasi dengan kustomer dan pembuatan dokumentasi struktur dari sistem.

a). Objek

Objek didefinisikan sebagai konsep, abstraksi atau benda dengan dengan batasan dan arti untuk suatu masalah. Objek menjelaskan dua tujuan, yaitu pengertian tentang dunia nyata dan dilengkapi dengan dasar praktek untuk implementasi komputer. Dekomposisi dari suatu masalah ke dalam objek tergantung pada keputusan dan kenyataan dari masalah tersebut. Tak ada sesuatu penyajian yang sempurna.

Objek didefinisikan sebagai konsep, abstraksi atau benda dengan dengan batasan dan arti untuk suatu masalah. Objek menjelaskan dua tujuan, yaitu pengertian tentang dunia nyata dan dilengkapi dengan dasar praktek untuk implementasi komputer. Dekomposisi dari suatu masalah ke dalam objek

tergantung pada keputusan dan kenyataan dari masalah tersebut. Tak ada sesuatu penyajian yang sempurna.

b). Kelas

Suatu *object class* menggambarkan kumpulan dari objek yang mempunyai sifat (atribut), peri laku umum (operasi), relasi umum dengan objek lain dan semantik umum. Orang, perusahaan, binatang, proses dan window adalah objek. Setiap orang mempunyai umur, IQ dan mungkin pekerjaan. Setiap proses mempunyai pemilik, prioritas, list dari sumber daya yang dibutuhkan.

Objek dan object classes sering sama sebagai benda dalam deskripsi masalah. Istilah kelas sering digunakan dari pada Kelas-obyek. Objek dalam kelas mempunyai atribut dan pola peri laku yang sama. Sebagian besar objek diturunkan sifatnya dari perbedaan nilai atributnya dan relasi dengan objek lain. Bagaimanapun juga, terdapat kemungkinan adanya objek dengan nilai atribut dan relasi yang identik.

Setiap objek mengetahui kelasnya. Sebagian besar bahasa pemrograman berorientasi objek dapat menentukan kelas yang dimiliki oleh objek dalam *run time*-nya. Bila objek difokuskan pada pemodelan objek, bagaimana dengan kelas? Dengan mengumpulkan objek ke dalam kelas, kita membuat abstraksi suatu masalah. Abstraksi memberikan pemodelan yang mempunyai kehandalan untuk membuat generalisasi dari beberapa hal yang spesifik menjadi suatu yang lebih umum.

2.2 Unified Modeling Language (UML)

Unified Modelling Language (UML) adalah sebuah “bahasa” yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun.Selain menggunakan UML, perancangan ssistem dapat juga menggunakan Diagram Objek.

Diagram objek melengkapi notasi grafik untuk pemodelan objek, kelas dan relasinya dengan yang lain. Diagram objek bermanfaat untuk pemodelan abstrak dan membuat perancangan program. Diagram objek sederhana, mudah dipahami dan bekerja dengan baik dalam praktek. Terdapat dua macam diagram objek, yaitu diagram kelas dan diagram instance.

Diagram kelas adalah skema, pola, atau template untuk menjelaskan banyak kemungkinan data dari instance. Diagram kelas menjelaskan klas-objek. Diagram instance menjelaskan bagaimana satu set objek tertentu berhubungan

dengan yang lainnya. Diagram instance menggambarkan *objectinstance.* Diagram instance bermanfaat untuk membuat dokumentasi dari suatu kasus.

Macam-Macam Diagram UML adalah :

a). Use Case Diagram

Use Case diagram adalah gambar dari beberapa atau seluruh aktor dan use case dengan tujuan mengenali interaksi mereka dalam suatu sistem. Oleh karena itu, use case diagram dapat membantu menganalisa kebutuhan suatu sistem. Dalam use case diagram terdapat istilah seperti aktor, use case dan use case relationship.

b). Kelas Diagram

Kelas Diagram berfungsi untuk menjelaskan tipe dari object sistem dan hubungannya dengan object yang lain. Object adalah nilai tertentu dari setiap attribute kelas entity. Pada penggambaran kelas diagram ada dikenal dengan kelas analisis yaitu kelas ber-stereotype. Tapi yang biasanya dipakai adalah kelas diagram tanpa stereotype.

c). State Diagram

State diagram menggambarkan urutan keadaan yang dilalui object dalam suatu kelas, karena suatu kejadian menyebabkan suatu perpindahan aktivitas/state. State dari objek adalah penggolongan dari satu atau lebih nilai attribute pada kelas.

d). Activity Diagram

Activity Diagram berupa flow chart yang digunakan untuk memperlihatkan aliran kerja dari sistem. Notasi yang digunakan dalam activity diagram adalah sebagai berikut: Aliran kerja notasi ini menandakan bahwa beberapa aktivitas dapat diselesaikan secara bersamaan (pararel).

e). Sequence Diagram

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya sequence diagram adalah gambaran tahap demi tahap yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan use case diagram.

