

A dark blue vertical bar is on the left. A blue arrow points right from it, containing the date.

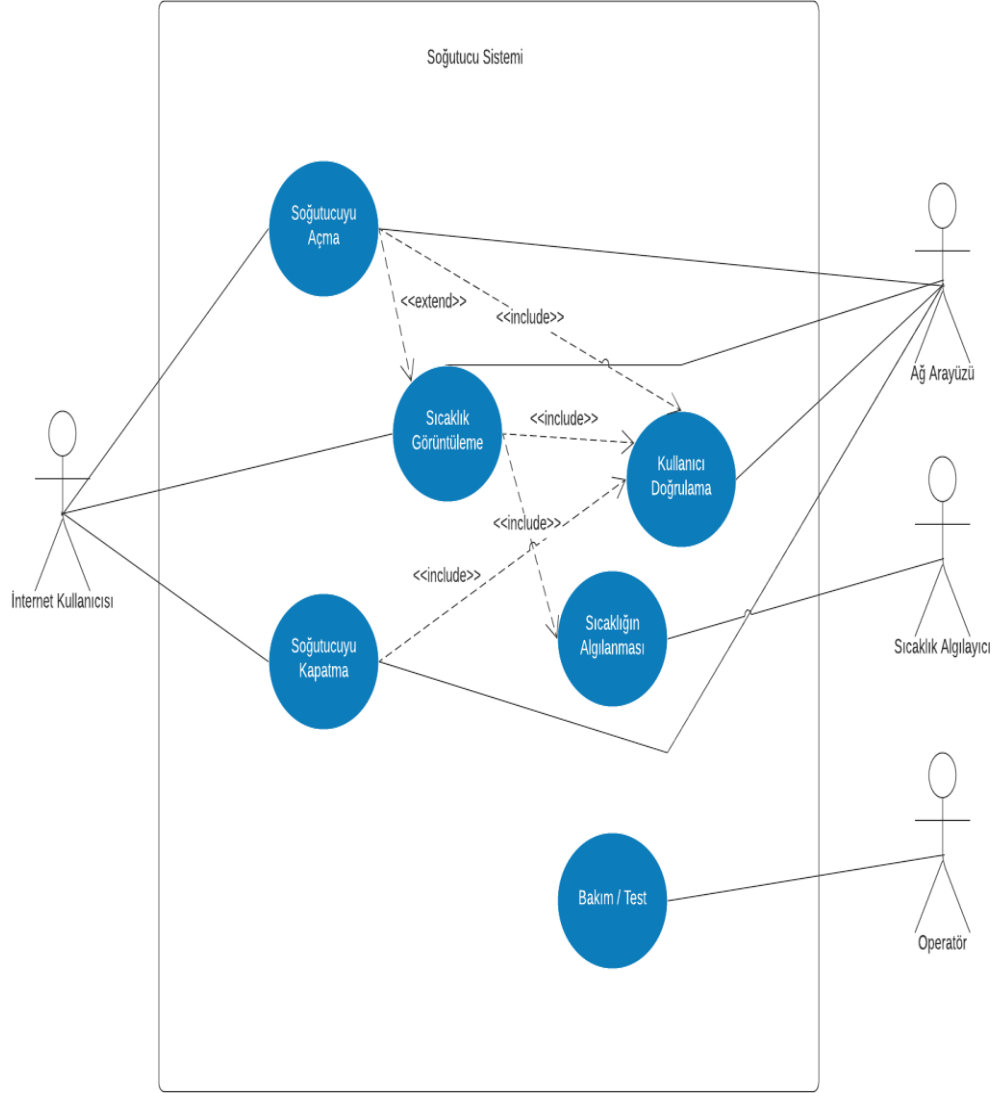
28.03.2020

DUHAN UZUN

B181210051 / 22B

duhan.uzun@ogr.sakarya.edu.tr

Several thin, curved lines in dark blue and light grey originate from the bottom left and curve upwards and to the right.



USE CASE – “SOĞUTUCUYU ÇALIŞTIRMA”

*SOĞUTUCUYU ÇALIŞTIRMA KULLANIM DURUMU

*EŞSİZ BİR AD: Soğutucuyu çalıştırma

*Soğutucuyu çalıştırmayı tanımlar

*20.03.2015, 30.03.2015 v6.6 duhan

*İLGİLİ AKTÖRLER : İnternet kullanıcısı

*GİRİŞ KOŞULU : Kullanıcı, sisteme giriş yapmıştır.

*ÇIKIŞ KOŞULU : İnternet kullanıcısı, soğutucu kullanımını yeterli görmüştür.

