T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ÖDEV BAŞLIĞI

B181210393 - Melih Ensar BARIŞIK

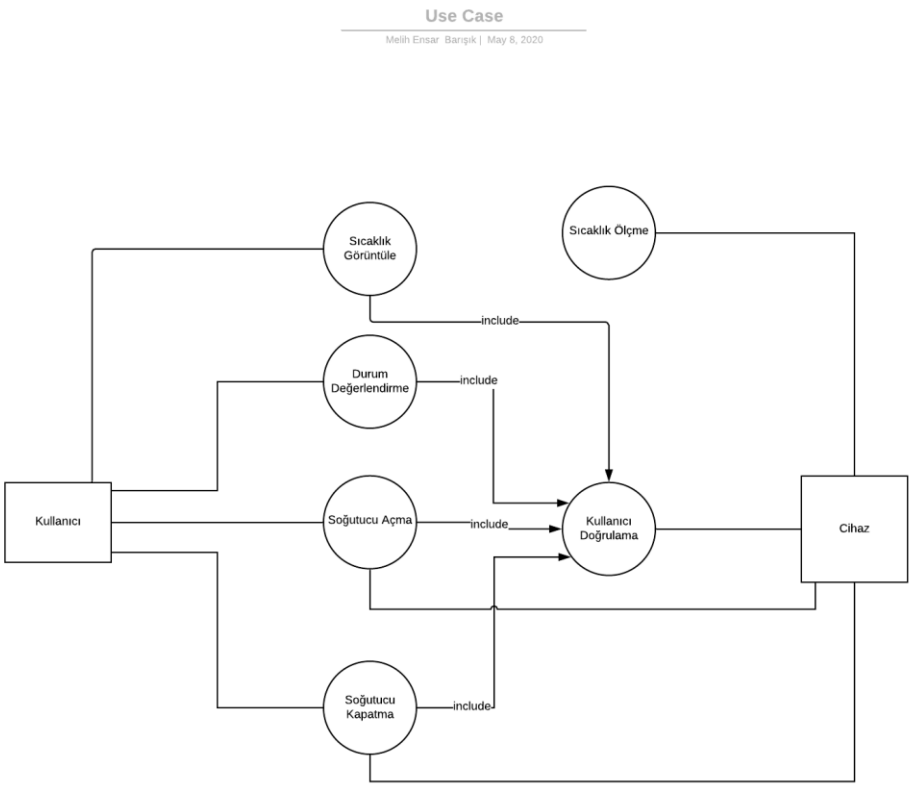
2.Öğrenim B Sınıfı

ensar.barisik@ogr.sakarya.edu.tr

SAKARYA

Mayıs, 2020

A.İnternet kullanıcısı aktörü için kullanım durumu (Use Case) diyagramı.



B. İnternet üzerinden “sıcaklığın görüntülenmesi” ve “soğutucunun çalıştırılması” kullanım durumlarına ait metinsel tanımlar.

B1.Sıcaklık Görüntüleme

Sıcaklık görüntüleme işlemi kullanıcının anlık sıcaklık değerini görmek istemesidir. Görüntüleme isteği öncelikle ağ arayüzü tarafından alınır. Alınan istek ağ arayüzü aracılığı ile merkezi işlem birimine daha sonra buradan sıcaklık algılayıcı birime iletilir. Sıcaklık algılayıcı birim sıcaklığı hesaplar ve hesaplanan değer merkezi işlem birimine iletilir. Anlık sıcaklık merkezi işlem biriminden ağ arayüzü yardımı ile kullanıcıya gösterilir.

Hazırlayan: Melih Ensar Barışık, Sürüm: 1.0 Tarih: 24/03/2020

İlgili Aktörler: Kullanıcı, Merkezi İşlem Birimi ve Sıcaklık Algılayıcı Birimi

Giriş Koşulu: Kullanıcı tarafından sıcaklık değerinin istenmesi.

Çıkış Koşulu: Kullanıcıya sıcaklığın gösterilmesi.

Ana Olay Akışı:

1.Ağ arayüzü kullanıcı adı ve şifre bilgilerini ister.

2.Kullanıcı adı ve şifre bilgileri kullanıcı tarafından girilir.

3.Girilen kullanıcı adı ve şifre bilgileri veri tabanı tabanından kontrol edilir.

4.Girilen bilgiler onaylanır.

5.Kullanıcı menüsü ağ arayüzü tarafından kullanıcıya gösterilir.

6. Kullanıcı ağ arayüzü kullanarak sıcaklığı görüntüle seçeneğini seçer.

7.Kullanıcının seçeneği ağ arayüzü tarafından alınır ve merkezi işlem birimine iletilir.

8. Merkezi işlem birimi sıcaklık hesaplanması için sıcaklık algılayıcı birime komut gönderir.

9.Sıcaklık algılayıcı birim sıcaklığı hesaplar.

10.Sıcaklık algılayıcı birim hesaplanan sıcaklık değerini merkezi işlem birimine iletir.

11.Hesaplanan sıcaklık değeri ağ arayüzü ile kullanıcıya gösterir.

Alternatif Olay Akışları

A1.Kullanıcı Adı ve Şifre Hatalı İse

4.Girilen bilgiler onaylanamaz.

5.Kullanıcıya 3 kez deneme hakkı verilir.

6.Kullanıcı doğru değer giremez ise program kapanır.

