



## NESNE YÖNELİMLİ ANALİZ VE TASARIM

AD : RAİF

SOYAD : AKYOL

ÖĞRENCİ NO : G191210017

GRUP : 2.ÖĞRETİM A GRUBU

E-POSTA : raif.akyol@ogr.sakarya.edu.tr

NESNELERİN İNTERNETİ SİSTEMLERİ İÇİN AKILLI CİHAZ TASARIMI

## İÇİNDEKİLER

- KULLANICI DOĞRULAMA EKRANI
- SOĞUTUCUNUN AÇILMASI EKRANI
- SOĞUTUCUNUN KAPATILMASI EKRANI
- SICAKLIĞIN GÖRÜNTÜLENMESİ EKRANI
- VERİTABANI BİLGİLERİ
- DEPENDENCY INVERSION DESIGN
- BUILDER DESIGN
- OBSERVER DESIGN
- GİTHUB ADRESİ
- YOUTUBE VIDEO ADRESİ

## KULLANICI DOĞRULAMA EKRANI

Uygulama çalıştırıldığında ilk olarak aşağıdaki ekran çıkar.

```
Run: Main x
C:\Users\akyol61\.jdk\openjdk-14.0.1\bin\java.exe
-----
Author = Raif Akyol
Date = 08.05.2001
Version = 1.0
-----
Kullanıcı adı ve şifre girin.
Kullanıcı Adı:|
```

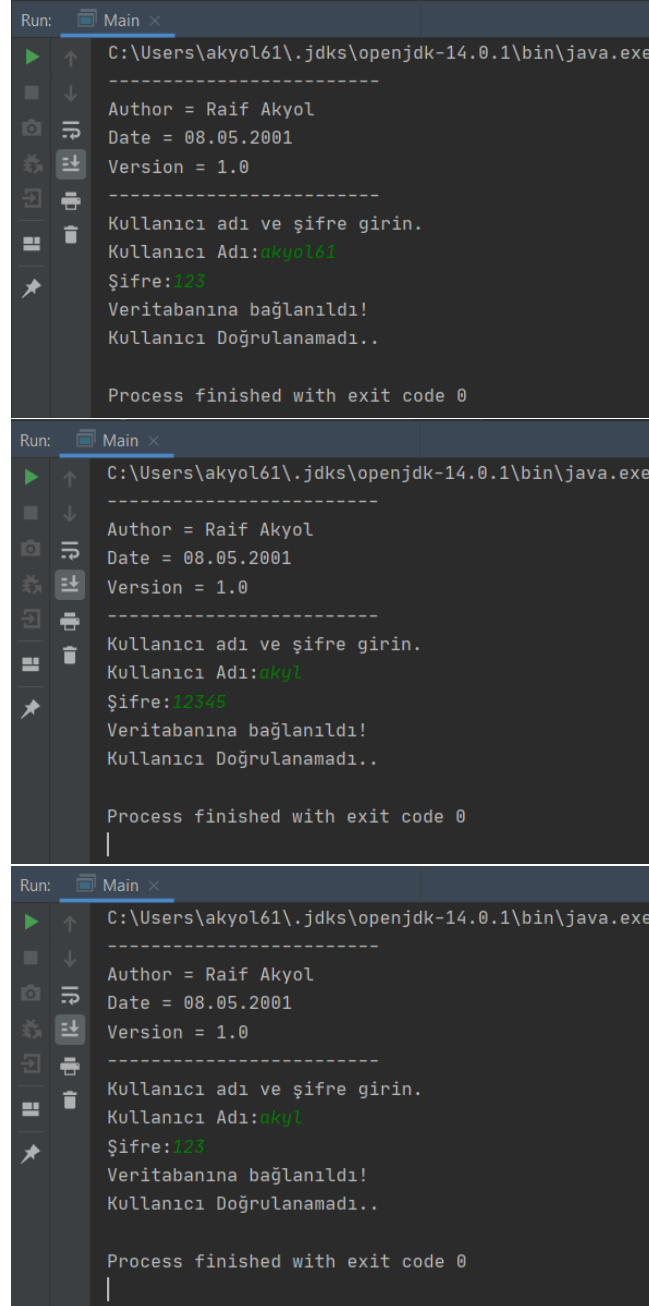
Kullanıcıdan veri tabanına kaydettiği Kullanıcı Adını ve Şifresini girmesini istenir.

```
Run: Main x
C:\Users\akyol61\.jdk\openjdk-14.0.1\bin\java.exe
-----
Author = Raif Akyol
Date = 08.05.2001
Version = 1.0
-----
Kullanıcı adı ve şifre girin.
Kullanıcı Adı:akyol61
Şifre:12345
Veritabanına bağlandı!
Kullanıcı doğrulandı!
-----
1-Sogutucu Ac
2-Sogutucu Kapat
3-Sıcaklık Görüntüle
4-Cıkıs
-----
```