*ÖZEL GEREKSİNİMLER : İşlem gecikmesi en fazla 5 sn olmalı.

*OLAY AKIŞI :

*ANA OLAY AKIŞI :

1-Kullanıcı soğutucu aç sinyalini gönderir.

2- Çalıştırma sinyali ağ arayüzü üzerinden merkezi işlem birimine gönderilir.

3-Merkezi işlem birimi sinyali işleyip gerekli bilgiyi eyleyici modüle gönderir.

4-Eyleyici modül bilgiyi değerlendirip soğutucuyu çalıştırır.

*ALTERNATİF OLAY AKIŞI

A2-Soğutucu başlatılamadı (4)

5-Kullanıcıya “hata alındı, tekrar deniyoruz” mesajı gönderilir.

USE CASE – “SICAKLIK GÖRÜNTÜLEME”

*SICAKLIK GÖRÜNTÜLEME KULLANIM DURUMU

*EŞSİZ BİR AD: Sıcaklık görüntüleme

*Ortamın sıcaklığını kullanıcıya göstermeyi tanımlar

*20.04.2015, 30.05.2015 v6.6 duhan

*İLGİLİ AKTÖRLER : İnternet kullanıcısı

*GİRİŞ KOŞULU : Kullanıcı, sisteme giriş yapmıştır.

*ÇIKIŞ KOŞULU : İnternet kullanıcısı, soğutucudaki sıcaklık değerini okumuştur.

*ÖZEL GEREKSİNİMLER : İşlem gecikmesi en fazla 1 sn olmalı.

*OLAY AKIŞI :

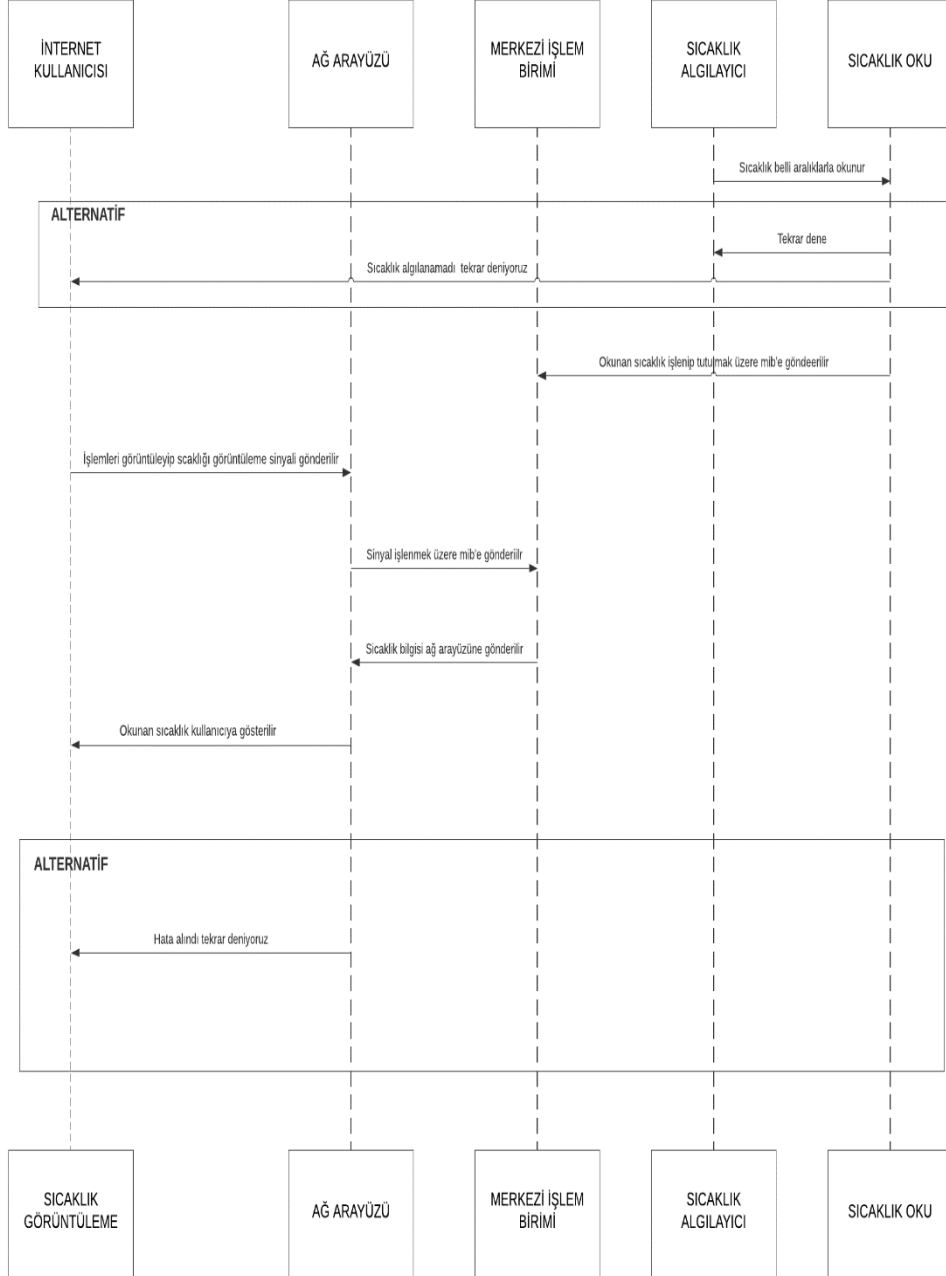
*ANA OLAY AKIŞI :

