

# Nesne Yönelimli Analiz ve Tasarım Projesi Çalışması

Akıllı Cihaz Tasarımı

Kadriye Oral

G191210090—2B—2.Ö kadriye.oral@ogr.sakarya.edu.tr

## VERİTABANINA BAĞLANMA

```
*****Veritabanına Bağlanıldı*****  
Kullanıcı adınızı giriniz: Kader  
Şifrenizi giriniz: 12345  
*****Giriş Başarılı!*****
```

- Veritabanımdaki kullanıcı adı ve şifre yukarıda gösterildiği gibidir. Dolayısıyla girişimiz başarıyla gerçekleşmiştir.

## SICAKLIĞIN GÖRÜNTÜLENMESİ

```
*****İşlem Seçiniz*****  
1-Sıcaklık Görüntüle  
2-Soğutucu Aç  
3-Soğutucu Kapat  
4-Akıllı Cihaz Hakkında  
5-Çıkış Yap  
*****  
1  
Ortam Sıcaklığı:10°C
```

```
*****İşlem Seçiniz*****  
1-Sıcaklık Görüntüle  
2-Soğutucu Aç  
3-Soğutucu Kapat  
4-Akıllı Cihaz Hakkında  
5-Çıkış Yap  
*****  
1  
Ortam Sıcaklığı:16°C
```

- Ortam sıcaklığı rastgele olarak üretiliyor...




## SOĞUTUCUNUN AÇILMASI

```
*****İşlem Seçiniz*****
1-Sıcaklık Görüntüle
2-Soğutucu Aç
3-Soğutucu Kapat
4-Akıllı Cihaz Hakkında
5-Çıkış Yap
*****
2
Soğutucu Açıldı!
```

## SOĞUTUCUNUN KAPATILMASI

```
*****İşlem Seçiniz*****
1-Sıcaklık Görüntüle
2-Soğutucu Aç
3-Soğutucu Kapat
4-Akıllı Cihaz Hakkında
5-Çıkış Yap
*****
3
Soğutucu kapatıldı!
```

## VERİTABANI

Data Output	Explain	Messages	Notifications
	Adı character varying 	Şifre [PK] character varying 	
1	Kader	12345	

## DEPENDENCY INVERSION

- Bağımlılıkları tersine çevirmektir.
- Üst seviye modüllerin, alt seviye modüllerine olan bağımlılıklarını tersine çevirip yeniden kullanılabilirlik, esneklik gibi önemli özellikleri çözüme katmamıza olanak sağlayan bir SOLID tasarım prensibidir.

```
public interface IDip {  
    void dereceArtır();  
    void dereceAzalt();  
}
```

- IDip diye bir arayüzümüz var ve dereceArtır() ve dereceAzalt() metotlarını içeriyor.

```
public class SicaklikKontrol {  
    IDip _dip;  
    public SicaklikKontrol(IDip dip){  
        this._dip=dip;  
    }  
  
    public SicaklikKontrol() {  
        _dip.dereceArtır();  
        _dip.dereceAzalt();  
    }  
}
```

- Burda SicaklikKontrol sınıfımızda interface ten bir field oluşturuyoruz.
- SicaklikKontrol default constructor ında IDip interfacesi parametre olarak veriyoruz.

```

@Override
public void dereceArtir() {
    System.out.println("1-Soğutucunun Derecesini arttır");
}

@Override
public void dereceAzalt() {
    System.out.println("2-Soğutucunun Derecesini azalt");
}

```

- SicaklikAlgilyicisi classında Idip interfaceini implements ettik.

```

public void sicaklikDegistir()
{
    IDip DİP =new SicaklikAlgilyicisi();
    SicaklikAlgilyicisi algilyicisi=new SicaklikAlgilyicisi();
    while(true) {
        algilyicisi.dereceArtir();
        algilyicisi.dereceAzalt();

        secim=islemSec.nextInt();
        if (secim == 1) {

            sicaklik = sicaklik + (int) (Math.random() * ((50 - 30) + 20));
            System.out.println("Arttırıldı!\nYeni Derece:" + sicaklik+"°C");
            continue;
        }
        if (secim == 2) {
            sicaklik = sicaklik - (int) (Math.random() * ((50 - 24) + 1));
            System.out.println("Azaltıldı!\nYeni Derece:" + sicaklik+"°C");
        }
    }
    break;
}

```

## BUİLER DESENİ

- Yapıcıya giden parametreler azaltılır ve daha kolay okunabilir yöntem sağlanmış olur.
- Constructordaki bütün parametreleri almak istemiyorsak bu işimizi kolaylaştırır.
- Inner class kullanılır ve burada hangi parametreyi istiyorsak onu alırız.