Data Output	Explain	Messages	Notifications
<div><div></div><div>id</div><div>[PK] integer</div></div>	<div><div></div><div>kullaniciAdi</div><div>character varying (50)</div></div>	<div><div></div><div>sifre</div><div>character varying (50)</div></div>	
1	1	akyol61	12345

Kullanıcı Adı ve Şifre girilir. Sonra veri tabanına bağlanılır. Ekrana VeriTabanına Bağlanıldı! Şeklinde kullanıcının görmesi için mesaj yazılır. Kullanıcı Adı ve Şifre veri tabanındaki bilgilere uygun girildiğinde kullanıcı doğrulanır ve ekrana Kullanıcı Doğrulandı! Şeklinde mesaj yazılır ve işlem menüsü gelir.

Eğer kullanıcı, kullanıcı adı ve şifresinden herhangi birini veya hepsini yanlış yazmış olursa,



```
Run: Main x
C:\Users\akyol61\.jdk\openjdk-14.0.1\bin\java.exe
-----
Author = Raif Akyol
Date = 08.05.2001
Version = 1.0
-----
Kullanıcı adı ve şifre girin.
Kullanıcı Adı:akyol61
Şifre:123
Veritabanına bağlanıldı!
Kullanıcı Doğrulanamadı..

Process finished with exit code 0

Run: Main x
C:\Users\akyol61\.jdk\openjdk-14.0.1\bin\java.exe
-----
Author = Raif Akyol
Date = 08.05.2001
Version = 1.0
-----
Kullanıcı adı ve şifre girin.
Kullanıcı Adı:akyol
Şifre:12345
Veritabanına bağlanıldı!
Kullanıcı Doğrulanamadı..

Process finished with exit code 0

Run: Main x
C:\Users\akyol61\.jdk\openjdk-14.0.1\bin\java.exe
-----
Author = Raif Akyol
Date = 08.05.2001
Version = 1.0
-----
Kullanıcı adı ve şifre girin.
Kullanıcı Adı:akyol
Şifre:123
Veritabanına bağlanıldı!
Kullanıcı Doğrulanamadı..

Process finished with exit code 0
```

Önce veri tabanına bağlanılır kullanıcın girdiği bilgilerin doğruluğu kontrol edilir. Bilgiler uyuşmadığından Kullanıcı Doğrulanamadı! şeklinde ekrana mesaj yazılır ve program sonlandırılır.