- 1-Sıcaklık algılayıcı ortamın sıcaklığını belli periyotlarla otomatik okur.
- 2-Okunan sıcaklık merkezi işlem birimine gönderilir ve gerekli bilgi burada tutulur.
- 3-Kullanıcı sıcaklık gönder sinyalini gönderir .
- 4-Oluşturulan sinyal ağ arayüzü üzerinden merkezi işlem birimine gönderilir.
- 5-Merkezi işlem biriminde sinyal değerlendirilip, tutulan bilgi ağ arayüzüne gönderilir.
- 6-Ağ arayüzü bilgiyi kullanıcıya gösterir.

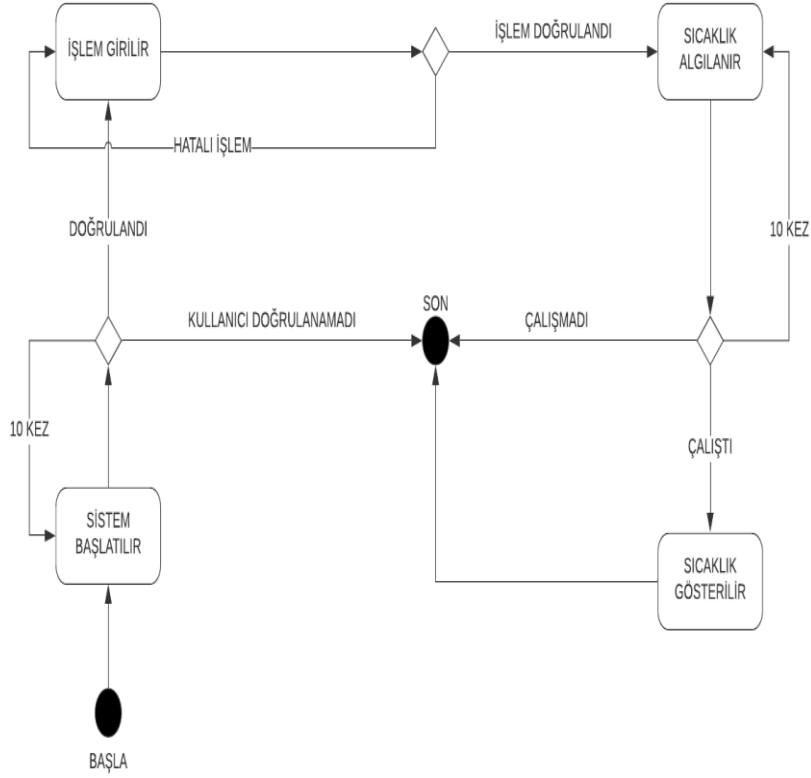
*ALTERNATİF OLAY AKIŞI

- A1-Ortamın sıcaklığı okunamadı (1)
- 2-Kullanıcıya “sıcaklık algılanamadı, tekrar deniyoruz” mesajı gönderilir.
- A2-Sıcaklık gösterilemedi (6)
- 7-Kullanıcıya “hata alındı, tekrar deniyoruz” mesajı gönderilir.