B2.Soğutucu Açma

Soğutucu açma işlemi kullanıcının cihazı aktif etme işlemidir. Soğutucu açma işlemi öncelikle ağ arayüzü yardımı ile kullanıcıdan alınır. Alınan istek ağ arayüzü aracılığı ile merkezi işlem birimine daha sonra buradan eyleyici birime gönderilir. Eyleyici birim eğer soğutucu kapalı konumda ise soğutucu açar ve bilgilendirme mesajını merkezi işlem birimi üzerinden ağ arayüzü yardımı ile kullanıcıya gösterir. Soğutucu açık konumda ise eyleyici bunun ile ilgili bir uyarı mesajını merkezi işlem birimi üzerinden ağ arayüzü yardımı ile kullanıcıya gösterir.

Hazırlayan: Melih Ensar Barışık Sürüm: 1.1 Tarih: 18.03.2020

İlgili Aktörler: Kullanıcı ve Eyleyici

Giriş Koşulu: Kullanıcı tarafından ağ ara yüzü kullanılarak soğutucunun açılmak istenmesi.

Çıkış Koşulu: Eyleyicinin soğutucuyu açması.

Ana Olay Akışı:

1.Ağ arayüzü kullanıcı adı ve şifre bilgilerini ister.

2.Kullanıcı adı ve şifre bilgileri kullanıcı tarafından girilir.

3.Girilen kullanıcı adı ve şifre bilgileri veri tabanı tabanından kontrol edilir.

4.Girilen bilgiler onaylanır.

5.Kullanıcı menüsü ağ arayüzü tarafından kullanıcıya gösterilir.

6.Kullanıcı ağ arayüzünü kullanarak soğutucu açma seçeneğini seçer.

7.Kullanıcının seçeneği ağ arayüzü tarafından alınır ve merkezi işlem birimine iletilir.

8. Merkezi işlem birimi soğutucuyu başlatması için eyleyiciye komut gönderir.

9.Eyleyici birim soğutucuyu açar.

10.Soğutucu açıldı mesajı merkezi işlem birimine iletilir.

11. Soğutucu açıldı mesajı ağ arayüzü ile kullanıcıya gösterilir.

Alternatif Olay Akışları

A1.Kullanıcı Adı ve Şifre Hatalı İse

4.Girilen bilgiler onaylanamaz.

5.Kullanıcıya 3 kez deneme hakkı verilir.

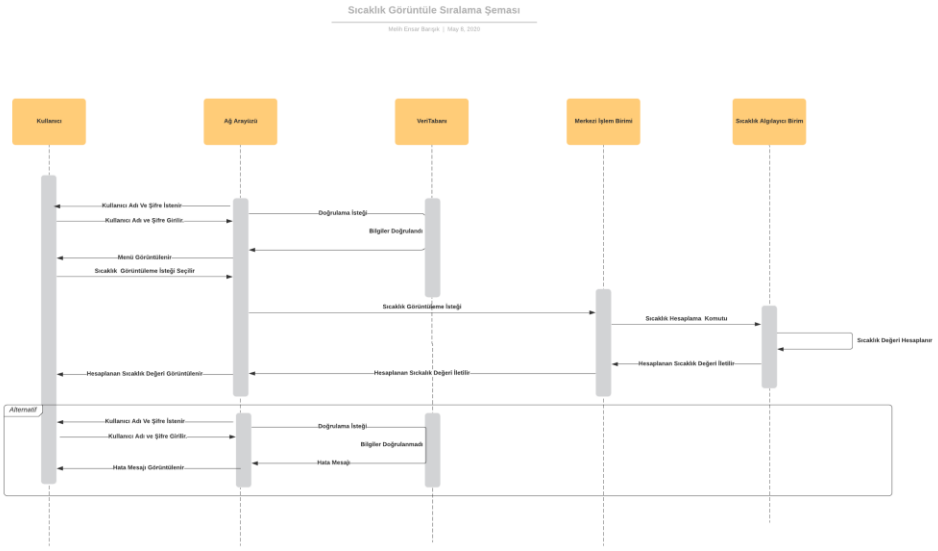
6.Kullanıcı doğru değer giremez ise program kapanır.

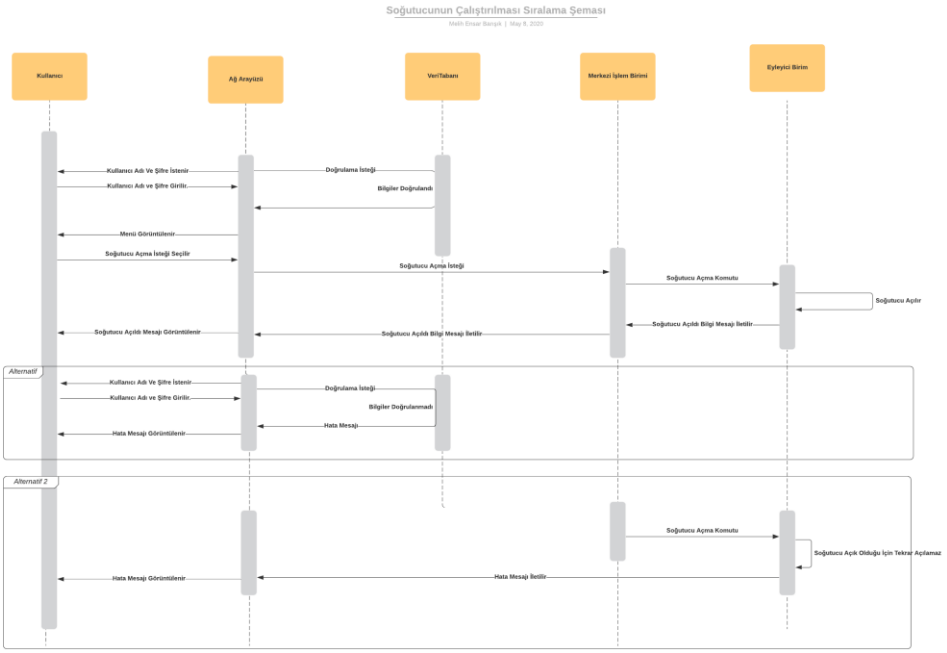
A2.Soğutucu Cihaz Açık Konumda İse

8.Eyleyici soğutucunun açık olduğunu içeren mesajını merkezi işlem birimine iletilir.

