



yusuf.aydogmus@ogr.sakarya.edu.tr

LİNKLER:

Github : https://github.com/Yusuf-Aydogmus/NYA_SogutucuCihaz

Youtube: <https://youtu.be/mEk4hNLvPqQ>

İÇİNDEKİLER

- a.** Kullanıcı doğrulama ekranı ve açıklaması.
- b.** Sıcaklığın görüntülenmesi ve soğutucunun açılıp kapatılmasıyla ilgili ekran görüntüleri ve açıklaması.
 - 1) Sıcaklığın Görüntülenmesi
 - 2) Soğutucunun Açılması
 - 3) Soğutucunun Kapatılması
- c.** Veritabanınızın görüntüsü (kullanıcı verilerinin saklandığı tablonun, verileri içeren görüntüsü).
- d.** “Dependency Inversion” ilkesinin ne olduğu ve uygulama içerisinde nasıl gerçekleştirildiği.
- e.** “Builder” ve “Observer” desenlerinin ne olduğu ve uygulama içerisinde nasıl gerçekleştirildiği
 - 1) Builder Tasarım Deseni
 - 2) Observer Tasarım Deseni
- f.** Uygulamanın kaynak kodları
- g.** Çalışmanın anlatıldığı videonun adresi.

a) Kullanıcı Doğrulama ekranı

```
main x
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\
Kullanici Adi : kullanici1
Sifre : 1

*****
Hosgeldiniz,Sayın: kullanici1
*****
```

Kullanıcı'dan kullanıcı adı ve şifre bilgileri istenir.Kullanıcı Bilgilerini doğru girmişse."Karşılama Ekranı" gösterilir.

```
main x
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" "-javaagent:C:\Program Files\Je
Kullanici Adi : kullanici2
Sifre : 3

*****
Kullanici Bilgileri Dogrulanamadi.Lutfen Tekrar Deneyiniz
*****

Kullanici Adi :
```

Kullanıcının girdiği bilgiler veritabanında mevcut değilse. "Hata Ekranı" gösterilerek yineleme yapılır

```
public boolean KullaniciDogrula(Kullanici kullanici) {
    boolean sonuc = false;
    String tabloAdi=" \\KullaniciGiris\\" ";
    String username=" \\kullaniciAdi\\" ";
    String password=" \\sifre\\" ";
    try {
        Connection baglanti= this.Baglan();
        String sql="SELECT * FROM"+tabloAdi+"WHERE"+username+"='"+kullanici.getKullaniciAdi()+"'AND"+password+"='"+kullanici.getSifre()+"' ";
        Statement stmt=baglanti.createStatement();
        ResultSet rs=stmt.executeQuery(sql);
        baglanti.close();
        if(!rs.next()){
            System.out.println("");
            System.out.println("*****");
            System.out.println( ConsoleColors.RED_BACKGROUND+" Kullanici Bilgileri Dogrulanamadi.Lutfen Tekrar Deneyiniz"+ConsoleColors.RESET);
            System.out.println("*****");
            System.out.println("");
            sonuc=false;
        }
        else{
            sonuc=true;
            System.out.println("");
            System.out.println("*****");
            System.out.println(" "+ConsoleColors.YELLOW_BACKGROUND+"Hosgeldiniz,Sayın: "+kullanici.getKullaniciAdi()+ConsoleColors.RESET);
            System.out.println("*****");
            System.out.println("");
        }
    }
    rs.close();
}
```

b) Sıcaklığın Görüntülenmesi ve Soğutucu İşlemleri

1) Sıcaklığın Görüntülenmesi

```
*****
***      ANA MENU      ***
***  1-)Sicaklik Goruntule  ***
***  2-)Sogutucu Calistir  ***
***  3-)Sogutucu Kapat    ***
***  4-)Surum Kontrol     ***
***  5-)Cikis            ***
*****
SECİM: 1
~ İşlem Gerçekleştiriliyor ~
Sicaklik: 37 derece
```

Kullanıcı girişini tamamlayan kullanıcılar Ana menüye ulaşırlar.