## SOĞUTUCUNUN AÇILMASI EKRANI

```
Run: Main x
C:\Users\akyo161\.jdk\openjdk-14.0.1\bin\java.exe
-----
Author = Raif Akyol
Date = 08.05.2001
Version = 1.0
-----
Kullanıcı adı ve şifre girin.
Kullanıcı Adı:akyo161
Şifre:12345
Veritabanına bağlanıldı!
Kullanıcı doğrulandı!
-----
1-Sogutucu Ac
2-Sogutucu Kapat
3-Sıcaklık Görüntüle
4-Cıkıs
-----
1
Soğutucu Açılıyor!
Lütfen Bekleyiniz.
SOĞUTUCU : AÇIK

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

Kullanıcı, uygulamaya giriş yapar. Kullanıcı adı ve şifresini veri tabanındaki kayıtlara uygun olarak girdiğinde önce veri tabanına bağlanılır kullanıcı adı ve şifre doğru olduğundan ekrana Kullanıcı doğrulandı mesajı yazılır ve menü ekranı gelir. Console 'a Soğutucu Aç işleminin numarası olan "1" yazılırsa soğutucunun açılması işleminin yapılmasına başlanır. İlk olarak ekrana "Soğutucu Açılıyor!" şeklinde mesaj yazılır. İşlem devam ederken kullanıcının beklemesi için "Lütfen Bekleyiniz." şeklinde mesaj yazılır. İşlem tamamlandığında soğutucunun durumunun kullanıcıya bildirilmesi için "SOĞUTUCU : AÇIK" şeklinde bilgi mesajı yazılır.

Kullanıcı başka işlem yapmak isteyebilir bunun için Devam Etmek İstiyormusunuz e(E)/h(H) şeklinde bir seçim ekranı çıkar kullanıcının tercihinine göre işlem yapılır. Kullanıcı işlem yapmaya devam etmek isterse console' a "e" ya da "E" yazması yeterlidir. "e" ya da "E" yazarsa menü tekrar ekrana yazılır ve işlem seçilir. Kullanıcı işlem yapmak istemiyorsa console' a "h" ya da "H" yazması yeterlidir. "h" ya da "H" yazarsa ekrana "Çıkış Yapılıyor!" şeklinde mesaj yazılır ve program sonlanır.

```
-----  
Kullanıcı adı ve şifre girin.  
Kullanıcı Adı:akyaol61  
Şifre:12345  
Veritabanına bağlanıldı!  
Kullanıcı doğrulandı!  
-----  
1-Sogutucu Ac  
2-Sogutucu Kapat  
3-Sıcaklık Görüntüle  
4-Cıkıs  
-----  
1  
Soğutucu Açılıyor!  
Lütfen Bekleyiniz.  
SOĞUTUCU : AÇIK  
  
Devam Etmek İstiyormusunuz? e(E)/h(H)  
E  
-----  
1-Sogutucu Ac  
2-Sogutucu Kapat  
3-Sıcaklık Görüntüle  
4-Cıkıs  
-----
```

```
-----  
Kullanıcı adı ve şifre girin.  
Kullanıcı Adı:akyaol61  
Şifre:12345  
Veritabanına bağlanıldı!  
Kullanıcı doğrulandı!  
-----  
1-Sogutucu Ac  
2-Sogutucu Kapat  
3-Sıcaklık Görüntüle  
4-Cıkıs  
-----  
1  
Soğutucu Açılıyor!  
Lütfen Bekleyiniz.  
SOĞUTUCU : AÇIK  
  
Devam Etmek İstiyormusunuz? e(E)/h(H)  
h  
Çıkış yapılıyor!  
  
Process finished with exit code 0  
|
```

## SOĞUTUCUNUN KAPATILMASI EKRANI

```
Run: Main x
C:\Users\akyol61\.jdk\openjdk-14.0.1\bin\java.exe
-----
Author = Raif Akyol
Date = 08.05.2001
Version = 1.0
-----
Kullanıcı adı ve şifre girin.
Kullanıcı Adı:akyol61
Şifre:12345
Veritabanına bağlanıldı!
Kullanıcı doğrulandı!
-----
1-Sogutucu Ac
2-Sogutucu Kapat
3-Sıcaklık Görüntüle
4-Cıkıs
-----
2
Soğutucu Kapatılıyor!
Lütfen Bekleyiniz.
SOĞUTUCU : KAPALI

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

Kullanıcı, uygulamaya giriş yapar. Kullanıcı adı ve şifresini veri tabanındaki kayıtlara uygun olarak girdiğinde önce veri tabanına bağlanılır kullanıcı adı ve şifre doğru olduğundan ekrana Kullanıcı doğrulandı mesajı yazılır ve menü ekranı gelir. Console 'a Soğutucu Kapat işleminin numarası olan "2" yazılırsa soğutucunun kapatılması işleminin yapılmasına başlanır. İlk olarak ekrana "Soğutucu Kapatılıyor!" şeklinde mesaj yazılır. İşlem devam ederken kullanıcının beklemesi için "Lütfen Bekleyiniz." şeklinde mesaj yazılır. İşlem tamamlandığında soğutucunun durumunun kullanıcıya bildirilmesi için "SOĞUTUCU : KAPALI" şeklinde bilgi mesajı yazılır.