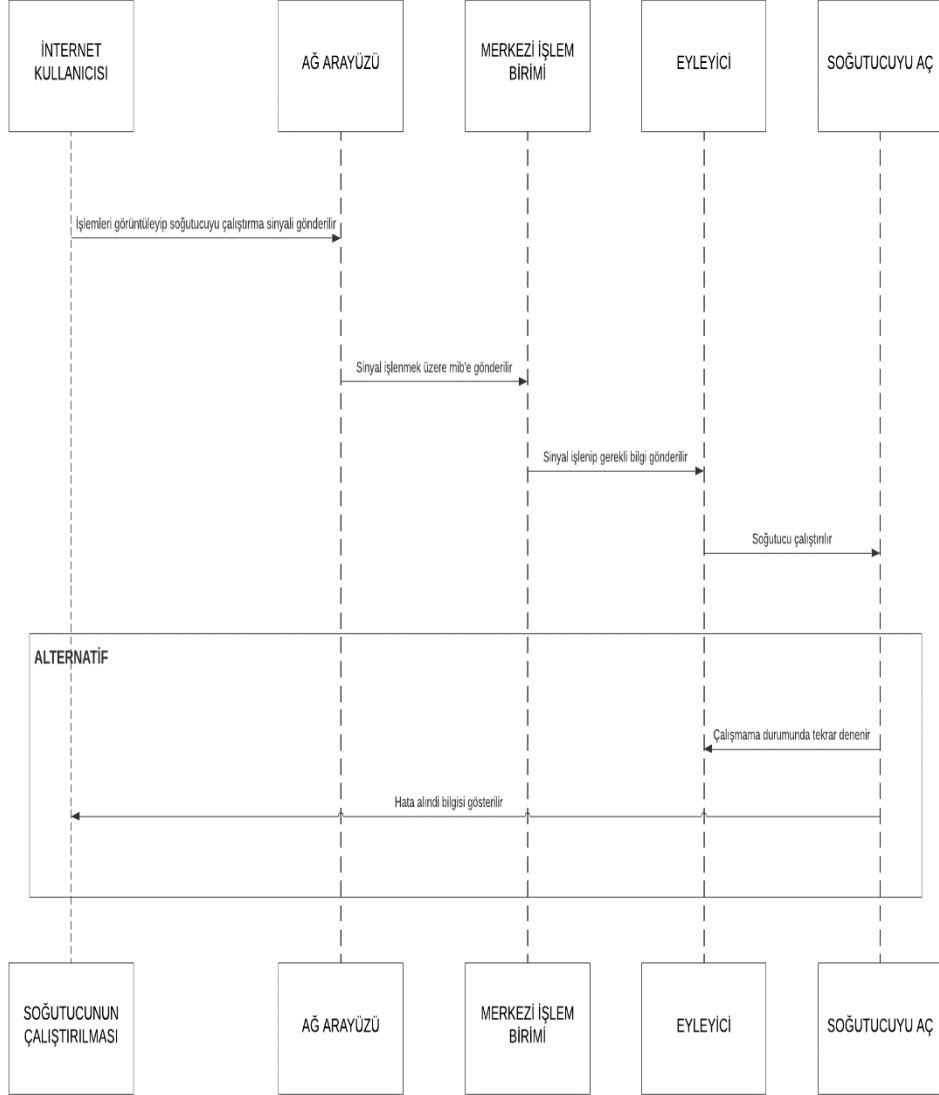
SICAKLIĞIN
GÖRÜNTÜLENMESİ
SIRALAMA ŞEMASI



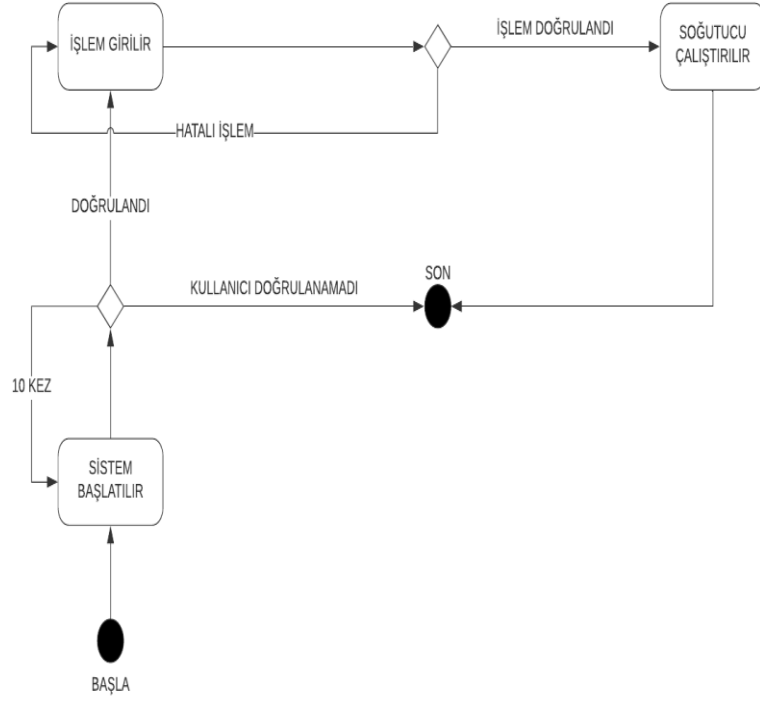
SICAKLIĞIN GÖRÜNTÜLENMESİ ETKİNLİK ŞEMASI



