SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

NESNE YÖNELİMLİ ANALİZ VE TASARIM DERSİ

AKILLI CİHAZ TASARIMI PROJESİ

AD SOYAD: BARIŞ YILMAZ

ÖĞRENCİ NO: G191210303

ÖĞRENİM TÜRÜ/ŞUBE: 2/A

E-POSTA: baris.yilmaz5@ogr.sakarya.edu.tr

Mayıs, 2021

Kullanıcı doğrulama ekranı ve açıklaması

```
Hoşgeldiniz, cihaz başlatılıyor...
Kullanıcı adı ve şifre giriniz.
Kullanıcı adı:
baris
Sifre:
12345
Sogutucu veritabanina bağlaniliyor...
Veritabanına bağlandı!
Kullanıcı sorgulanıyor...
Kullanıcı doğrulanamadı! Lütfen tekrar deneyiniz.
Kullanıcı adı ve şifre giriniz.
Kullanıcı adı:
```

Kullanıcı bilgilerini girdiğinde veritabanından kullanıcı bilgileri kontrol ediliyor ve bilgiler eşleşmiyorsa kullanıcı bilgileri yeniden isteniyor. Kullanıcı 3 defa yanlış bilgi girerse programdan çıkış yapılıyor.

```
3 defa başarısız giriş yaptığınızdan dolayı sistemden çıkış yapılıyor...

Process finished with exit code 0
```

```
Hoşgeldiniz, cihaz başlatılıyor...
Kullanıcı adı ve şifre giriniz.
Kullanıcı adı:
baris
Sifre:
baris
Sogutucu veritabanina bağlaniliyor...
Veritabanına bağlandı!
Kullanıcı sorgulanıyor...
Kullanıcı doğrulandı! Ana menüye yönlendiriliyor...
*********************************************
Ana Menu
1-Soğutucuyu Aç
2-Soğutucuyu Kapa
3-Sıcaklığı Görüntüle
4-Cikis
Seciminiz:
*********************************************
```

Kullanıcı bilgilerinin veritabanındaki kullanıcı bilgileri ile eşleşmesi durumunda giriş yapılıyor ve işlem menüsü görüntüleniyor.

Sıcaklığın görüntülenmesi ve soğutucunun açılıp kapatılması

```
Ana Menu
1-Soğutucuyu Aç
2-Soğutucuyu Kapa
3-Sıcaklığı Görüntüle
4-Cikis
Seciminiz:
**********************************************
3
Sıcaklık bilgisi alınıyor...
Sicaklik: 11.0°C
**********************************************
Ana Menu
1-Soğutucuyu Aç
2-Soğutucuyu Kapa
3-Sıcaklığı Görüntüle
4-Cikis
Seciminiz:
**********************************************
```

Ana menüden sıcaklık görüntüleme işlemi seçildiğinde random olarak soğutucu kapalı ise 0 ile 60 derece, soğutucu açık ise 0 ile -20 derece arasında sıcaklık görüntüleniyor.

```
Ana Menu
1-Soğutucuyu Aç
2-Soğutucuyu Kapa
3-Sıcaklığı Görüntüle
4-Cikis
Seciminiz:
**********************************************
1
Soğutucu açılıyor...
Soğutucu açıldı!
**********************************************
Ana Menu
1-Soğutucuyu Aç
2-Soğutucuyu Kapa
3-Sıcaklığı Görüntüle
4-Cikis
Seciminiz:
**********************************************
```

Ana menüden soğutucunun açılması işlemi seçildiğinde ana işlem platformundan eyleyiciye soğutucuyu açması talebi gönderiliyor ve soğutucu açılıyor. Soğutucu zaten açık ise işlem yapılmıyor ve zaten açık bilgisi veriliyor.