f). Collaboration Diagram

Collaboration adalah cara alternatif untuk mengetahui tahap-tahap terjadinya suatu aktivitas. Perbedaan antara collaboration dan sequence diagram adalah collaboration diagram memperlihatkan bagaimana hubungan antara beberapa objek, sedangkan yang kedua sequence diagram memperlihatkan bagaimana urutan kejadian.

g). Component Diagram

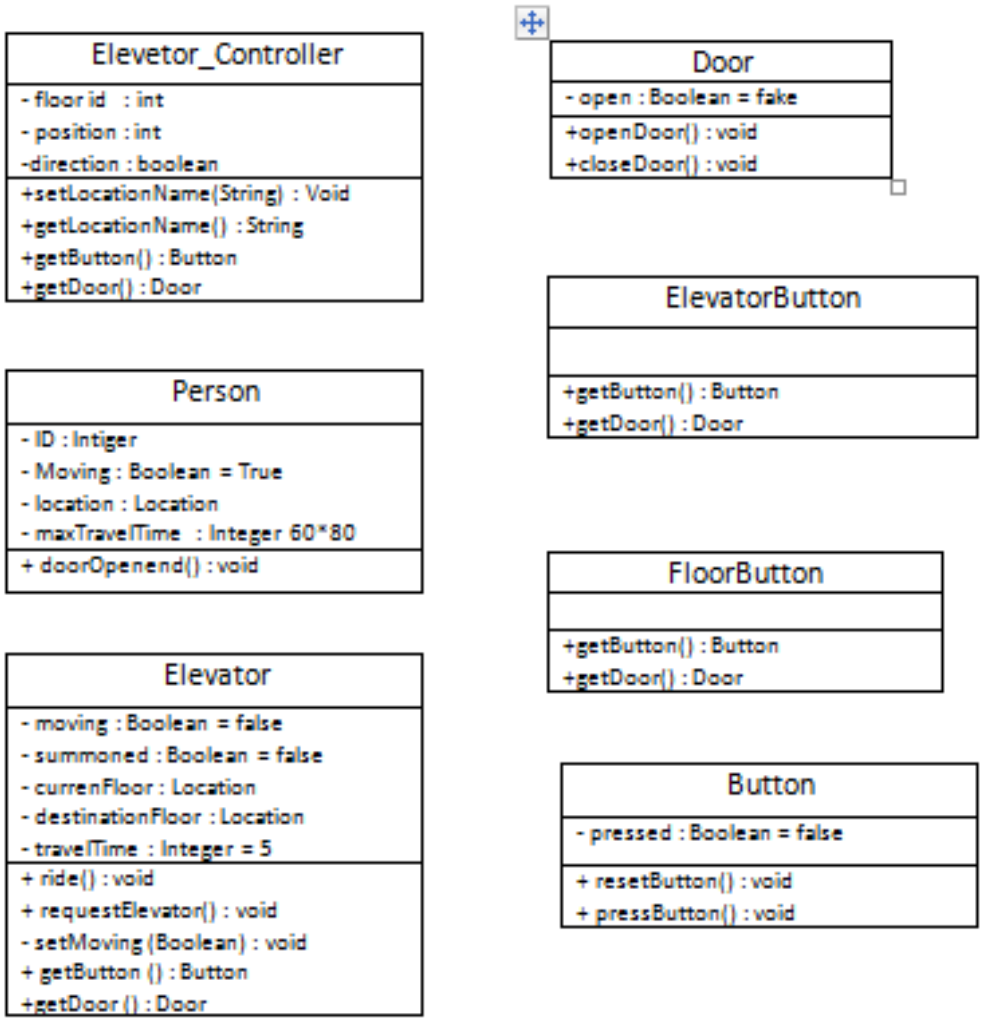
Component diagram berfungsi untuk menggambarkan komponen run-time dan executable yang dibuat untuk sistem. Komponen saling berelasi menggunakan depedecy relation (Hubungan ketergantungan, yang ditandai dengan garis putus-putus). Komponen run-time memperlihatkan pengelompokan kelas untuk run-time library seperti Java Applet, Active-X Component dan Dynamic Libraries. Komponen executable memperlihatkan interface dan memanggil dependencies beberapa executable. Interface kelas diperlihatkan seperti lollypop.