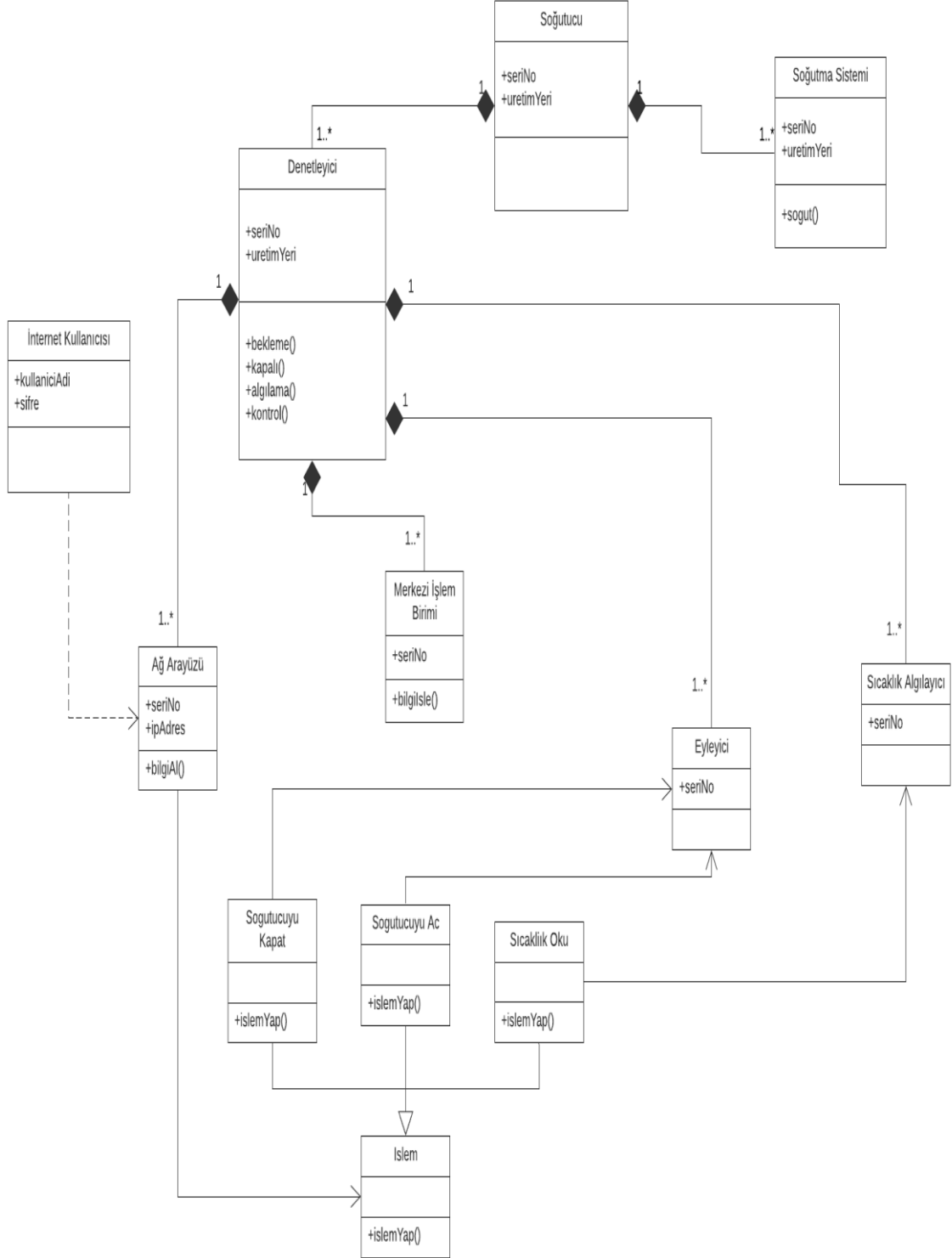
SOĞUTUCUNUN
ÇALIŞTIRILMASI
SIRALAMA ŞEMASI