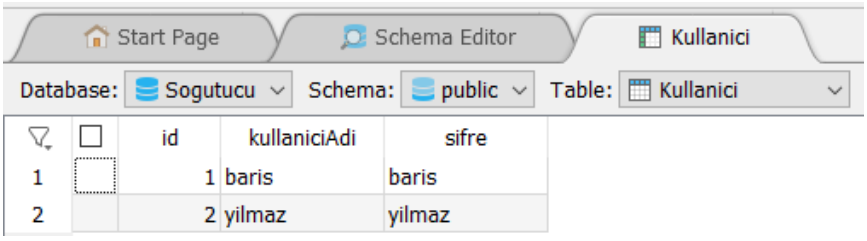
```
Soğutucu açılıyor...
Soğutucu zaten açık!
```

```
Ana Menu
1-Soğutucuyu Aç
2-Soğutucuyu Kapa
3-Sıcaklığı Görüntüle
4-Cikis
Seciminiz:
**********************************************
2
Soğutucu kapatılıyor...
Soğutucu kapatıldı!
**********************************************
Ana Menu
1-Soğutucuyu Aç
2-Soğutucuyu Kapa
3-Sıcaklığı Görüntüle
4-Cikis
Seciminiz:
**********************************************
```

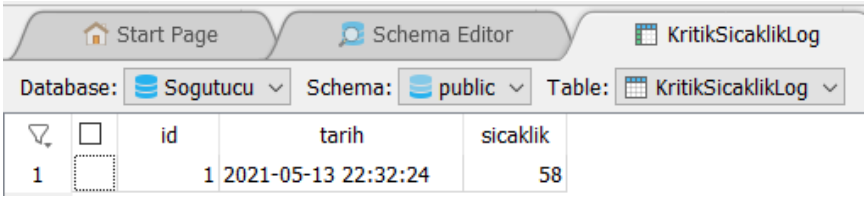
Ana menüden soğutucunun kapatılması işlemi seçildiğinde ana işlem platformundan eyleyiciye soğutucuyu kapatması talebi gönderiliyor ve soğutucu kapatılıyor. Soğutucu zaten kapalı ise işlem yapılmıyor ve zaten kapalı bilgisi veriliyor.

```
Soğutucu kapatılıyor...
Soğutucu zaten kapalı!
```

Veritabanının görüntüsü



Kullanıcılar Kullanici tablosunda veritabanında tutuluyor.

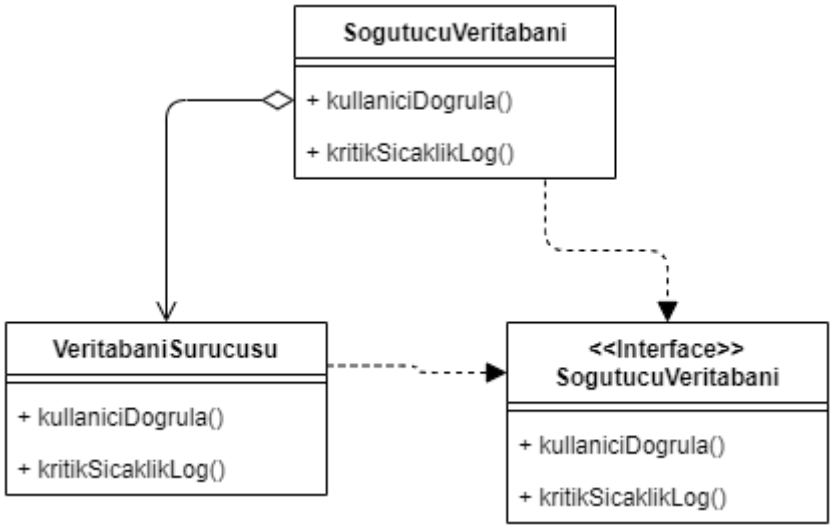


Sıcaklık, kritik eşiği (40 derece) geçtiği zaman KritikSicaklikLog tablosuna loglanıyor.

Dependency Inversion İlkesi

Bu prensip bize somutlara değil, soyutlamalara bağlı kalınması gerektiğini söylüyor. Yani bir modül başka bir modülü kullanacağı zaman o modülün gerçeklemesine değil soyutuna bağlı olmalıdır, onu kullanmalıdır. Bu sayede modüllerin birbirlerine bağımlılıklarının azaltılması sonrasında modülde yapılacak değişiklikler diğer modülleri etkilememiş olur ve sonradan değişiklik yapılabilmesi, yeni özellik eklenebilmesi kolaylaşmış olur. Bu soyutlama işlemini interfaceler veya abstract sınıflarla sağlarız.

