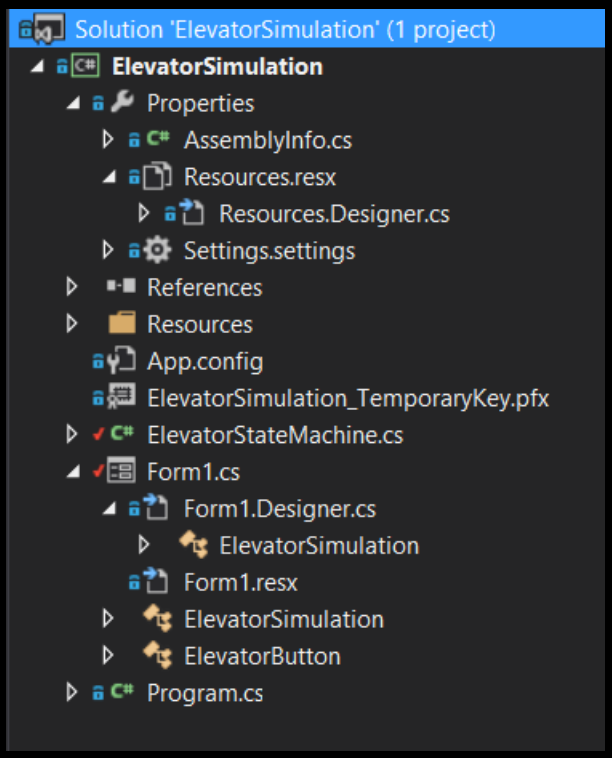
Aşağıda ElevatorStateMachine.cs ve Form1.cs dosyaları verilmiştir.




```
/*****************************************
 * ElevatorStateMachine.cs
 * State diagrama bağlı olarak
 * ElevatorStateMachine nesnesi için gerekli
 * class tanımlamalarını içerir.
 *****************************************/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ElevatorStateMachine
{

    public enum ElevatorState
    {
        Kat1, Kat2Up, Kat2Down, Kat3
    }
    public enum Command
    {
        a, b, c, d
    }
    public class Elevator
    {
        class StateTransition
        {

            readonly ElevatorState CurrentState;
            readonly Command Command;

            public StateTransition(ElevatorState currentState, Command command)
            {
                CurrentState = currentState;
                Command = command;
            }
            public override int GetHashCode()
            {
                return 17 + 31 * CurrentState.GetHashCode() + 31 * Command.GetHashCode();
            }
            public override bool Equals(object obj)
            {
                StateTransition other = obj as StateTransition;
                return other != null && this.CurrentState == other.CurrentState && this.Command ==
other.Command;
            }

        }

        Dictionary<StateTransition, ElevatorState> transitions;
        public ElevatorState CurrentState { get; private set; }

        public Elevator()
        {
            //Dictionary içerisinde transition table benzeri bir şekilde tanımlanmıştır.
            CurrentState = ElevatorState.Kat2Up;
            transitions = new Dictionary<StateTransition, ElevatorState>
            {
                { new StateTransition(ElevatorState.Kat1, Command.a), ElevatorState.Kat2Up},
                { new StateTransition(ElevatorState.Kat1, Command.c), ElevatorState.Kat2Down},
                { new StateTransition(ElevatorState.Kat3, Command.b), ElevatorState.Kat2Up},
                { new StateTransition(ElevatorState.Kat3, Command.d), ElevatorState.Kat2Down},
                { new StateTransition(ElevatorState.Kat2Up, Command.a), ElevatorState.Kat3},
                { new StateTransition(ElevatorState.Kat2Up, Command.b), ElevatorState.Kat1},
```

```
                { new StateTransition(ElevatorState.Kat2Down, Command.a), ElevatorState.Kat3},
                { new StateTransition(ElevatorState.Kat2Down, Command.b), ElevatorState.Kat1}

            };

        }
        public ElevatorState GetNext(Command command)
        {
            StateTransition transition = new StateTransition(CurrentState, command);
            ElevatorState nextState;
            if (!transitions.TryGetValue(transition, out nextState))
                throw new Exception("Invalid transition: " + CurrentState + " -> " + command);
            return nextState;
        }

        public ElevatorState MoveNext(Command command)
        {
            CurrentState = GetNext(command);
            return CurrentState;
        }
    }
 }
```

```
/*************************************************
 * Form1.cs
 * State machine kullanarak
 * 3. katlı bir asansör simüle eder.