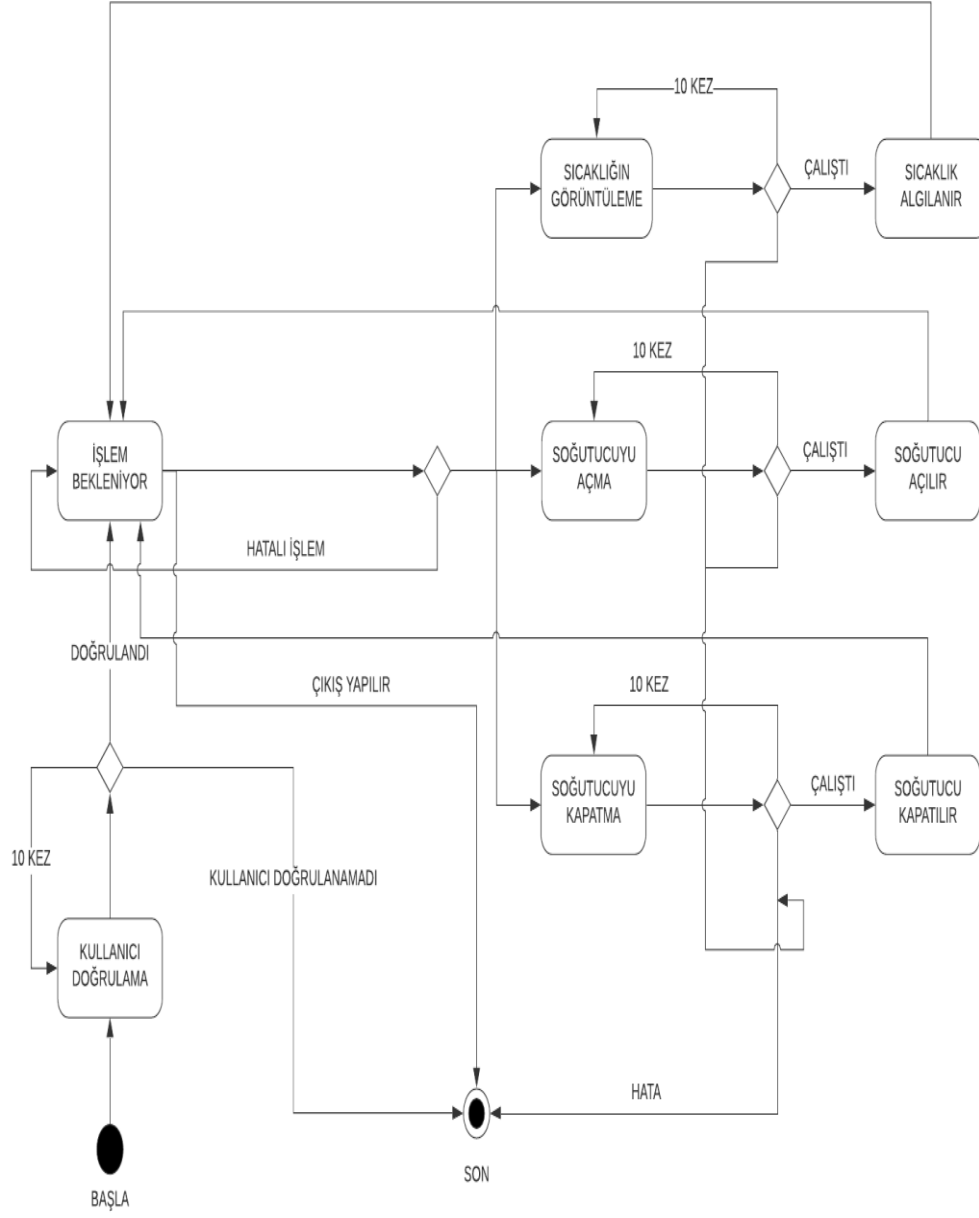
SOĞUTUCUNUN
ÇALIŞTIRILMASI ETKİNLİK
ŞEMASI



SINIF ŞEMASI



DURUM DİYAGRAMI



KULLANICI DOĞRULAMA EKRANI VE AÇIKLAMASI

```
do{
    System.out.print("Kullanici adinizi girin: \n");
    Scanner girdi = new Scanner(System.in);

    ad=girdi.nextLine();

    System.out.print("Sifrenizi girin: \n");
    Scanner girdi2 = new Scanner(System.in);

    sifre=girdi2.nextLine();

    while(rs.next())
    {
        kullaniciBilgileri.add(rs.getString("KullaniciAdi"));
        kullaniciBilgileri.add(rs.getString("Sifre"));
    }

    for(int i = 0; i < kullaniciBilgileri.size(); i++ )
    {
        if(ad.equals(kullaniciBilgileri.get(i)))
        {
            if(sifre.equals(kullaniciBilgileri.get(i+1)))
            {
                System.out.println("Bilgiler dogrulandi, lutfen bekleyin...");
                kontrol = 1;
            }
        }
    }
}
```

Önce veritabanına bağlanıyoruz, veritabanından kullanıcı adı ve şifreler çekilerek bir arrayliste atıyoruz sonra kullanıcıdan gerekli bilgiler alındıktan sonra doğrulama yapmak için arraylistte tuttuğumuz veriler üzerinde gezdiriyoruz ve kontrol ediyoruz. Eğer kullanıcı adı ve şifre veritabanında varsa sistemi devam ettiriyoruz yoksa yeniden giriş yapmasını istiyoruz.