Kullanıcı başka işlem yapmak isteyebilir bunun için Devam Etmek İstiyormusunuz e(E)/h(H) şeklinde bir seçim ekranı çıkar kullanıcının tercihinine göre işlem yapılır. Kullanıcı işlem yapmaya devam etmek isterse console' a "e" ya da "E" yazması yeterlidir. "e" ya da "E" yazarsa menü tekrar ekrana yazılır ve işlem seçilir. Kullanıcı işlem yapmak istemiyorsa console' a "h" ya da "H" yazması yeterlidir. "h" ya da "H" yazarsa ekrana "Çıkış Yapılıyor!" şeklinde mesaj yazılır ve program sonlanır.



```
-----  
Kullanıcı adı ve şifre girin.  
Kullanıcı Adı:akyo161  
Şifre:12345  
Veritabanına bağlanıldı!  
Kullanıcı doğrulandı!  
-----  
1-Sogutucu Ac  
2-Sogutucu Kapat  
3-Sıcaklık Görüntüle  
4-Cıkıs  
-----  
2  
Soğutucu Kapatılıyor!  
Lütfen Bekleyiniz.  
SOĞUTUCU : KAPALI  
  
Devam Etmek İstiyormusunuz? e(E)/h(H)  
e  
-----  
1-Sogutucu Ac  
2-Sogutucu Kapat  
3-Sıcaklık Görüntüle  
4-Cıkıs  
-----
```



```
-----  
Kullanıcı adı ve şifre girin.  
Kullanıcı Adı:akyo161  
Şifre:12345  
Veritabanına bağlanıldı!  
Kullanıcı doğrulandı!  
-----  
1-Sogutucu Ac  
2-Sogutucu Kapat  
3-Sıcaklık Görüntüle  
4-Cıkıs  
-----  
2  
Soğutucu Kapatılıyor!  
Lütfen Bekleyiniz.  
SOĞUTUCU : KAPALI  
  
Devam Etmek İstiyormusunuz? e(E)/h(H)  
H  
Çıkış yapılıyor!  
  
Process finished with exit code 0  
|
```