h). Deployment Diagram

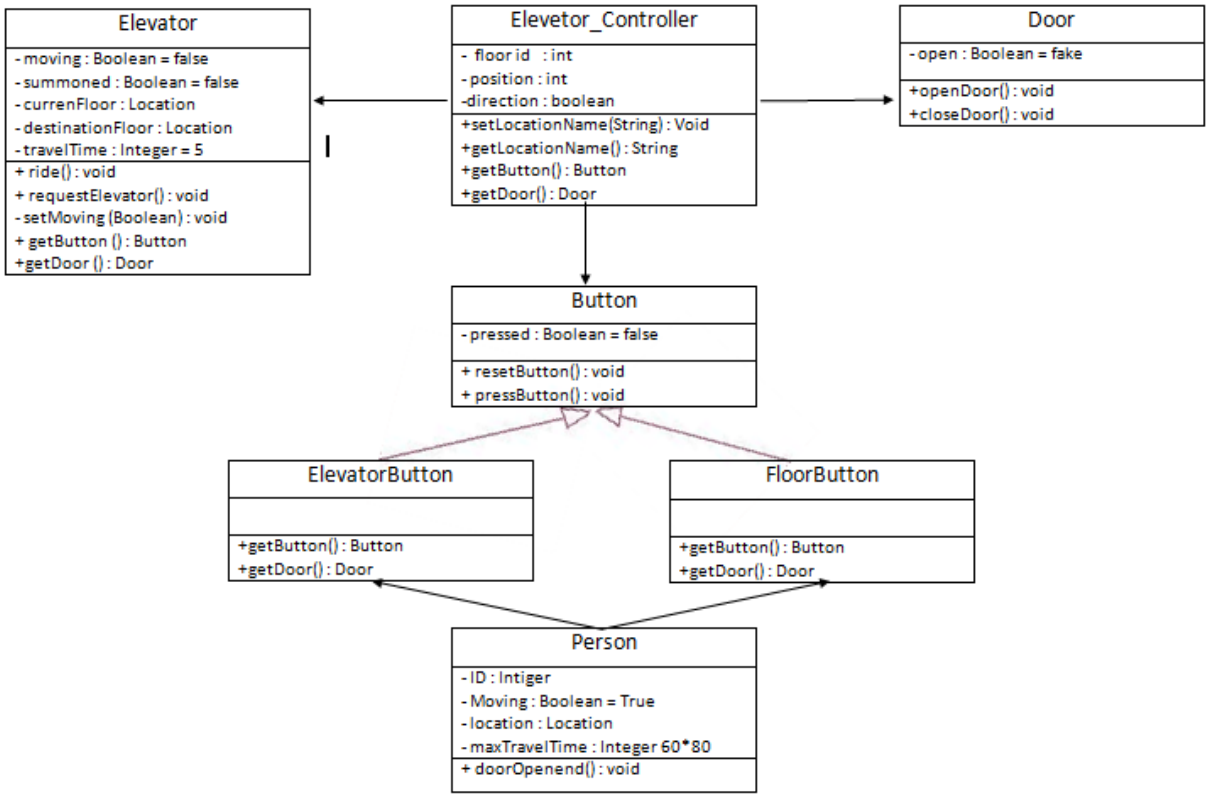
Deployment Diagram memperlihatkan konfigurasi pada jalannya proses run-time elements dan proses software yang ada pada diagram. Run-time elements menggambarkan node yang berkoneksi menandakan adanya komunikasi diantaranya. Diagram ini membantu tim untuk mengerti sistem topology