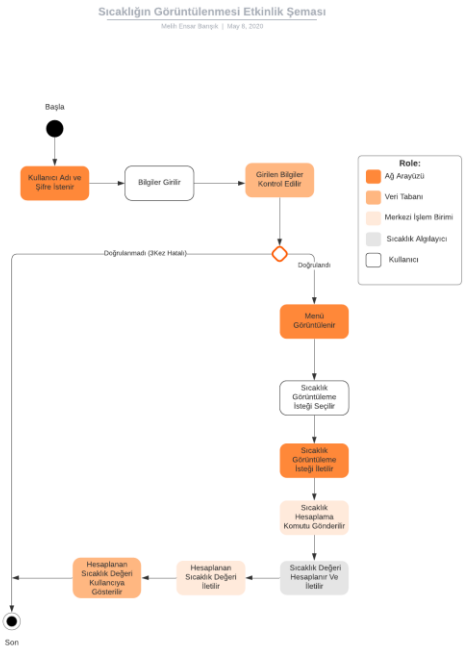
9.İletilen mesaj ağ arayüzü ile kullanıcıya gösterilir.

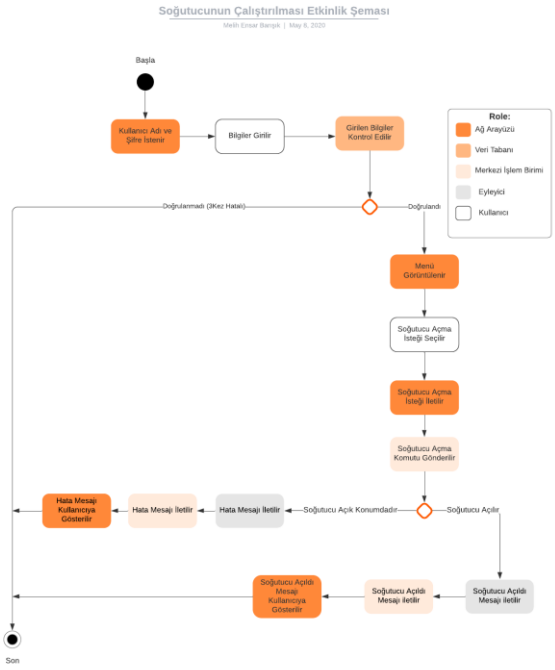
C. İnternet üzerinden “sıcaklığın görüntülenmesi” ve “soğutucunun çalıştırılması” kullanım durumlarına ait sıralama şemaları.



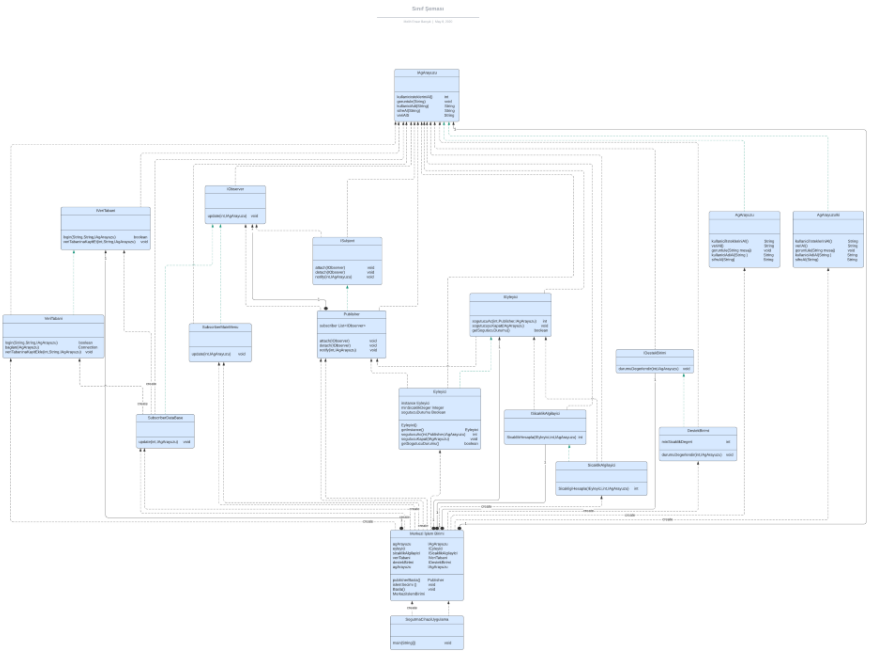


D. İnternet üzerinden “sıcaklığın görüntülenmesi” ve “soğutucunun çalıştırılması” kullanım durumlarına ait etkinlik şemaları.