SICAKLIĞIN GÖRÜNTÜLENMESİ EKRANI VE AÇIKLAMASI

```
27     }
28
29
30     public void SicaklikOku ()
31     {
32         if(calismaDurumu == 1)
33         {
34
35             System.out.println("Sicaklik bilgisi sicaklik algilayicidan merkezi islem birimine iletildi");
36             MerkeziIslemBirimi.merkeziIslemBirimineBaglan().sicaklikBilgisiniCevapla();
37
38             sicaklik = r.nextInt(35);
39         }
40
41         else
42         {
43             System.out.println("Sicaklik algılanamadi, tekrar deniyoruz");
44         }
45     }
46
47
48     public int SicakligiEkranaYaz()
49     {
50         return sicaklik;
51     }
52
```

Sıcaklık görüntülenmek istendiğinde SicaklikOku() fonksiyonu çağrılır ve bu fonksiyon 0-35 arasında rastgele bir değer üretip ürettiği değeri sıcaklık değişkenine atar. Daha sonra sıcaklık ekrana yazdırılmak istendiğinde SicakligiEkranaYaz() fonksiyonu çağrılır ve bu fonksiyon rastgele üretilen sıcaklık değerini döndürür.

SOĞUTUCUNUN AÇILMASI EKRANI VE AÇIKLAMASI

```
@Override
public void sogutucuyuAcBilgisiniIlet()
{
    if(calismaDurumu == 1)
    {
        if(durum != 1)
        {
            System.out.println("sogutucu aciliyor");
            durum = 1;
        }
        else
        {
            System.out.println("Sogutucu zaten acik, lutfen baska bir islem deneyiniz");
        }
    }

    else
    {
        System.out.println("Hata alindi, tekrar deniyoruz");
    }
}
```

Merkezi işlem biriminden eyleyiciye gerekli bilgi geldiğinde soğutucuyuAcBilgisiniIlet() fonksiyonu çağrılır ve bu fonksiyon öncelikle çalışma ihtimalini kontrol eder yani eğer sistem bozursa kullanıcıya mesaj gönderilir. Sonra soğutucunun çalışır durumda olup olmadığını kontrol eder, eğer çalışıyorsa sistem uyarı verir, çalışmıyorsa soğutucu çalıştırılır.

SOĞUTUCUNUN KAPATILMASI EKRANI VE AÇIKLAMASI

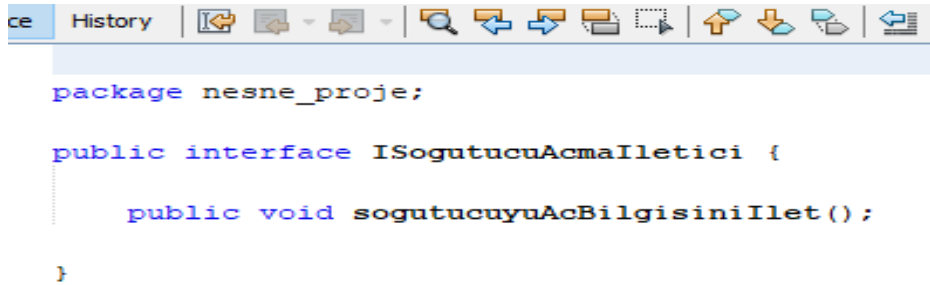
```
@Override
public void sogutucuyuKapatBilgisiniIlet()
{
    if(durum != 0)
    {
        System.out.println("sogutucu kapatiliyor");
        durum = 0;
    }

    else
    {
        System.out.println("Sogutucu zaten kapali, lutfen baska bir islem deneyiniz ");
    }
}
```

Merkezi işlem biriminden eyleyiciye gerekli bilgi geldiğinde `sogutucuyuKapatBilgisiniIlet()` fonksiyonu çağrılır ve bu fonksiyon soğutucunun çalışır durumda olup olmadığını kontrol eder, eğer kapalıysa sistem uyarı verir, çalışıyorsa soğutucu kapatılır.