 * ElevatorStateMachine içesirisinde tanımlanan
 * nesneler burada kullanılmaktadır.
 *************************************************/
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Media;

using ElevatorStateMachine;


namespace ElevatorSimulation
{
       public partial class ElevatorSimulation : Form
    {
        //Asansör ve Buton nesneleri oluştur.
        Elevator p = new Elevator();
        ElevatorButton k1 = new ElevatorButton();
        ElevatorButton k2 = new ElevatorButton();

        ElevatorButton k3 = new ElevatorButton();
        ElevatorButton b1 = new ElevatorButton();
        ElevatorButton b2 = new ElevatorButton();
        ElevatorButton b3 = new ElevatorButton();
        bool turn=true;

        //Asansörün yön değerlerini tanımla.
        enum direction
        {
            Up, Down
        }
```

```
//yön nesnesi oluştur.
direction which = new direction();
bool waitflag=false;
//Asansörün kat değerlerini tanımla.
enum floor
{
    first, second, third
}
//Kat nesnesi oluştur.
floor goingTo = new floor();

public ElevatorSimulation()
{
    InitializeComponent();
    log("Current State = " + p.CurrentState);
}

private void timer1_Tick(object sender, EventArgs e)
{
    //Hareket anında 2. kattan basışları algılama
    if (pictureBox1.Top > 175 && pictureBox1.Top < 185 &&(k2.click || b2.click))
    {
        if(p.CurrentState==ElevatorStateMachine.ElevatorState.Kat1)
        {log("Command.a: Current State=" + p.MoveNext(Command.a)); waitflag = true;}
        if (p.CurrentState == ElevatorStateMachine.ElevatorState.Kat3)
        {log("Command.b: Current State=" + p.MoveNext(Command.b)); waitflag = true;}
        goingTo = floor.second;

        floorStop();
    }

    else {
    switch (which)
    {
        case direction.Up:
            //Resmi yukarı oynat.
            pictureBox1.Top = pictureBox1.Top - 1;
            break;
        case direction.Down:
            //Resmi aşağı oynat.
            pictureBox1.Top = pictureBox1.Top + 1;
            break;
    }
    //katta dur.
    floorStop();
    }
}

public void floorStop()
{
    SoundPlayer zilsesi = new SoundPlayer(Properties.Resources.doorbell);
    switch (goingTo)
    {
        //birinci katta durulmuş.
        case floor.first:
            if (pictureBox1.Top > 360)
            {
                timer1.Stop();
                pictureBox1.Top = 360;
                zilsesi.Play();
                turn = true;

            }
            break;
        //ikinci katta durulmuş.
        case floor.second:
            if (pictureBox1.Top > 175 && pictureBox1.Top < 185)
```

```
            {
                timer1.Stop();
                pictureBox1.Top = 180;
                zilsesi.Play();
                turn = true;

            }
            break;
        //üçüncü katta durulmuş.
        case floor.third:
            if (pictureBox1.Top == 0)
            {
                timer1.Stop();
                pictureBox1.Top = 0;
                zilsesi.Play();
                turn = true;
            }
            break;
    }
    stater();
}
public void stater()
{
    if (turn == true) {
    switch (p.CurrentState)
    {
        case ElevatorStateMachine.ElevatorState.Kat3://Kat3'de
            {
                //aşağıdaki satırlar kata gelindiğinde butonları inaktif eder.
                k3.click = false; pictureBox3.Image = Properties.Resources.d3;
                b3.click = false; pb_b3.Image = Properties.Resources.d3;
                which = direction.Down;
                if( (k1.click && !k2.click) || (b1.click && !b2.click) )
                {
                    goingTo = floor.first;
                    log("Command.d: Current State=" + p.MoveNext(Command.d));//kat1'e direk gidiş.
                    log("Command.b: Current State=" + p.MoveNext(Command.b));
                    turn = false;
                    timer1.Start();
                    break;
                }
                if (k2.click || b2.click)
                {
                    which = direction.Down;
                    goingTo = floor.second;
                    log("Command.b: Current State=" + p.MoveNext(Command.b));//kat ikiye gidiş.