BAB III
PEMBAHASAN

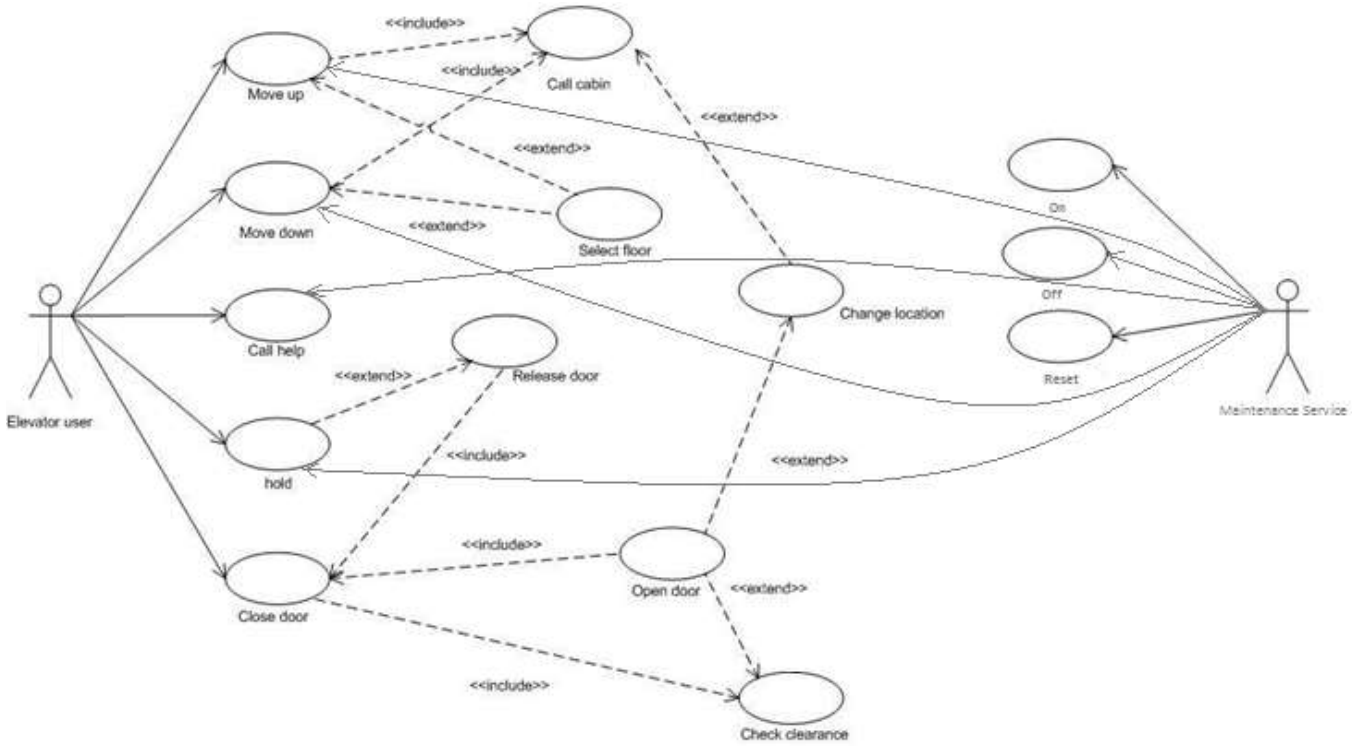
3.1 UML



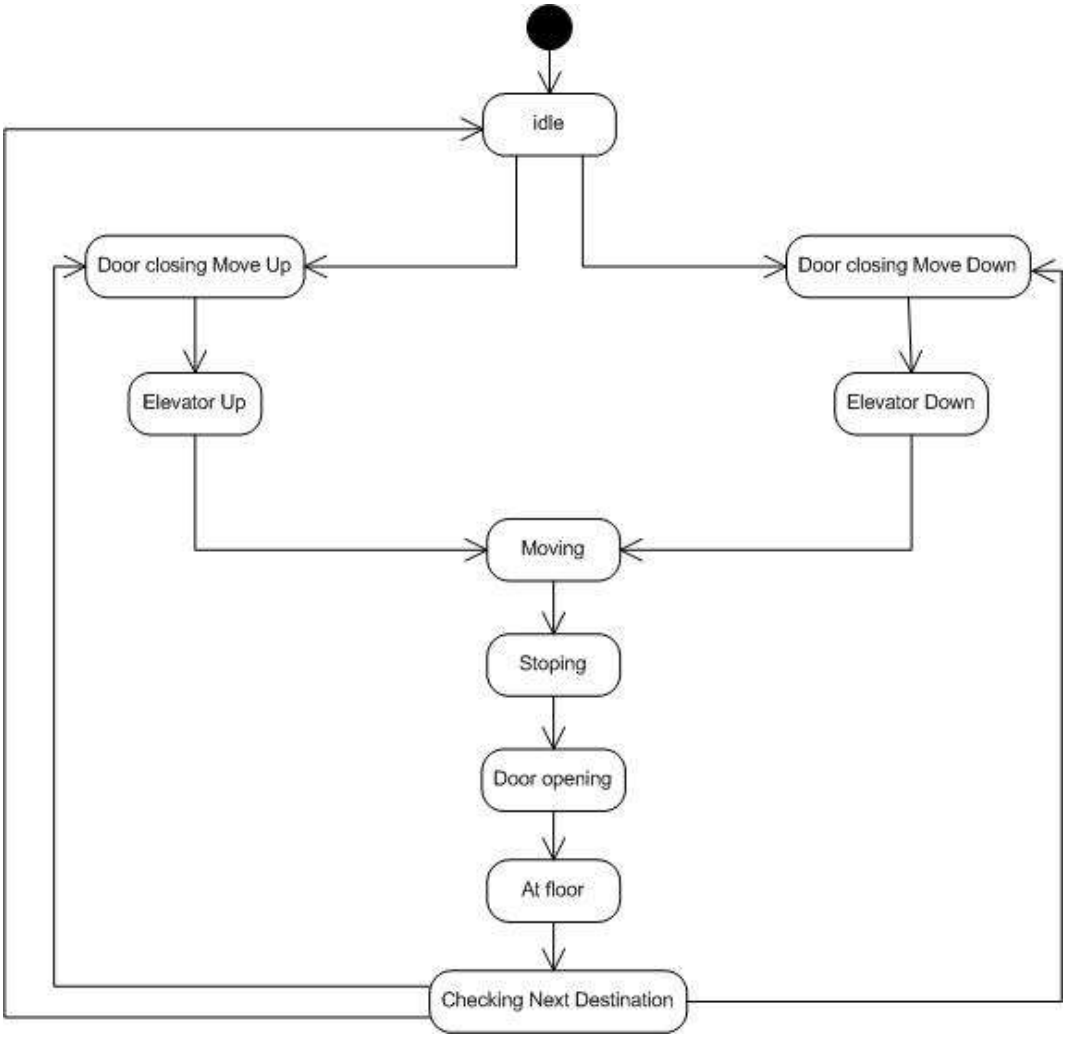
3.2 Class Diagram



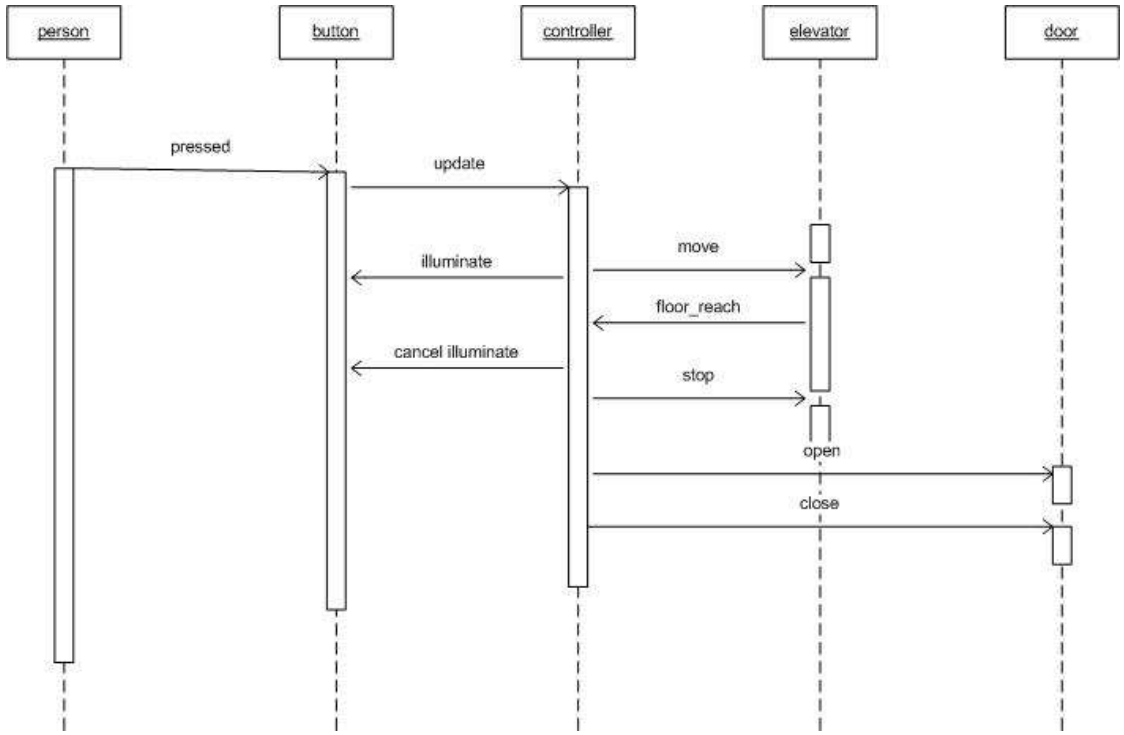
3.3 Diagram Use-Case



3.4 State Diagram



3.5 Sequence Diagram



BAB IV
IMPLEMENTASI

1. ElevatorSimulation.java

```
1   /**
2    *
3    * @author Agahxyz
4    */
5
6   import java.awt.Color;
7   import java.awt.event.ActionEvent;
8   import java.awt.event.ActionListener;
9   import javax.swing.*;
10
11  //Class ElevatorSimulation sebagai Controller
12  public class ElevatorSimulation extends JFrame implements
13  ActionListener {
14      JButton liftBtn[][] = new JButton[13][2];
15      JButton buttonFloor [] = new JButton[13];
```

```
16      private JButton reset, off, on, liftY;
17      private int x = 149, y = 536, yF = 536;
18      int wX = 45, wY = 26;
19      Timer timerLift = new Timer(20, this);
20      Timer timeDoorClose = new Timer(10, this);
21      Timer timeStopped = new Timer(2000, this);
22      boolean liftBtnStatus[][] = new boolean[13][2];
23      int currFloor = 1;
24      boolean liftAct = true;
25      boolean onTheWay = false;
26      int countStep = 0, stoppedCount = 0;
27      boolean stoppedAtDest = true;
28
29      //Constructor
30      public ElevatorSimulation() {
31          setContentPane(new JLabel(new
32  ImageIcon(getClass().getResource("BackgroundNew2.jpg"))));
33          liftBtn[0][0] = liftBtn[12][1] = null;
34
35          liftY = new JButton();
36          liftY.addActionListener(this);
37          liftY.setBounds(x, y, wX, wY);
38          liftY.setBackground(Color.MAGENTA);
39          add(liftY);
40
41          reset = new JButton("reset");
42          reset.addActionListener(this);
43          reset.setBounds(25, 535, 65, 20);
44          add(reset);
45
46          off = new JButton("off");
47          off.addActionListener(this);
48          off.setBounds(25, 505, 65, 20);
49          add(off);
50
51          on = new JButton("on");
52          on.addActionListener(this);
53          on.setBounds(25, 475, 65, 20);
54          add(on);
55
56          for (int i = 0; i < 2; i++) {
57              String in [] = new String[2];
58              in[i] = String.valueOf(i+1);
59              buttonFloor[i] = new JButton(in[i]);
60              buttonFloor[i].setBounds(295 + (60*i), 320, 50, 30);
61              buttonFloor[i].addActionListener(this);
62              add(buttonFloor[i]);
63          }
64
65          for (int i = 2; i < 4; i++) {
66              String in [] = new String[4];
```

```
67              in[i] = String.valueOf(i+1);
68              buttonFloor[i] = new JButton(in[i]);
69              buttonFloor[i].setBounds(295 + (60*(i-2)), 280, 50, 30);
70              buttonFloor[i].addActionListener(this);
71              add(buttonFloor[i]);
72          }
73
74          for (int i = 4; i < 6; i++) {
75              String in [] = new String[6];
76              in[i] = String.valueOf(i+1);