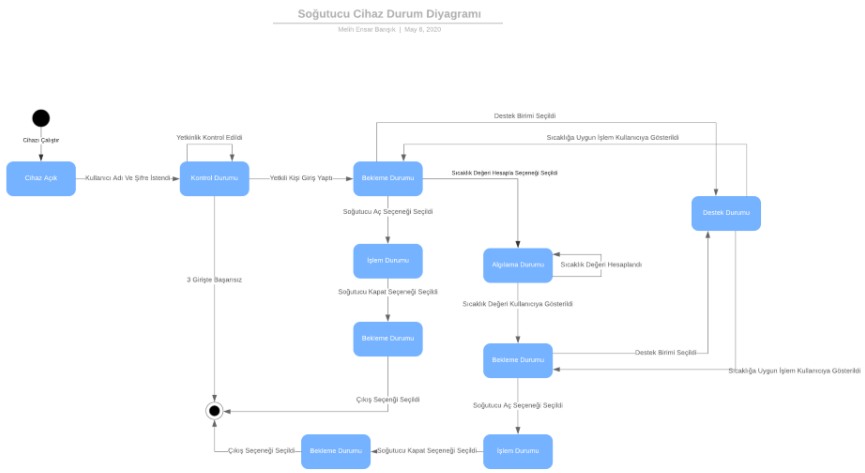




E. Geliştirdiğim sistemin sınıf şeması.



F. Sistemin durum diyagramı.



G. Kullanıcı doğrulama ekranı ve açıklaması.

Uygulama başlatıldığı zaman karşımıza bizden kullanıcı adı ve şifreyi isteyen bir ekran çıkıyor. Çıkan ekranda önce kullanıcı adımızı daha sonra şifremizi sırası ile girmemiz isteniyor.

```
Denetleyici Sistemi
Kullanıcı Adınızı Giriniz :Admin
Şifrenizi Giriniz :
```

Girilen bilgileri kontrol etmek için önce kullanıcı adı ve şifrelerin depolandığı veri tabanına erişim sağlanıyor.

```
connection = DriverManager.getConnection( url: "jdbc:postgresql://localhost:5432/YetkiliKisiler", user: "postgres", password: "12345");
```

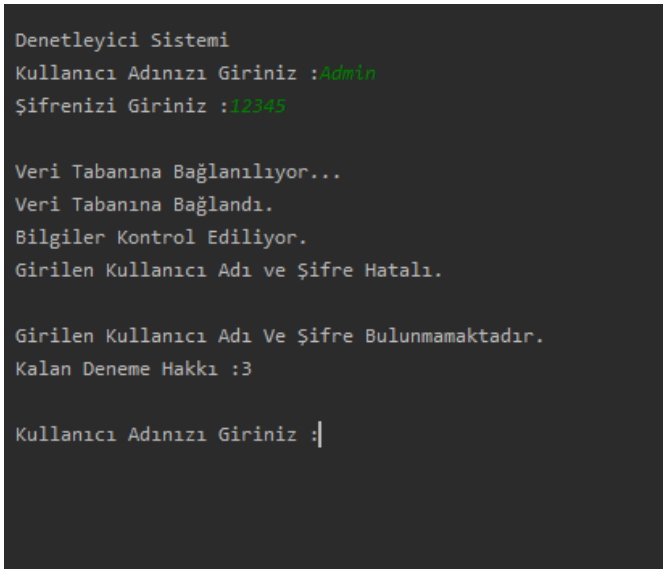
Girilen bilgilerin bağlanılan veri tabanında olup olmadığı kontrol ediliyor.

```
String sqlKomut = "select * from \"Yetkililer\".\"Kisiler\" where \"kullaniciAdi\"=? and sifre=?";
```

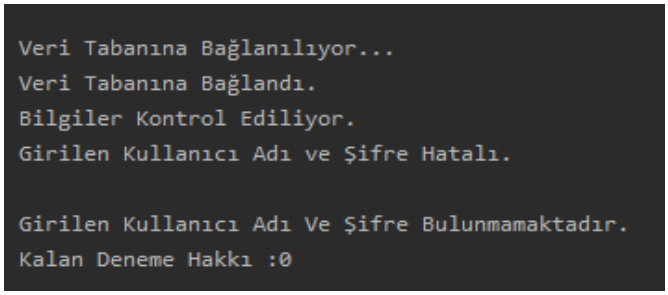
Girilen bilgiler veri tabanında var ise asıl işlemlerimizin gerçekleştiği menümüz ekrana yazdırlıyor.

```
"C:\Program Files\Java\jdk-11.0.6\bin\java.exe" "-javaagent:

Denetleyici Sistemi
Kullanıcı Adınızı Giriniz :Admin
Şifrenizi Giriniz :123

Veri Tabanına Bağlanılıyor...
Veri Tabanına Bağlandı.
Bilgiler Kontrol Ediliyor.
Kullanıcı Adı ve Şifre Doğrulandı.

Ana Menu
1-Sıcaklık Değerini Hesapla     2-Soğutucuyu Aç
3-Soğutucuyu Kapat              4-Durumu Değerlendir
5-Çıkış
Seçim :
```

Eğer girilen bilgiler veri tabanındaki bilgiler ile uyuşmuyor ise kullanıcıya 3 hak daha tanınıyor.



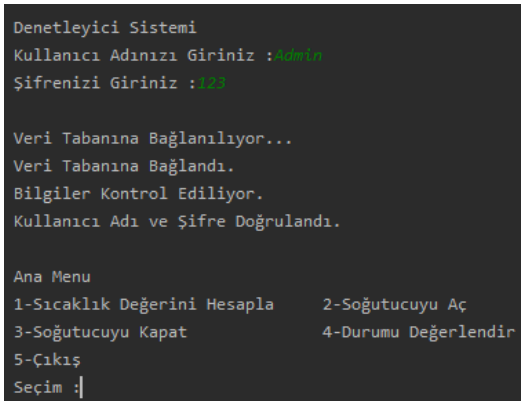
Verilen 3 haktan herhangi birinde kullanıcı doğru bir kullanıcı adı ve şifre giremez ise uygulama kendini kapatıyor.