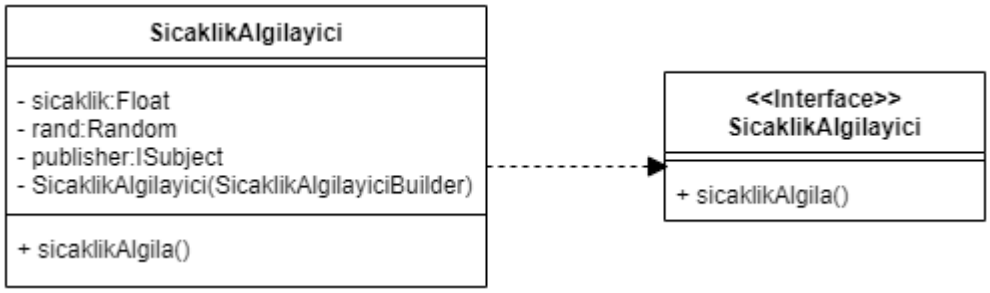
Projemde bu prensibi Veritabanı, Sıcaklık Algılayıcı, Eyleyici ve Ana İşlem Merkezi modüllerini gerçeklemek için kullandım.



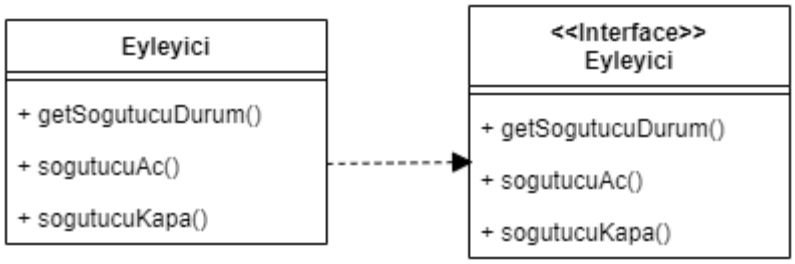

```
ISogutucuVeritabani sogutucuVeritabani=new SogutucuVeritabani(new PostgreSQLSurucu());
```

SogutucuVeritabani arayüzünden SogutucuVeritabani nesnesine parametre olarak projemde VeritabaniSurucusu yani PostgreSQL'i gönderiyorum veritabanım PostgreSQL için çalışıyor. Sonrasında başka bir veritabanı sürücüsü entegre etmek istersem yalnızca parametre olarak o sürücüyü vereceğim ve bu prensip sayesinde bunu çok ufak bir değişiklikle yapabilmiş olacağım.

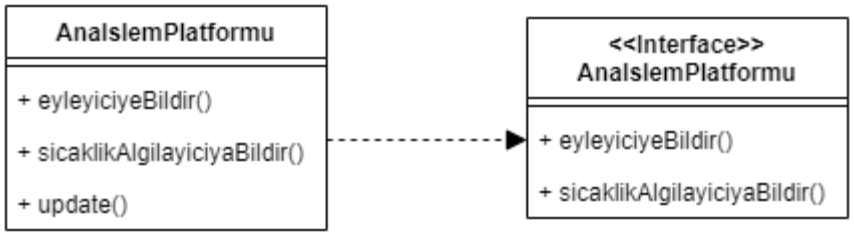
Aynı mantık ile bu prensip sayesinde Eyleyici, Sıcaklık Algılayıcı ve Ana İşlem Merkez'lerini daha sonra farklı markaların aynı işi yapan bu cihazlarıyla ufak bir değişiklikle değiştirebilir durumdayım.




```
ISicaklikAlgilayici sicaklikAlgilayici=new SicaklikAlgilayici.SicaklikAlgilayiciBuilder().publisher(publisher).build();
```




```
IEyleyici eyleyici=new Eyleyici();
```