77              buttonFloor[i] = new JButton(in[i]);
78              buttonFloor[i].setBounds(295 + (60*(i-4)), 240, 50, 30);
79              buttonFloor[i].addActionListener(this);
80              add(buttonFloor[i]);
81          }
82
83          for (int i = 6; i < 8; i++) {
84              String in [] = new String[8];
85              in[i] = String.valueOf(i+1);
86              buttonFloor[i] = new JButton(in[i]);
87              buttonFloor[i].setBounds(295 + (60*(i-6)), 200, 50, 30);
88              buttonFloor[i].addActionListener(this);
89              add(buttonFloor[i]);
90          }
91
92          for (int i = 8; i < 10; i++) {
93              String in [] = new String[10];
94              in[i] = String.valueOf(i+1);
95              buttonFloor[i] = new JButton(in[i]);
96              buttonFloor[i].setBounds(295 + (60*(i-8)), 160, 50, 30);
97              buttonFloor[i].addActionListener(this);
98              add(buttonFloor[i]);
99          }
100
101         for (int i = 10; i < 12; i++) {
102             String in [] = new String[12];
103             in[i] = String.valueOf(i+1);
104             buttonFloor[i] = new JButton(in[i]);
105             buttonFloor[i].setBounds(295 + (60*(i-10)), 120, 50, 30);
106             buttonFloor[i].addActionListener(this);
107             add(buttonFloor[i]);
108         }
109
110
111             buttonFloor[12] = new JButton("13");
112             buttonFloor[12].setBounds(325, 80, 50, 30);
113             buttonFloor[12].addActionListener(this);
114             add(buttonFloor[12]);
115
116
117
```

```
118
119         for (int i = 0; i < 13; i++) {
120             for (int j = 0; j < 2; j++) {
121                 liftBtn[i][j] = new JButton("");
122                 if(i != 0 && j == 0){
123                     liftBtn[i][j].setBounds(120, 510 - (42 * (i-1)),
124 13, 13);
125                     liftBtnStatus[i][j] = false;
126                 }
127                 if(i != 12 && j == 1){
128                     liftBtn[i][j].setBounds(120, 535 - (42 * (i)), 13,
129 13);
130                     liftBtnStatus[i][j] = false;
131                 }
132                 liftBtn[i][j].addActionListener(this);
133                 add(liftBtn[i][j]);
134             }
135         }
136         timerLift.start();
137     }
138
139 /*Method ini berfungsi mengembalikan nilai tombol yg ditekan..*/
140     public int getNextFloor(){
141         // Elevator bergerak keatas jika liftAct bernilai true..
142         if (liftAct == true) {
143
144             // check lantai diatas currFloor --> check tombol up
145             for (int i = currFloor-1; i < 13; i++) {
145                 if(liftBtnStatus[i][1] == true){
146                     return i+1;
147                 }
148             }
149
150             // check lantai dibawah currFloor --> check tombol down
151             for (int i = 0; i < currFloor; i++) {
152                 if(liftBtnStatus[i][0] == true){
153 //                    System.out.println("-> " + i+1);