H. Sıcaklığın görüntülenmesi ve soğutucunun açılıp kapatılmasıyla ilgili ekran görüntüleri ve açıklaması.

H1.Sıcaklığın Görüntülenmesi

Kullanıcı doğru kullanıcı adı ve şifre ile giriş yaptıktan sonra karşısına bir menü çıkıyor.



Çıkan menüde 1 yani sıcaklık değerini hesapla seçeneği seçildiği zaman merkezi işlem birimi sıcaklık algılayıcı birime sıcaklığı hesaplaması için istek gönderiyor.

```
case SICAKLIK_HESAPLA:
    anlikSicaklik = sicaklikAlgilayici.SicakligiHesapla(eyleyici,anlikSicaklik,agArayuzuIki);

    agArayuzuIki.goruntule( mesaj: "Cihazin Anlık Sıcaklığı :");
    agArayuzuIki.goruntule(String.valueOf(anlikSicaklik));

    agArayuzu.goruntule( mesaj: "\n \n");
    break;
```

Sıcaklık algılayıcı birimin içinde bulunan SicaklikHesapla fonksiyonunu çağıran bu istek sıcaklık algılayıcı birimin içinde devam ediyor.

```
@Override
public int  SicakligiHesapla(IEyleyici eyleyici,int sicaklik,IAgArayuzu agArayuzu)
{
    if(eyleyici.getSogutucuDurumu() ==true)
    {

        agArayuzu.goruntule( mesaj: "Sıcaklık Algılayıcı Çalıştı.");
        Random random = new Random();
        sicaklik = random.nextInt( bound: 3)-(18);
    }
    else
    {

        Random random = new Random();
        sicaklik = random.nextInt( bound: 5)-(13);
    }


    return sicaklik;
}
```

Benim yazdığım senaryoya uygun olarak soğutucu cihazın açık olup olmaması hesaplanacak rastgele sıcaklık değerini etkiliyor. Senaryomda soğutucunun çalıştığı odanın ortalama –18 civarında olması gerekmekte. Soğutucu açıldığı zaman ortam sıcaklığını -18 yapıyor. Soğutucumuz açık iken sıcaklık değerini hesaplamak istersek sıcaklık değeri rastgele olarak –16,-17,-18 dereceden biri oluyor. Bunun amacı soğutucu açıkken sıcaklığın çok değişmeyeceğidir.

```
if(eyleyici.getSogutucuDurumu() ==true)
{

    agArayuzu.goruntule( mesaj: "Sıcaklık Algılayıcı Çalıştı.");
    Random random = new Random();
    sicaklik = random.nextInt( bound: 3)-(18);
}
```

İkinci durumda soğutucu kapalı iken sıcaklık hesaplanmak isterse ise soğutucu daha yeni kapanmış olabileceği için hesaplanacak rastgele değer -18,-17 olabileceği gibi soğutucunun kapatılması üzerinden vakit geçebileceği için oluşturulacak rastgele değer -10 dereceye kadar düşebilmektedir.

```
else
{
    agArayuzu.goruntule( mesaj: "Sıcaklık Algılayıcı Çalıştı.");
    Random random = new Random();
    sicaklik = random.nextInt( bound: 8)-(18);
}
```

İki durumun sonucunda da hesaplanan sıcaklık değerimiz aynı fonksiyon içinde merkezi işlem birimine gerçi çevrilip burada ağ arayüzündeki goruntule fonksiyonu ile kullanıcıya gösteriliyor.

```
agArayuzuIki.goruntule( mesaj: "Cihazin Anlık Sıcaklığı :");
agArayuzuIki.goruntule(String.valueOf(anlikSicaklik));
```

H2.Soğutucunun Açılması

Kullanıcı doğru kullanıcı adı ve şifre ile giriş yaptıktan sonra karşısına bir menü çıkıyor.

```
Denetleyici Sistemi
Kullanıcı Adınızı Giriniz :Admin
Şifrenizi Giriniz :123

Veri Tabanına Bağlanılıyor...
Veri Tabanına Bağlandı.
Bilgiler Kontrol Ediliyor.
Kullanıcı Adı ve Şifre Doğrulandı.

Ana Menu
1-Sıcaklık Değerini Hesapla     2-Soğutucuyu Aç
3-Soğutucuyu Kapat              4-Durumu Değerlendir
5-Çıkış
Seçim :
```

Çıkan menüde 2 yani soğutucuyu aç seçeneği seçildiği zaman merkezi işlem birimi üzerinden eyleyici birime soğuyucuyu açması komutu gönderiliyor.

```
case SOGUTUCU_ACMA:
    anlikSicaklik = eyleyici.sogutucuyuAc(anlikSicaklik,p,agArayuzuIki);
    break;
```

Olay eyleyici birimin içinde bulunan sogutucuyuAc fonksiyonu ile devam ediyor. Soğutucu cihazımız program başladığı zaman kapalı konumda bulunuyor.

```
public int sogutucuyuAc(int guncelSicaklikDegeri,Publisher p,IAgArayuzu agArayuzuIki) throws SQLException {

    if(sogutucuDurumu == false)
    {
            agArayuzuIki.goruntule( mesaj: "Soğutucu Cihaz Açılıyor. \n");
            agArayuzuIki.goruntule( mesaj: "Sıcaklık Değeri Ayarlanıyor... \n \n");
           while(guncelSicaklikDegeri >= minSicaklikDegeri-3)
           {
               guncelSicaklikDegeri--;
           }
           agArayuzuIki.goruntule( mesaj: "Sıcaklık Değeri Ayarlandı. \n");

           p.notify(guncelSicaklikDegeri,agArayuzuIki);
           sogutucuDurumu=true;


    }
    else
    {
        agArayuzuIki.goruntule( mesaj: "Soğutucu Zaten Cihaz Açık Konumda. \n \n");

    }
    return guncelSicaklikDegeri;

}
```