Burada Sıcaklık Görüntülenmesi seçimi yapılır.

```
public void anaMenu() throws InterruptedException {
    menuTemizle();
    System.out.println("*****");
    System.out.println("***+ConsoleColors.BLUE_BOLD + "      ANA MENU      "+");
    System.out.println("***+ "      1-)Sicaklik Goruntule  "+");
    System.out.println("***+ "      2-)Sogutucu Calistir  "+");
    System.out.println("***+ "      3-)Sogutucu Kapat    "+");
    System.out.println("***+ "      4-)Surum Kontrol     "+");
    System.out.println("***+ "      5-)Cikis            "+");
    System.out.println("*****");
    secim=secimYap();

    if(secim==1){
        SicakliGoruntule();
    }
    else if(secim==2){
        SogutucuCalistir();
    }
    else if(secim==3){
        SogutucuKapat();
    }
    else if(secim==4){
        SurumKontrol();
    }
}

public void SicakliGoruntule(){
    System.out.print("Sicaklik: ");
    System.out.print(algilayici1.SicaklikGoruntule());
    System.out.println(" derece ");
}
```

Kullanıcının seçimi doğrultusunda menü sınıfının SicakliGörüntüle Metodu Çağrılır.

SicakliGoruntule metodu ise algilayici sınıfının SicaklikGoruntule metodunu kullanır.

```
import java.util.Random;

public class Algilayici implements IAlgilayici{
    static int Sicaklik;
    public Algilayici(){
        Random rastgele=new Random();
        this.Sicaklik=rastgele.nextInt( bound: 30)+10;
    }

    public int SicaklikGoruntule() {
        return Algilayici.Sicaklik;
    }
}
```

Algilayici sınıfından 10 ile 40 arasında rastgele bir sıcaklık değeri oluşturulması sağlanır

2) Soğutucunun Açılması

```
*****  
***      ANA MENU      ***  
***      1-)Sıcaklık Görüntüle  ***  
***      2-)Sogutucu Calistir  ***  
***      3-)Sogutucu Kapat  ***  
***      4-)Surum Kontrol  ***  
***      5-)Cikis  ***  
*****  
SECİM: 2  
~ İşlem Gerçekleştiriliyor ~  
Sıcaklık Degeri : 36  
Sıcaklık Degeri : 35  
Sıcaklık Degeri : 34  
Sıcaklık Degeri : 33  
Sıcaklık Degeri : 32
```

```
public void anaMenu() throws InterruptedException {  
    menuTemizle();  
    System.out.println("*****");  
    System.out.println("***"+ConsoleColors.BLUE_BOLD + "      ANA MENU      " + "***");  
    System.out.println("***"+ "      1-)Sıcaklık Görüntüle  " + "***");  
    System.out.println("***"+ "      2-)Sogutucu Calistir  " + "***");  
    System.out.println("***"+ "      3-)Sogutucu Kapat  " + "***");  
    System.out.println("***"+ "      4-)Surum Kontrol  " + "***");  
    System.out.println("***"+ "      5-)Cikis  " + "***");  
    System.out.println("*****");  
    secim=secimYap();  
  
    if(secim==1){...}  
    else if(secim==2){  
        SogutucuCalistir();  
    }  
}
```

```
public void SogutucuCalistir() throws InterruptedException {  
  
    agArayuzu.SogutucuAc();  
}
```

Kullanıcı Soğutucu Çalıştırı seçince menu sınıfının ilgili metodu çalıştırılır ve ag arayüzü üzerinden SogutucuAc metodu çağrılır

```
public class Eyleyici implements IEyleyici {  
    static boolean SogutucuDurumu=false;  
    // @Override  
    public static void SogutucuAc() {  
        if(SogutucuDurumu==false){  
            if(Algilyici.Sicaklik>15){  
                for(int i=0;i<5;i++){  
                    Araclar.bekle();  
                    Algilyici.Sicaklik--;  
                    System.out.println("Sıcaklık Degeri : "+Algilyici.Sicaklik);  
                }  
            }  
            else{  
                System.out.println("Hava yeterince Soğuk...");  
            }  
        }  
        else{  
            System.out.println(ConsoleColors.RED_BOLD+" Soğutucu ZATEN Açık "+ConsoleColors.RESET);  
        }  
        SogutucuDurumu=true;  
    }  
}
```

```
@Override  
public void SogutucuAc( ) throws InterruptedException {  
    // Cihaz cihaz=new Cihaz();  
    eyleyici.SogutucuAc();  
}
```