Yaptığım örnek üzerinden anlatmaya çalışacağım;

```
public class AkilliCihaz {
    private String Name;
    private String Surum;
    private String Dil;
    private String Id;
    public static class Builder {
        private String name ;
        private String surum ;
        private String dil;
        private String id;
        public Builder Name(String name) {...}
        public Builder Surum(String surum) {...}
        public Builder Dil(String dil) {...}
        public Builder Id(String id){...}
        public AkilliCihaz build() {return new AkilliCihaz(builder: this);}
    }
    public AkilliCihaz(Builder builder) {
        Name = builder.name;
        Surum = builder.surum;
        Dil= builder.dil;
    }
    public String toString(){return this.Name + "," + this.Surum + "," + this.Dil/*+ ","+this.Id*/;}
}
```

- Burada akıllı cihazıma ait dört özellik bulunmakta fakat ben sadece üçünün kullanıcıya gösterilmesini istedim. Bunu normal bildiğimiz yapıcı metotla yapmış olsaydık dört parametreyi de girmem gerekirdi. Ve son olarak nesneler oluşturulup bilgiler girildi.

```
AkilliCihaz Adi = new AkilliCihaz.Builder()
    .Name("HerEveLazım")
    .build();
AkilliCihaz Surum = new AkilliCihaz.Builder()
    .Surum("1.0.3")
    .build();

AkilliCihaz Dil = new AkilliCihaz.Builder()
    .Dil("English")
    .build();
```

## OBSERVER DESENİ

- Observer (gözlemci) tasarım deseni, içerisinde bir nesne ve o nesnenin bağımlılıklarını güncelleyen ve tüm durum değişikliklerini o nesneye haber veren bir gözlemci bulunduran bir yazılım tasarım desenidir. Gözlemci deseninde bir nesnenin durumu değiştiğinde diğer nesnelere haber verilir ve haberi alan nesneler kendilerini günceller.

```
public interface IObservable {  
    void sogutucuAcik(IObserver observer);  
    void sogutucuKapali(IObserver observer);  
}  
  
public interface IObserver {  
    void update();  
}
```

- Observer arayüzü yukarıda gözüken metotları Observer arayüzünü parametre olarak kullanıyor.update metodu akıllı cihazın güncel olup olmadığını haber veriyor.

```
public class Arayuz implements IObservable {  
    private boolean sogutucuDurum;  
    Eyleyici eyle = new Eyleyici();  
    SicaklikAlgilayicisi sicaklikAlgila = new SicaklikAlgilayicisi();  
    @Override  
    public void sogutucuAcik(IObserver observer) {  
        if (!this.sogutucuDurum) {  
            System.out.println("Soğutucu Açıldı!");  
        }  
    }  
  
    @Override  
    public void sogutucuKapali(IObserver observer) {  
        if (!this.sogutucuDurum )  
        {  
            System.out.println("Soğutucu kapatıldı!");  
        }  
    }  
}
```

- Burada da eyleyicideki soğutucunun durumuna göre ekrana yapılan işlem gösteriliyor.

```
boolean cihazGuncelMi=true;
@Override
public void update() {

    if(!cihazGuncelMi){
        System.out.println("Akıllı cihaz için güncelleme mevcut.....");
        System.out.println("Akıllı cihaz güncelleniyor.....");
        System.out.println("Güncellemeler başarıyla tamamlandı.....");

    }else{
        System.out.println(" Akıllı cihaz güncel");
    }

}
```

**Github linki :** <https://github.com/tutunamayanlar2021/NYAT/tree/main/NYATProjesi/src>

**Youtube linki :** <https://www.youtube.com/watch?v=8c11G4XC00U>