Soğutucu açma isteği geldiği zaman soğutucu cihaz kapalı durumda ise cihaz açılıyor ve bu isteğin gerçekleştiği ile ilgili bilgi ağ arayüzü ile kullanıcıya gösteriliyor. Soğutucu cihaz açık durumuna getiriliyor. Senaryoma uygun olarak soğutucu açıldığı zaman sıcaklık değeri kademeli olarak -18 derece oluyor.

```
if (sogutucuDurumu == false) {
    agArayuzuIki.goruntule( mesaj: "Soğutucu Cihaz Açılıyor. \n");
    agArayuzuIki.goruntule( mesaj: "Sıcaklık Değeri Ayarlanıyor... \n \n");
    while (guncelSicaklikDegeri >= minSicaklikDegeri - 3) {
        guncelSicaklikDegeri--;
    }
    agArayuzuIki.goruntule( mesaj: "Sıcaklık Değeri Ayarlandı. \n");

    p.notify(guncelSicaklikDegeri, agArayuzuIki);
    sogutucuDurumu = true;
```

Eğer istek yapıldığı zaman soğutucu cihazımız zaten açık durumda ise gene ağ arayüzü yardımı ile kullanıcımıza soğutucunun açık konumda olduğu bildiriliyor.

```
} else {
    agArayuzuIki.goruntule( mesaj: "Soğutucu Cihaz Zaten Açık Konumda. \n \n");
```

H3.Soğutucunun Kapatılması

Kullanıcı doğru kullanıcı adı ve şifre ile giriş yaptıktan sonra karşısına bir menü çıkıyor.

```
Denetleyici Sistemi
Kullanıcı Adınızı Giriniz :Admin
Şifrenizi Giriniz :123

Veri Tabanına Bağlanılıyor...
Veri Tabanına Bağlandı.
Bilgiler Kontrol Ediliyor.
Kullanıcı Adı ve Şifre Doğrulandı.

Ana Menu
1-Sıcaklık Değerini Hesapla     2-Soğutucuyu Aç
3-Soğutucuyu Kapat              4-Durumu Değerlendir
5-Çıkış
Seçim :
```

Çıkan menüde 3 yani soğutucuyu kapat seçeneği seçildiği zaman merkezi işlem birimi üzerinden eyleyici birime soğuyucuyu kapaması komutu gönderiliyor.

```
case SOGUTUCU_KAPATMA:
    eyleyici.sogutucuyuKapat(agArayuzuIki);
    break;
```

Olay eyleyici birimin içinde bulunan sogutucuyuKapat fonksiyonu ile devam ediyor. Soğutucu cihazımız program başladığı zaman kapalı konumda bulunuyor.

```
public void sogutucuyuKapat(IAgArayuzu agArayuzuIki) {

    if (sogutucuDurumu == true) {
        agArayuzuIki.goruntule( mesaj: "Soğutucu Başarılı Bir Şekilde Kapatıldı. \n \n");

        sogutucuDurumu = false;
    } else {
        agArayuzuIki.goruntule( mesaj: "Soğutucu Zaten Kapalı Konumda. \n \n");

    }


}
```

Soğutucu kapatma isteği geldiği cihaz açık durumda ise soğutucu cihaz kapatılıyor ve bu isteğin gerçekleştiği ile ilgili bilgi ağ arayüzü ile kullanıcıya gösteriliyor. Soğutucu cihaz açık durumdan kapalı duruma getiriliyor.

```
if (sogutucuDurumu == true) {
    agArayuzuIki.goruntule( mesaj: "Soğutucu Başarılı Bir Şekilde Kapatıldı. \n \n");

    sogutucuDurumu = false;
```

Eğer istek yapıldığı zaman soğutucu cihazımız zaten kapalı durumda ise gene ağ arayüzü yardımı ile kullanıcımıza soğutucunun kapalı durumda olduğu bildiriliyor.

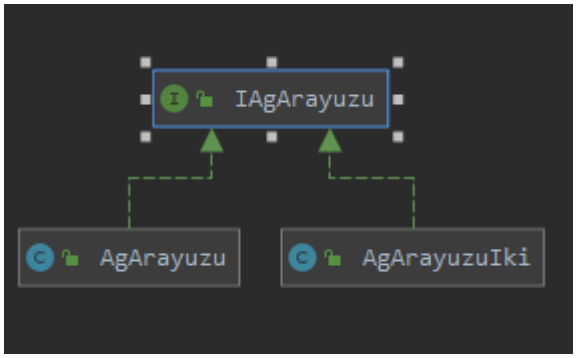
```
else {
  agArayuzuIki.goruntule( mesaj: "Soğutucu Zaten Kapalı Konumda. \n \n");
```

I. “Open/Closed” ilkesinin ne olduğu ve uygulama içerisinde nasıl gerçeklendiği.

Open/Closed İlkesi

Open/Closed ilkesi nesne yönelimli programlamada yazılım tasarımını daha esnek, anlaşır ve kolay hale getirmeyi amaçlayan beş tasarım ilkesinden birisidir. Temel olarak değişikliğe kapalı ancak yeni davranışların eklenmesine açık kod yazmayı hedeflemektedir.

Open/Closed ilkesini koduma uygularken iki adet menü yapmayı düşündüm. Bunun için kullanıcıya menüyü ve mesajları gösteren ve aynı zamanda kullanıcı isteklerini alıp buna uygun işlemler yapan ağ arayüzünden iki adet tasarlayıp bunları bir interface ile kodladım.