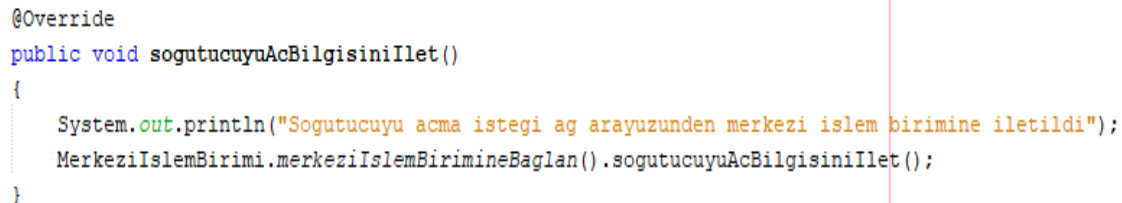
OPEN/CLOSED İLKESİ VE GERÇEKLEMESİ

OPEN/CLOSED İlkesi mevcut kodu değiştirmeden yeni kodlar ekleyerek, yeni özellikler eklemeyi sağlar ve bunu modülün kendisi yerine soyutunu kullanarak yapabiliriz.



```
ce History | [Icons] |  
  
package nesne_proje;  
  
public interface ISogutucuAcmaIletici {  
    public void sogutucuyuAcBilgisiniIlet();  
}
```

Mesela soğutucuyu açma bilgisi birçok farklı yerde farklı farklı şekillerde kullanıldığı için bu fonksiyonu soyutladık.



```
@Override  
public void sogutucuyuAcBilgisiniIlet()  
{  
    System.out.println("Sogutucuyu acma istegi ag arayuzunden merkezi islem birimine iletildi");  
    MerkeziIslemBirimi.merkeziIslemBirimineBaglan().sogutucuyuAcBilgisiniIlet();  
}
```

Ve gerekli yerlerde override ederek bu fonksiyonu kullanmış olduk, mesela fonksiyon burda mib'e iletim ile ilgili bir mesaj yazarken başka bir yerde farklı bir mesaj yazabilir. Bu da bize kodun genişlemesini sağlar.

SINGLETON DESENİ VE GERÇEKLEMESİ

Singleton tasarım deseninin amacı bir sınıfın yalnızca tek nesnesi olmasını ve bu nesneye global olarak erişilmesini sağlamaktır.

```
public class Eyleyici implements ISogutucuAcmaIletici,  
  
    private static Eyleyici eyleyici;  
    private int durum = 0;  
  
    private int calismaDurumu = 1;  
  
    private Eyleyici()  
    {  
  
    }  
  
    public static Eyleyici eyleyiciyeBaglan()  
    {  
        if(eyleyici == null){  
            eyleyici = new Eyleyici();  
        }  
  
        return eyleyici;  
    }
```

Eyleyici sınıfından örnek verecek olursak sınıfın yapıcı metodu private olarak tanımlanmış, bu şekilde tek nesne oluşturulabilmesini sağlıyoruz.

Yapıcı gibi çalışan statik bir metod tanımlıyoruz ve böylelikle global olarak erişiliyor.

OBSERVER DESENİ VE GERÇEKLEMESİ

Observer tasarım desenin amacı bir nesnede meydana gelen değişikliği onunla bağlantısı olan başka nesneye veya nesnelere bildirilmesi, işlem yapılması vs..

```
break;
case 4:

    System.out.println("---Sistem Bilgileri---");

    p.attach(Eyleyici.eyleyiciyeBaglan());
    p.attach(SicaklikAlgilayici.sicaklikAlgilayicisinaBaglan());
    MerkeziIslemBirimi.merkeziIslemBirimineBaglan().publisher = p;
    MerkeziIslemBirimi.merkeziIslemBirimineBaglan().calismaDurumuDuzenle("pasif");
    p.detach(Eyleyici.eyleyiciyeBaglan());
    p.detach(SicaklikAlgilayici.sicaklikAlgilayicisinaBaglan());

    cikis = 1;
```

IObserver, ISubject ve Publisher sınıflarımız ve arayüzlerimiz bu observer tasarım deseni için kullanılmıştır. Bildirim göndereceğimiz sınıfları Publisher sınıfına subscriber olarak ekliyoruz. Merkezi işlem birimini yönetim kısmına atıyoruz ve kullanıcı sistemden çıkış yaptığında eyleyici ve algılayıcı durumlarını pasif hale getiriyor.

KAYNAK KODLARI VE VİDEO LİNKİ

Kaynak kodlar: https://github.com/duhan18/nesne_proje

Anlatım Videosu: <https://www.youtube.com/watch?v=IRWjvTZdgRM>