eyleyici sınıfının ilgili metodu çağrılır.Eyleyici ilk olarak soğutucunun açık olup olmasını kontrol eder ve buna göre işlem yapar.Sistemde soğutucu ortama +-5 derece müdahale yapmaktadır.

3) Soğutucunun Kapatılması

```
*****
```

```
***      ANA MENU      ***
```

```
***      1-)Sıcaklık Görüntüle      ***
```

```
***      2-)Sogutucu Calistir      ***
```

```
***      3-)Sogutucu Kapat      ***
```

```
***      4-)Surum Kontrol      ***
```

```
***      5-)Cikis      ***
```

```
*****
```

```
SEÇİM: 3
```

```
~ İşlem Gerçekleştiriliyor ~
```

```
Sıcaklık Degeri : 33
```

```
Sıcaklık Degeri : 34
```

```
Sıcaklık Degeri : 35
```

```
Sıcaklık Degeri : 36
```

```
Sıcaklık Degeri : 37
```

```
Devam Etmek İstiyormusunuz? e(E)/h(H)
```

```
public void anaMenu() throws InterruptedException {  
    menuTemizle();  
    System.out.println("*****");  
    System.out.println("***"+ConsoleColors.BLUE_BOLD+"      ANA MENU      "+C  
    System.out.println("***+"      1-)Sıcaklık Görüntüle      "+"***");  
    System.out.println("***+"      2-)Sogutucu Calistir      "+"***");  
    System.out.println("***+"      3-)Sogutucu Kapat      "+"***");  
    System.out.println("***+"      4-)Surum Kontrol      "+"***");  
    System.out.println("***+"      5-)Cikis      "+"***");  
    System.out.println("*****");  
    secim=secimYap();  
}
```

```
if(secim==1){...}  
else if(secim==2){...}  
else if(secim==3){  
    SogutucuKapat();  
}
```

```
public void SogutucuKapat() throws InterruptedException {  
    agArayuzu.SogutucuKapat();  
}
```

Kullanıcı Soğutucu kapatı seçince menu sınıfının ilgili metodu çalıştırılır ve ag arayüzü üzerinden Sogutucukapat metodu çağrılır

```
public static void SogutucuKapat() throws InterruptedException {  
    if(SogutucuDurumu==true){  
        for(int i=0;i<5;i++){  
            Thread.sleep( millis: 1000);  
            Algilayici.Sicaklik++;  
            System.out.println("Sıcaklık Degeri : "+Algilayici.Sicaklik);  
        }  
    }  
    else{  
        System.out.println(ConsoleColors.RED_BOLD+" Soğutucu ZATEN Kapalı  
    }  
    SogutucuDurumu=false;  
}
```

```
public void SogutucuKapat( ) throws InterruptedException {  
    // Cihaz cihaz=new Cihaz();  
    eyleyici.SogutucuKapat();  
}
```

Arayüz üzerinden eyleyici sınıfının ilgili metodu çağrılır.Eyleyici ilk olarak soğutucunun kapalı olup olmasını kontrol eder ve buna göre işlem yapar.Sistemde soğutucu ortama +5 derece müdahale yapmaktadır.

c) Veri Tabanı İşlemleri

The screenshot shows the Valentina Studio interface. The main window displays a table named 'KullaniciGiris' with the following data:

	kullaniciAdi	sifre	surum
1	kullanici3	3	0
2	kullanici1	1	1
3	kullanici2	2	2

Below the table, a 'Design Table 'KullaniciGiris' - Valentina Studio' window is open, showing the table's structure:

Table	Fields (3)	Constraints	Indexes (0)	Triggers	
	Name	Type	Nullable	Indexed	Unique
	kullanici	Character Varying	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	sifre	Character Varying	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	surum	Integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Sistemde kullanıcı doğrulaması yapılırken kullanılan SQL Komutu:

```
String sql="SELECT *  
FROM"+tabloAdi+"WHERE"+username+"='"+kullanici.getKullaniciAdi()+"'+  
AND"+password+"='"+kullanici.getSifre()+"'";
```

Sistemde surum kontrol yapılırken kullanılan SQL Komutu:

```
String sql = "SELECT" + surum + "From" + tabloAdi + "Where" + username +  
"=" + "'" + kullanici.getKullaniciAdi() + "'";
```

d) Dependency Inversion(Bağımlılıkların Tersine Çevrilmesi)

Bu prensibe göre somut sınıflara olan bağımlılıklar soyut sınıflar ve interface sınıflar kullanılarak ortadan kaldırılmalıdır,çünkü somut sınıflar sık sık değişikliğe uğrarlar ve bu sınıflara bağımlı olan sınıflarında yapısal değişikliğe uğramalarına sebep olur. Bu prensibin uygulanması somut sınıfların kullanımlarından doğan değişikliğin azaltılmasını sağlar.

```
IAlgilayici algilayici1;  
IAgArayuzu agArayuzu;  
  
public Menu(){  
    algilayici1=new Algilayici();  
    agArayuzu=new AgArayuzu();  
}
```

Örneğin uygulamamdaki Menu sınıfında oluşturduğum nesneler bulundukları sınıfın arayüzleri üzerinden yapılmıştır.Bu sayede bir üst sınıfa olan bağımlılık tersine çevrilmiştir

```
public void kullaniciDogrula() {  
    IVeriTabani veriTabani=new PostgreSQL();  
    Kullanici kullanici;  
    Scanner oku=new Scanner(System.in);
```

Gene kullanıcı doğrula metodum Interface üzerinden oluşturularak bu prensip mantığıyla oluşturulmuştur.

e) Builder-Observer Tasarım Desenleri

1-)Builder Tasarım Deseni(YAPICI)

Yapıcı (builder) tasarım şablonu da soyut fabrika tasarım şablonunda olduğu gibi istenilen bir tipte nesne oluşturmak için kullanılmaktadır. İki tasarım şablonu arasındaki fark, yapıcı tasarım şablonunun karmaşık yapıdaki bir nesneyi değişik parçaları bir araya getirerek oluşturmasında yatmaktadır. Birden fazla adım içeren nesne üretim sürecinde, değişik parçalar birleştirilir ve istenilen tipte nesne oluşturulur.

```
public static class Builder{
    private String kullanıcıAdi,sifre;
    int surum;

    public Builder(){
    }
    public Builder kullanıcıAdi(String kullanıcıAdi){
        this.kullanıcıAdi=kullanıcıAdi;
        return this;
    }
    public Builder sifre(String sifre){
        this.sifre=sifre;
        return this;
    }
    public Builder surum(int surum){
        this.surum=surum;
        return this;
    }
    public Kullanici build(){
        Kullanici kullanıcı=new Kullanici( kullanıcıBuilder: this);
        return kullanıcı;
    }
}

public class Kullanici {
    private String kullanıcıAdi;
    private String sifre;
    private int surum;

    public Kullanici(Builder kullanıcıBuilder){
        this.kullanıcıAdi=kullanıcıBuilder.kullanıcıAdi;
        this.sifre=kullanıcıBuilder.sifre;
        this.surum=kullanıcıBuilder.surum;
    }

    public String getSifre() { return sifre; }

    public String getKullanıcıAdi() { return kullanıcıAdi; }
}
```

Uygulamamda Kullanıcı sınıfında builder tasarım desenini uyguladım.Uygulamada doğrulama aşaması ve sürüm kontrol aşamaları kısmında işlerini yapmışlardır.

Doğrulama aşamasında kullanıcı adı ve şifre ile

Sürüm kontrol aşamasında kullanıcı adı ve sürüm ile nesne oluşturmamıza olanak sağlamıştır.