                    turn = false;
                    timer1.Start();
                    break;
                }
                break;
            }

        case ElevatorStateMachine.ElevatorState.Kat1://Kat1'de
            {
                k1.click = false; pictureBox4.Image = Properties.Resources.d3;
                b1.click = false; pb_b1.Image = Properties.Resources.d3;
                which = direction.Up;
                if ((k3.click && !k2.click) || (b3.click && !b2.click))
                {
                    log("Command.c: Current State=" + p.MoveNext(Command.c));//Birden 3'e direk
                    log("Command.a: Current State=" + p.MoveNext(Command.a));
                    turn = false;
                    goingTo = floor.third;
                    timer1.Start();
                    break;
                }
```

```
                    if (k2.click || b2.click)
                    {
                        goingTo = floor.second;
                        turn = false;
                        log("Command.a: Current State=" + p.MoveNext(Command.a));//Birden 2'ye

                        timer1.Start();
                        break;
                    }

                    break;

                }


            case ElevatorStateMachine.ElevatorState.Kat2Up:
                {
                    if (((k3.click || b3.click) || (k1.click || b2.click)) && (k2.click || b2.click)) {
waitflag = true; }
                    k2.click = false; pictureBox2.Image = Properties.Resources.d3;
                    b2.click = false; pb_b2.Image = Properties.Resources.d3;
                    if (waitflag == true) { System.Threading.Thread.Sleep(500); waitflag = false; }
                    if (k1.click || b1.click) {
                        which = direction.Down;
                        goingTo = floor.first;
                        log("Command.a: Current State=" + p.MoveNext(Command.b));
                        turn = false;
                        timer1.Start();
                        break;
                    }
                    if (k3.click || b3.click) {
                        which = direction.Up;
                        goingTo = floor.third;
                        log("Command.a: Current State=" + p.MoveNext(Command.a));
                        turn = false;
                        timer1.Start();
                        break;
                    }
                    break;

                }

        }

        }




    }

    private void pictureBox3_Click(object sender, EventArgs e)
    {
        k3.click = true;
        //log("3. Kattan Asansör Çağırıldı.");
        pictureBox3.Image = Properties.Resources.d4;
        if(turn) stater();

    }

    private void pictureBox2_Click(object sender, EventArgs e)
    {
        k2.click = true;
        //log("2. Kattan Asansör Çağırıldı.");
        pictureBox2.Image = Properties.Resources.d4;
        if(turn) stater();
    }
```

```
        private void pictureBox4_Click(object sender, EventArgs e)
        {
            k1.click = true;
            //log("1. Kattan Asansör Çağırıldı.");
            pictureBox4.Image = Properties.Resources.d4;
            if(turn) stater();
        }
        public void log(string s)
        {
            //log fonksiyonu listbox'a metin girmek amaçlı.
            lb_screen.Items.Add(s);
            //Aşağıdaki satırlar scroolbar'ı otomatik aşağı çeker.
            lb_screen.SelectedIndex = lb_screen.Items.Count - 1;
            lb_screen.SelectedIndex = -1;

        }
        private void pb_b3_Click(object sender, EventArgs e)
        {
            b3.click = true;
            //log("Asansörün 3. Tuşuna Basıldı.");
            pb_b3.Image = Properties.Resources.d4;
            if(turn) stater();
        }

        private void pb_b1_Click(object sender, EventArgs e)
        {
            b1.click = true;
            //log("Asansörün 1. Tuşuna Basıldı.");
            pb_b1.Image = Properties.Resources.d4;
            if(turn) stater();
        }

        private void pb_b2_Click(object sender, EventArgs e)
        {
            b2.click = true;
            //log("Asansörün 2. Tuşuna Basıldı.");
            pb_b2.Image = Properties.Resources.d4;
            if(turn) stater();


        }
    }
    //buton objeleri oluşturmak için sınıf tanımı.
    public class ElevatorButton
    {
        //Aşağıdaki komut bool property'e default false değeri vermek için.
        [System.ComponentModel.DefaultValue(false)]
        public bool click { get; set; }
    }
}
```