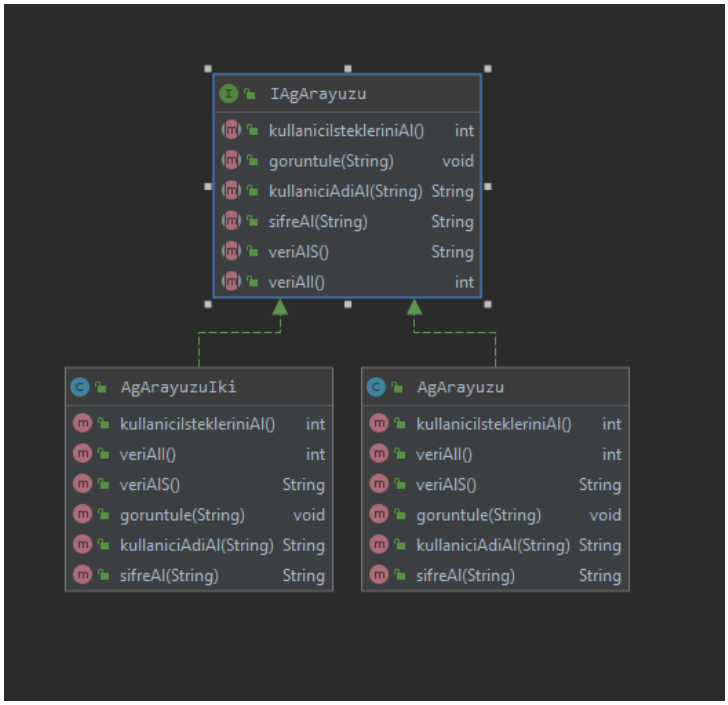


Kullandığım bu interface ve sınıflardan nesneler oluşturdum.

```
IAgArayuzu agArayuzu;
IAgArayuzu agArayuzuIki;
```

```
agArayuzu = new AgArayuzu();
agArayuzuIki=new AgArayuzuIki();
```

Aynı interfaceye bağlı oldukları için içerilerinde bulunan metotları da büyük ölçüde aynı kodladım.



İki sınıf arasındaki asıl fark ise yapmak istediğim 2 farklı metot oldu burada bir metodu tamamen dikey diğerini ise yatay çıktı verecek şekilde kodladım.

```
public class AgArayuzuIki implements IAgArayuzu {


    @Override
    public int kullaniciIstekleriniAl() {


        goruntule( mesaj: "Ana Menu \n");
        goruntule( mesaj: "1-Sıcaklık Değerini Hesapla      ");
        goruntule( mesaj: "2-Soğutucuyu Aç \n");
        goruntule( mesaj: "3-Soğutucuyu Kapat               ");
        goruntule( mesaj: "4-Durumu Değerlendir \n");
        goruntule( mesaj: "5-Çıkış \n");
        goruntule( mesaj: "Seçim :");
        return veriAlI();
    }
```

```
public class AgArayuzu implements IAgArayuzu{

    @Override
    public int kullaniciIstekleriniAl() {

        int secim;

        Scanner kullaniciSecimi = new Scanner(System.in);

        goruntule( mesaj: "Ana Menu \n");
        goruntule( mesaj: "1-Sıcaklık Değerini Hesapla \n");
        goruntule( mesaj: "2-Soğutucuyu Aç \n");
        goruntule( mesaj: "3-Soğutucuyu Kapat \n");
        goruntule( mesaj: "4-Durumu Değerlendir \n");
        goruntule( mesaj: "5-Çıkış \n");
        goruntule( mesaj: "Seçim :");
        secim=kullaniciSecimi.nextInt();


        return secim;
    }
```

```
Ana Menu
1-Sıcaklık Değerini Hesapla     2-Soğutucuyu Aç
3-Soğutucuyu Kapat              4-Durumu Değerlendir
5-Çıkış
Seçim :
```

```
Ana Menu
1-Sıcaklık Değerini Hesapla
2-Soğutucuyu Aç
3-Soğutucuyu Kapat
4-Durumu Değerlendir
5-Çıkış
Seçim :
```

İstediğim menüyü kullanmak için sadece merkezi işlem biriminde kullaniciIstekleriniAl fonksiyonun çağırır iken oluşturduğum nesneyi değiştirdim.

```
kullaniciSecimi = agArayuzuIki.kullaniciIstekleriniAl();
kullaniciSecimi=agArayuzu.kullaniciIstekleriniAl();
```

J. “Singleton” ve “Observer” desenlerinin ne olduğu ve uygulama içerisinde nasıl gerçeklendiği.

J1.Singleton Deseni

Singleton ilkesi temel olarak sahip olduğumuz sınıftan tek bir nesne oluşturma ilkesidir. Yazdığımız kod sayesinde kullanıcı eğer sınıftan bir nesne oluşturulmamış ise nesne oluşturabilir, nesne zaten oluşturulmuş ise bu nesneyi tekrar tekrar kullanmaktadır.

Ben kendime koduma bu ilkeyi eyleyici sınıfında ekledim.