154                     return i+1;
155                 }
156             }
157
158             // check lantai dibawah currFloor --> check tombol up
159             for (int i = 0; i < currFloor; i++) {
160                 if(liftBtnStatus[i][1] == true){
161                     return i+1;
162                 }
163             }
164
165             // check tombol down
166             for (int i = currFloor-1; i < 13; i++) {
167                 if(liftBtnStatus[i][0] == true){
```

```
168                     return i+1;
169                 }
170             }
171 
172         } else {
173             //check lantai di bawah currFloor --> cek tombol down
174             for (int i = currFloor-1; i >= 0; i--) {
175                 if(liftBtnStatus[i][0] == true){
176                     return i+1;
177                 }
178             }
179 
180             // cek tombol up
181             for (int i = 0; i < 13; i++) {
182                 if(liftBtnStatus[i][1] == true){
183                     return i+1;
184                 }
185             }
186 
187             //check tombol down
188             for (int i = 0; i < 13; i++) {
189                 if(liftBtnStatus[i][0] == true){
190                     return i+1;
191                 }
192             }
193 
194             // check tombol up
195             for (int i = currFloor-1; i >= 0; i--) {
196                 if(liftBtnStatus[i][1] == true){
197                     return i+1;
198                 }
199             }
200         }
201         return 0;
202     }
203 
204     //Melakukan pergerakan naik..
205     public void moveUp(){
206         if(currFloor != 13){
207             y--;
208             liftY.setBounds(x, y, wX, wY);
209         }
210     }
211 
212     //Melakukan pergerakan turun
213     public void moveDown(){
214         if(currFloor != 1){
215             y++;
216             liftY.setBounds(x, y, wX, wY);
217         }
218     }
```

```
219
220     //Memberikan aksi disetiap tindakan..
221     @Override
222     public void actionPerformed(ActionEvent e) {
223         int nextFloor;
224         //Memberi aksi pada tombol reset
225         if (e.getSource() == reset) {
226             setEnabledButton(true, true);
227             liftY.setBounds(x, yF, wX, wY);
228 
229         //Memberi aksi pada tombol on
230         } else if (e.getSource() == on) {
231             setEnabledButton(true, true);
232             off.setBackground(reset.getBackground());
233 
234         //Memberi aksi pada tombol off
235         } else if (e.getSource() == off) {
236             setEnabledButton(false, true);
237             off.setBackground(Color.red);
238 
239         //Timer timeStopped
240         } else if (e.getSource() == timeStopped){
241 
242             timerLift.start();
243 
244         //Timer timerLift
245         } else if (e.getSource() == timerLift){
246             nextFloor = getNextFloor();
247             if(nextFloor > currFloor){