2-)Observer Tasarım Deseni(GÖZETLEYİCİ)

Sistem bünyesinde bir nesnede meydana gelen değişikliklerden haberdar olmak isteyen diğer nesneler olabilir.Bu durumda haberdar olmak isteyen nesneler abone olarak, abone oldukları nesnede meydana gelen değişikliklerden haberdar edilirler.Abone olan nesne aboneliğini iptal ederek abone olduğu nesne ile arasındaki ilişkiyi sonlandırabilir.

```
***      ANA MENU      ***
***      1-)Sıcaklık Görüntüle      ***
***      2-)Soğutucu Çalıştır      ***
***      3-)Soğutucu Kapat      ***
***      4-)Surum Kontrol      ***
***      5-)Çıkış      ***
*****
SEÇİM: 4
~ İşlem Gerçekleştiriliyor ~

Surumunuz:1
Yazılım Guncellemesi gelmiştir.Cihazınızı yeniden başlatın.
```


```
public void anaMenu() throws InterruptedException {
    menuTemizle();
    System.out.println("*****");
    System.out.println("***"+ConsoleColors.BLUE_BOLD+"
    System.out.println("***+"      1-)Sıcaklık Görüntüle
    System.out.println("***+"      2-)Soğutucu Çalıştır
    System.out.println("***+"      3-)Soğutucu Kapat
    System.out.println("***+"      4-)Surum Kontrol
    System.out.println("***+"      5-)Çıkış
    System.out.println("*****");
    secim=secimYap();

    if(secim==1){...}
    else if(secim==2){...}
    else if(secim==3){...}
    else if(secim==4){
        SurumKontrol();
    }
}
```

```
public void SurumKontrol(){
    System.out.println();
    System.out.println("Surumunuz:"+MerkeziIslemBirimi.surum);
    if(MerkeziIslemBirimi.surum==1){
        NoticeObservable n1=new NoticeObservable();
        EskiSurumKullanici surum1=new EskiSurumKullanici();
        n1.gozlemciEkle(surum1);
        n1.gozlemcilereHaberVer();
    }
    else{
        NoticeObservable n2=new NoticeObservable();
        YeniSurumKullanici surum2=new YeniSurumKullanici();
        n2.gozlemciEkle(surum2);
        n2.gozlemcilereHaberVer();
    }
}
```

Uygulamamda observer tasarım desenini Surum kontrol fonksiyonu ile uyguladım.Kullanıcıların cihazlarının surumunu gözetleyen ve yeni sürüm geldiğinde haberdar eden bir tasarım kurdum.

```
public interface IObserver {
    public void notify(String message);
}
```

```
public interface IObservable {
    
    void gozlemciEkle(IObserver observer);
    void gozlemciSil(IObserver observer);
    void gozlemcilereHaberVer();
}
```

```
import java.util.List;

public class NoticeObservable implements IObservable {
    private List<IObserver> observerList=new ArrayList<>();
    private String message="Yazılım Guncellemesi gelmiştir.Cihazınızı yeniden başlatın"

    @Override
    public void gozlemciEkle(IObserver observer) {
        observerList.add(observer);
    }

    @Override
    public void gozlemciSil(IObserver observer) { observerList.remove(observer); }

    @Override
    public void gozlemcilereHaberVer() {
        for(IObserver observer:observerList){
            observer.notify(message);
        }
    }
}
```

```
public class YeniSurumKullanicilar implements IObserver{
    @Override
    public void notify(String message) {
        System.out.println("Bir sonraki surume kadar en iyisi...");
    }
}
```

```
public class EskiSurumKullanicilar implements IObserver{
    @Override
    public void notify(String message) {
        System.out.println("Yazılım Guncellemesi gelmiştir.Cihazınızı yeniden başlatın.");
    }
}
```

f) Kaynak KOD:

Github:

https://github.com/Yusuf-Aydogmus/NYA_SogutucuCihaz

Video Linki:

<https://youtu.be/mEk4hNLvPqQ>