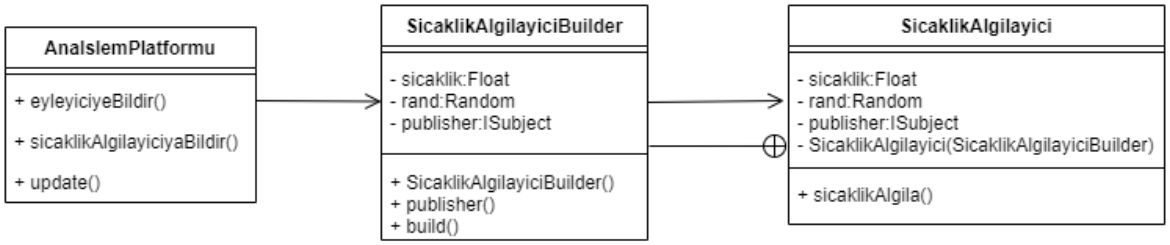



```
IAnaIslemPlatformu anaIslemPlatformu=new AnaIslemPlatformu(publisher);
```

Builder Deseni

Bu desen karmaşık nesnelerin yani içerisinde çok sayıda üye değişken ve üye nesne olan, farklı temsillerinin her biri için ayrı ayrı kurucuların oluşturulması ihtiyacı olan sınıflar için kullanılır. Bu desen sayesinde nesne oluşturma işini nesnenin kendisinden ayırmış oluruz.

Projemde bu desene uygun karmaşık bir sınıfım olmasa da kullanılması gerektiğinden dolayı observer deseni için kullandığım istemci olan sıcaklık algılayıcıda kullandım. Publisher nesnesi verilirse istemci olarak kullanılıyor, verilmez ise kullanılmıyor. Normalde bu işi 1 veya 2 parametreli kurucu ile yapacakken builder deseni sayesinde bu yükten kurtuluyoruz.

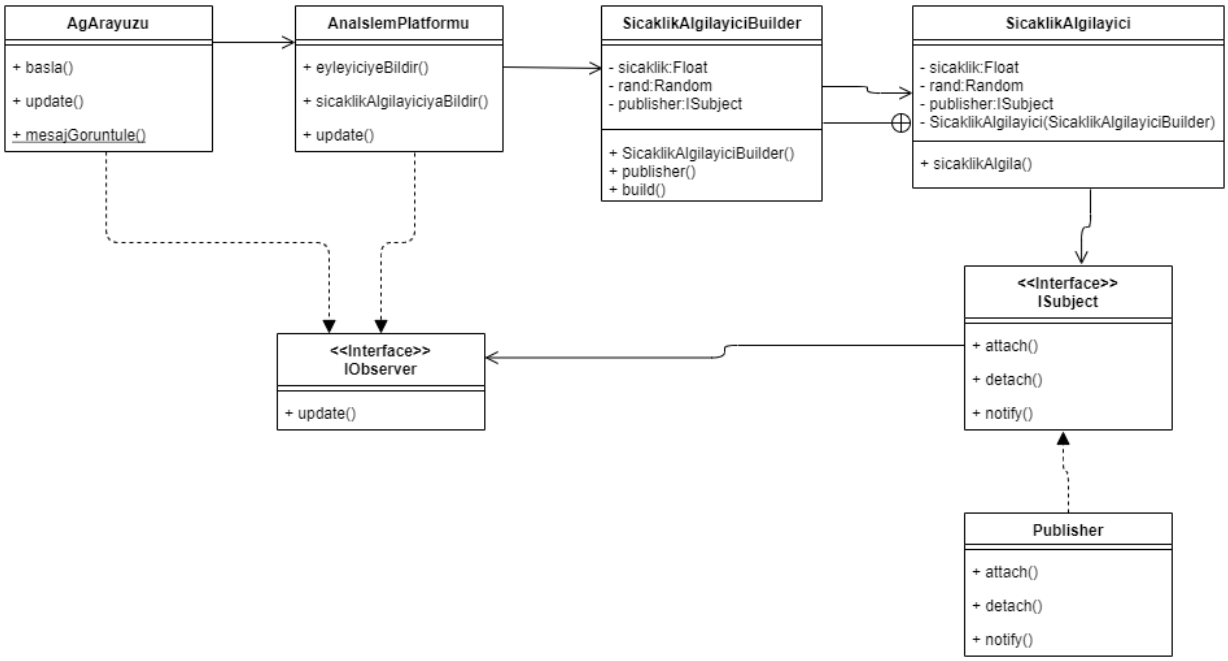



```java
public class SicaklikAlgilayici implements ISicaklikAlgilayici{
    private float sicaklik;
    private Random rand;
    private ISubject publisher;

    private SicaklikAlgilayici(SicaklikAlgilayiciBuilder builder)
    {
        this.rand = builder.rand;
        this.publisher=builder.publisher;
        this.sicaklik= builder.sicaklik;
    }
    private float randomSicaklik(boolean sogutucuDurum){
        if(sogutucuDurum)   return rand.nextInt( bound: 0 - (-20) + 1)+(-20);
        else    return rand.nextInt( bound: 61);
    }
    public void sicaklikAlgila(boolean sogutucuDurum){
        AgArayuzu.mesajGoruntule("Sicaklık bilgisi alınıyor...");
        Araclar.bekle( sure: 1000);
        sicaklik=randomSicaklik(sogutucuDurum);
        AgArayuzu.mesajGoruntule("Sicaklik: "+sicaklik+"°C");
        if(sicaklik>=40 && publisher!=null){
            publisher.notify( mesaj: "Sicaklik: "+sicaklik+"°C kritik esigi gecti!", sicaklik);
        }
    }

    public static class SicaklikAlgilayiciBuilder
    {
        private float sicaklik;
        private Random rand;
        private ISubject publisher;

        public SicaklikAlgilayiciBuilder() {
            rand = new Random(System.currentTimeMillis());
            sicaklik=0;
        }

        public SicaklikAlgilayiciBuilder publisher(ISubject publisher) {
            this.publisher=publisher;
            return this;
        }
        public SicaklikAlgilayici build() { return new SicaklikAlgilayici( builder: this); }
```