248                 liftAct = true;
249             } else if(nextFloor < currFloor){
250                 liftAct = false;
251             }
252             System.out.println("Sedang Berada di Lantai -> " +
253 currFloor);
254             if(nextFloor > 0){
255                 if(nextFloor != currFloor){
256                     stoppedAtDest = false;
257                     if(onTheWay == false){
258                         countStep = 0;
259                         stoppedCount = 0;
260                         onTheWay = true;
261                     }
262                     //Pergerakan Elevator..
263                     if(onTheWay == true){
264                         countStep++;
265                         if(liftAct){
266                             moveUp();
267                         } else {
268                             moveDown();
269                         }
```

```
270                         //Cek currFloor
271                         if(countStep == 42){
272                             if(liftAct){
273                                 currFloor++;
274                             } else {
275                                 currFloor--;
276                             }
277                             onTheWay = false;
278                         }
279                     }
280 
281                 } else {
282                     liftBtnStatus[currFloor-1][0] = false;
283                     liftBtnStatus[currFloor-1][1] = false;
284                     timerLift.stop();
285                     timeStopped.start();
286                 }
287             }
288         } else {
289             //Untuk memberikn aksi button up dan down di setiap
290 lantai..
291             for (int i = 0; i < 13; i++) {
292                 for (int j = 0; j < 2; j++) {
293                     if(liftBtn[i][j] == e.getSource()){
294                         liftBtnStatus[i][j] = true;
295                     }
296                 }
297             }
298             //Untuk memberikan aksi buttonFloor di setiap lantai..
299             for (int i = 0; i < 13; i++) {
300                 if (e.getSource() ==  buttonFloor[i]){
301                     if(liftAct){
302                         liftBtnStatus[i][1] = true;
303                     }else
304                         liftBtnStatus[i][0] = true;
305                 }
306             }
307 
308 
309         }
310     }
311 
312     //Method pengeset Press dan UnPress tombol..
313     public void setEnabledButton(boolean seb1, boolean seb2){
314         for (int i = 0; i < 13; i++) {
315             for (int j = 0; j < 2; j++) {
316                 if(i != 0 && j == 0){
317                     liftBtn[i][j].setEnabled(seb1);
318                 }
319                 if(i != 12 && j == 1){
320                     liftBtn[i][j].setEnabled(seb1);
```

```
                }

            }
        }
        reset.setEnabled(seb2);
        off.setEnabled(seb2);
        on.setEnabled(seb2);
    }
}