Öncelikle private static bir eyleyici değişkeni olan instance tanımladım.

```
private static Eyleyici instance;
```

Daha sonra normalin aksine bir private kurucu tanımladım. Kurucu içinde instance değişkenine null değeri atadım.

```
private Eyleyici() {
    instance = null;
    minSicaklikDegeri = -14;
    sogutucuDurumu = false;
}
```

İstenenleri sağlamak için bir getInstance fonksiyonu yazdım. Kurucu fonksiyon private olduğu için bu fonksiyon eyleyici nesnesi oluşturulmak istendiği zaman çağrılıyor.

```
IEyleyici eyleyici;
eyleyici=Eyleyici.getInstance();
```

Fonksiyonun içerisinde ilk olarak instance değerinin null olup olmadığı yani daha önce oluşturup oluşturulmadığına bakılıyor. Daha önce nesne oluşturulmamış ise new komutu ile yeni nesne oluşturuluyor fakat daha önce nesne oluşturulmuş ise aynı nesne geri döndürülüyor.

```
public static Eyleyici getInstance() {
    if (instance == null) {
        instance = new Eyleyici();
    }
    return instance;
}
```

J2.Observer Deseni

Tasarım deseninin temel mantığı nesneler arasında birçok ilişkisi olmasıdır. Okulumuzda kullanılan sabis içerisindeki öğretim üyelerinin öğrencilere mesaj atması ya da marketlerde ürünlerde indirim

olduğu zaman sürekli alıcıların hepsine birden mesaj atılması gibi düşünülebilir. Yapılan bir değişiklik istenilen yerlerde istenilen etkiyi uyandırıyor.

Ödevimde observer desenini soğutucunun açılması üzerine kurdum. Soğutucu cihaz açıldığı zaman sıcaklık senaryoma göre istenilen derece olan –18 dereceye geliyor. Sıcaklık değeri istenen dereceye geldiği zaman ise o anki tarih bilgileri ile veri tabanına kayıt ediliyor. Ayrıca merkezi işlem birimi üzerinde ağ arayüzü yardımı ile kullanıcıya değer gösteriliyor.

Öncelikle IObserver adında bir interface tanımladım.

```
public interface IObserver {

    public void update(int anlikSicaklik,IAgArayuzu agArayuzu) throws SQLException;
}
```

Daha sonra yapılacak değişikliklerden etkilenecek olan veri tabanı ve kullanıcı için iki adet sınıf açarak IObserver interfacesinden kalıtım aldım. IObserver içerisinde bulunan update fonksiyonu ile etkilenecek kullanıcılarda neler gerçekleşeceğini ayrı ayrı kodladım.

```
public class SubscriberMainMenu implements IObserver {

    @Override
    public void update(int anlikSicaklik,IAgArayuzu agArayuzu) throws SQLException {

        agArayuzu.goruntule( mesaj: "Ayarlanan sıcaklık değeri :"+anlikSicaklik+ "\n \n");
    }
}
```

```
public class SubscriberDataBase implements IObserver{


    @Override
    public void update(int anlikSicaklik,IAgArayuzu agArayuzu) throws SQLException {

        IVeriTabani veriTabani = new VeriTabani();
        Date simdikiTarih = Calendar.getInstance().getTime();
        DateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd hh:mm:ss");
        String simdikiTarihS = dateFormat.format(simdikiTarih);
        veriTabani.veriTabaninaKayitEt(anlikSicaklik,simdikiTarihS,agArayuzu);;


       agArayuzu.goruntule( mesaj: "Ayarlanan Sıcaklık Değeri Ve Tarih VeriTabanına Eklendi. \n \n \n");


    }
}
```

Daha sonra ISubject adında bir interface oluşturdum. Burada attach fonksiyonu ile yapılan değişiklikten haberi olmasını istediğimiz alıcıları ekleme, detach ile istenmeyen alıcıları çıkarma işlemi yapabilirken notfiy fonksiyonu ile bir döngü yardımıyla oluşturduğumuz tüm alıcılara mesaj yollama işlemini gerçekleştirdim.

```
public interface ISubject {

    public void attach(IObserver newSubscriber);
    public void detach(IObserver newSubscriber);
    public void notify(int anlikSicalik,IAgArayuzu agArayuzu) throws SQLException;

}
```

Oluşturduğum bu interfaceyi gerçeklemek için Publisher adında bir sınıf açıp buradan kalıtım aldım. Publisher sınıfında ekstra olarak IObserver türünde bir arraylist tanımlayarak alıcılarımı burada kayıt ettim. Daha sonra notify fonksiyonu ile bu alıcılara sıcaklık değerini gönderdim. Ekrana bilgi mesajı yazdırdığım için bu işlevi yapan AgArayuzu sınıfından da bir nesne yolladım.

```
public class Publisher implements ISubject{


    private List<IObserver> subscribers = new ArrayList<>();

    @Override
    public void attach(IObserver newSubscriber)
    {
        subscribers.add(newSubscriber);
    }

    @Override
    public void detach(IObserver newSubscriber)
    {
        subscribers.remove(newSubscriber);
    }

    @Override
    public void notify(int anlikSicaklik,IAgArayuzu agArayuzu) throws SQLException {
        for(IObserver subscriber: subscribers) {
            subscriber.update(anlikSicaklik,agArayuzu);
        }
    }
}
```

Gönderme işlemi benim senaryoma uygun olarak soğutucu aç fonksiyonu içinde soğutucu değeri hesaplandığı zaman gönderildi.

```
public int sogutucuyuAc(int guncelSicaklikDegeri, Publisher p, IAgArayuzu agArayuzuI

    if (sogutucuDurumu == false) {
        agArayuzuIki.goruntule( mesaj: "Soğutucu Cihaz Açılıyor. \n");
        agArayuzuIki.goruntule( mesaj: "Sıcaklık Değeri Ayarlanıyor... \n \n");
        while (guncelSicaklikDegeri >= minSicaklikDegeri - 3) {
            guncelSicaklikDegeri--;
        }
        agArayuzuIki.goruntule( mesaj: "Sıcaklık Değeri Ayarlandı. \n");

        p.notify(guncelSicaklikDegeri, agArayuzuIki);
        sogutucuDurumu = true;
```