Observer Deseni

Bu desenin amacı çok sayıda nesneye, gözlemledikleri nesnede meydana gelen olayı bildirmektir. Yayıncı-abone mantığı ile çalışır. Yayıncıda gerçekleşen bir değişiklik bilgisi abonelere bildirir.

Projemde bu deseni sıcaklığın kritik eşiğin üzerine (40 derece üzeri) çıktığı durum için yayıncının Sıcaklık Algılayıcı, abonelerin ise soğutucuya açma isteği gönderecek Ana İşlem Merkezi ve veritabanındaki KritikSicaklikLog tablosuna sıcaklığı, tarihi loglayacak Ağ Arayüzü olarak tasarladım.




```java
public interface IObserver {
    public void update(String mesaj, float sicaklik);
}
```

```java
public interface ISubject {
    void attach(IObserver o);
    void detach(IObserver o);
    void notify(String mesaj, float sicaklik);
}
```

```java
public class Publisher implements ISubject {
        private List<IObserver> subscribers = new ArrayList<>();
        @Override
        public void attach(IObserver subscriber) { subscribers.add(subscriber); }
        @Override
        public void detach(IObserver subscriber) { subscribers.remove(subscriber); }
        @Override
        public void notify(String mesaj, float sicaklik) {
            for(IObserver subscriber: subscribers) {
                subscriber.update(mesaj, sicaklik);
            }
        }
}
```

```java
if(sicaklik>=40 && publisher!=null){
    publisher.notify( mesaj: "Sicaklik: "+sicaklik+"°C kritik esigi gecti!", sicaklik);
}
```

```java
publisher=new Publisher();
publisher.attach( o: this);
```

Ağ Arayüzü'nde oluşturulan Publisher nesnesini Ana İşlem Merkezi'nin kurucusuna parametre olarak göndererek onu da attach ediyorum.

Sıcaklığın kritik eşiği geçtiği durum:

```
Sıcaklık bilgisi alınıyor...
Sicaklik: 51.0°C
AgArayuzune gelen mesaj-> Sicaklik: 51.0°C kritik esigi gecti!
Sogutucu veritabanina bağlaniliyor...
Veritabanına bağlandı!
Kritik sicaklik bilgisi loglaniyor...
AnaIslemPlatformuna gelen mesaj-> Sicaklik: 51.0°C kritik esigi gecti!
Soğutucu açılıyor...
Soğutucu açıldı!
```

Uygulamanın Kaynak Kodları

https://github.com/xbarisyilmaz/NesneYonelimliAnalizveTasarim

Çalışmanın anlatıldığı videonun adresi

https://youtu.be/ZFyxa8FHDmc