```

2. Lifting.java

```
/**
 *
 * @author Agahxyz
 */

import javax.swing.JFrame;

//Class Lifting sebagai Runner program
public class Lifting {
        //Main Method
        public static void main(String[] args) {
            ElevatorSimulation go = new ElevatorSimulation();
            go.setTitle("Elevator Simulation New PTIIK Building
Program");
            go.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            go.setBounds(455, 50, 455, 660);
            go.setVisible(true);
            go.setResizable(false);
            go.show();
    }
}
```

BAB V
PENUTUP

5.1 Kesimpulan

Elevator sangat bermanfaat bagi semua orang, ditambah adanya lahan bangunan yang terbatas sehingga lebih efisien kita membangun gedung yang lebih tinggi. Otomatis kita akan membutuhkan fasilitas yang mempermudah kita untuk mengakses semua lantai pada gedung tersebut. Elevatorlah jawabannya. Seiring berjalannya waktu dan berkembangnya teknologi, elevator telah berkembah lebih jauh. Tantangan sebenarnya adalah bukan hanya menyediakan fasilitas elevator yang mampu mempermudah aktifitas kita, akan tetapi juga meminimalkan waktu tunggu seefisien mungkin.

5.2 Saran

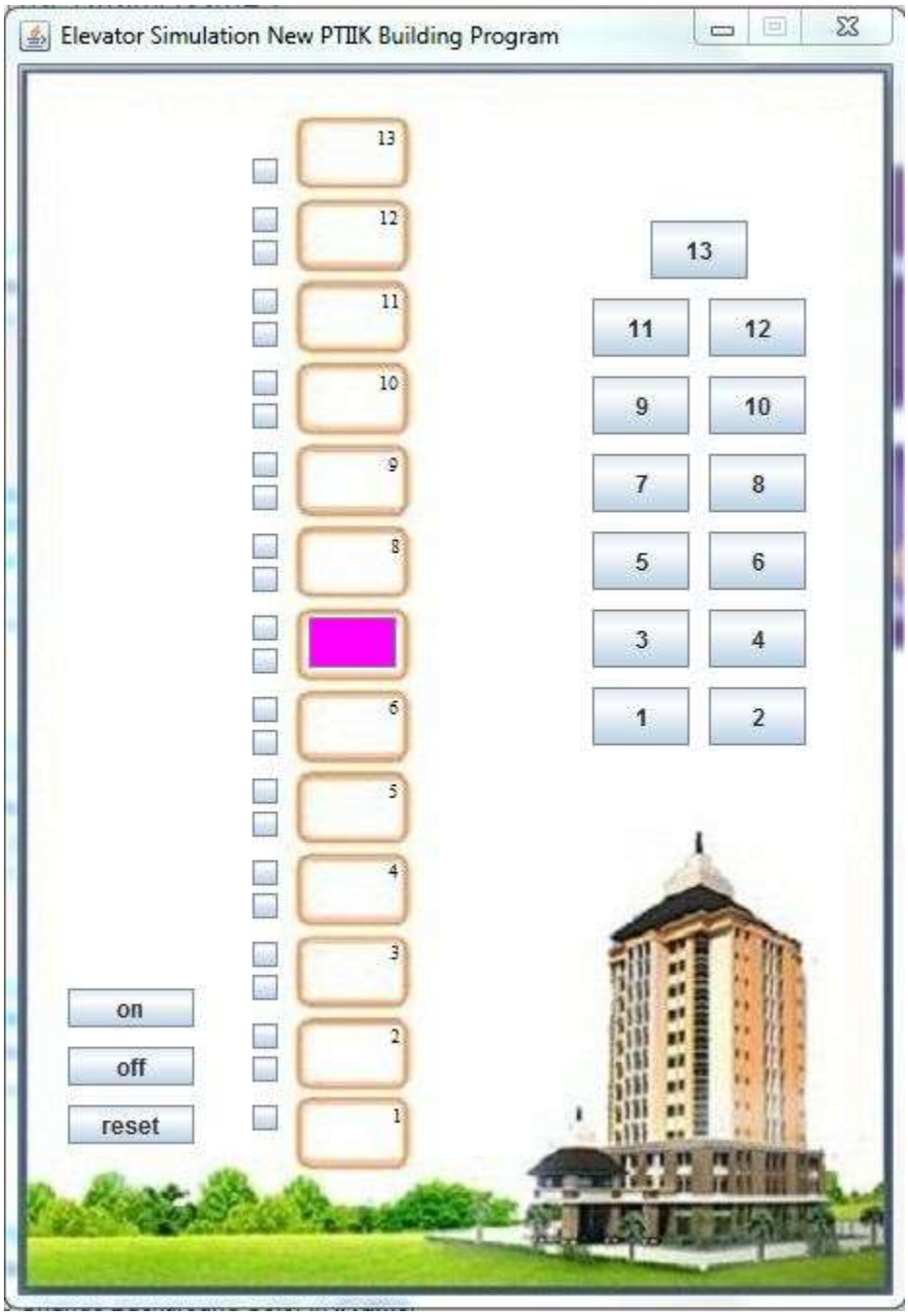
Berdasarkan hasil dari proyek akhir yang kami kerjakan, masih terdapat beberapa kekurangan yang bisa diperbaiki oleh pembaca untuk meningkatkan kinerja program ini sehingga dapat berjalan sesuai dengan ekspektasi kita dan juga berjalan lebih efisien dan meminimalkan waktu tunggu para calon pengguna elevator. Kami menerima saran dari pembaca dalam bentuk apapun guna menyempurnakan project ini.

DAFTAR PUSTAKA

Macheads101.2011. *Moving a Graphic*, http://www.youtube.com/watch?v=oynZhQjMv0c, (diakses 21 desember 2013)

MrJavaHelp.2010. *SWING/GUI in Java Tuturial*, http://www.youtube.com/watch?v=-sOqzUs1Hqk, (diakses 21 desember 2013)

LAMPIRAN


Elevator Simulation New PTIIK Building Program
13
12
11
10
9
8
6
5
4
3
2
1
13
11
12
9
10
7
8
5
6
3
4
1
2
on
off
reset