## SICAKLIĞIN GÖRÜNTÜLENMESİ EKRANI

```
Run: Main x
C:\Users\akyol61\.jdk\openjdk-14.0.1\bin\java.exe "-javaagent:C:\Program Files\
-----
Author = Raif Akyol
Date = 08.05.2001
Version = 1.0
-----
Kullanıcı adı ve şifre girin.
Kullanıcı Adı:akyol61
Şifre:12345
Veritabanına bağlanıldı!
Kullanıcı doğrulandı!
-----
1-Sogutucu Ac
2-Sogutucu Kapat
3-Sıcaklık Görüntüle
4-Cıkıs
-----
3
Sıcaklık Ölçülüyor.
Sıcaklık: 67
Abone1 e gelen mesaj->Sıcaklık 45 derecenin üzerine çıktı, soğutucuyu açınız.
Abone2 ye gelen mesaj->Sıcaklık 45 derecenin üzerine çıktı, soğutucuyu açınız.

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

Kullanıcı, uygulamaya giriş yapar. Kullanıcı adı ve şifresini veri tabanındaki kayıtlara uygun olarak girdiğinde önce veri tabanına bağlanılır kullanıcı adı ve şifre doğru olduğundan ekrana Kullanıcı doğrulandı mesajı yazılır ve menü ekranı gelir. Console 'a Sıcaklık Görüntüle işleminin numarası olan "3" yazılırsa sıcaklığın görüntülenmesi işleminin yapılmasına başlanır. İlk olarak ekrana "Sıcaklık Ölçülüyor." şeklinde mesaj yazılır. Random olarak hesaplanan sıcaklık değeri "Sıcaklık : **DEĞER**" olarak ekrana yazılır.

Eğer random sıcaklık değeri 45 dereceye eşit ve büyükse ekrana "Abone1 ve Abone2 ye gelen mesaj-> Sıcaklık 45 derecenin üzerine çıktı, soğutucuyu açınız." Şeklinde uyarı mesajı yazılır.

Eğer random sıcaklık değeri 25 dereceye eşit ve büyük; 45 dereceden küçükse ekrana "Abone1 ve Abone2 ye gelen mesaj-> Soğutucuyu açabilirsiniz." Şeklinde uyarı mesajı yazılır.

Eğer random sıcaklık değeri 25 dereceye eşit ve küçükse ekrana "Abone1 ve Abone2 ye gelen mesaj-> Sıcaklık 25 derecenin altına indi , soğutucuyu kapatınız." Şeklinde uyarı mesajı yazılır.

Kullanıcı başka işlem yapmak isteyebilir bunun için Devam Etmek İstiyormusunuz e(E)/h(H) şeklinde bir seçim ekranı çıkar kullanıcının tercihinine göre işlem yapılır. Kullanıcı işlem yapmaya devam etmek isterse console' a "e" ya da "E" yazması yeterlidir. "e" ya da "E" yazarsa menü tekrar ekrana yazılır ve işlem seçilir. Kullanıcı işlem yapmak istemiyorsa console' a "h" ya da "H" yazması yeterlidir. "h" ya da "H" yazarsa ekrana "Çıkış Yapılıyor!" şeklinde mesaj yazılır ve program sonlanır.


```
Run: Main x
C:\Users\akyo161\.jdk\openjdk-14.0.1\bin\java.exe "-javaagent:C:\Program Files\
-----
Author = Raif Akyol
Date = 08.05.2001
Version = 1.0
-----
Kullanıcı adı ve şifre girin.
Kullanıcı Adı:akyo161
Şifre:12345
Veritabanına bağlanıldı!
Kullanıcı doğrulandı!
-----
1-Sogutucu Ac
2-Sogutucu Kapat
3-Sıcaklık Görüntüle
4-Cıkıs
-----
3
Sıcaklık Ölçülüyor.
Sıcaklık: 21
Abone1 e gelen mesaj->Sıcaklık 25 derecenin altına indi, soğutucuyu kapatınız.
Abone2 ye gelen mesaj->Sıcaklık 25 derecenin altına indi, soğutucuyu kapatınız.

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

```
Run: Main x
C:\Users\akyo161\.jdk\openjdk-14.0.1\bin\java.exe "-javaagent:C:\Program
-----
Author = Raif Akyol
Date = 08.05.2001
Version = 1.0
-----
Kullanıcı adı ve şifre girin.
Kullanıcı Adı:akyo161
Şifre:12345
Veritabanına bağlanıldı!
Kullanıcı doğrulandı!
-----
1-Sogutucu Ac
2-Sogutucu Kapat
3-Sıcaklık Görüntüle
4-Cıkıs
-----
3
Sıcaklık Ölçülüyor.
Sıcaklık: 44
Abone1 e gelen mesaj->Soğutucuyu açabilirsiniz.
Abone2 ye gelen mesaj->Soğutucuyu açabilirsiniz.

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





## VERİTABANI BİLGİLERİ


 public.KullaniciGiris/proje2021/postgres@PostgreSQL 13

Query Editor   Query History

```
1 SELECT * FROM public."KullaniciGiris"
2 ORDER BY id ASC
```


Data Output   Explain   Messages   Notifications







	 id [PK] integer 	kullaniciAdi character varying (50) 	sifre character varying (50) 
1	1	akyol61	12345

KullaniciGiris 

General   Columns   Advanced   Constraints   Parameters   Security   SQL

Inherited from table(s)

Columns 

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
 	id	integer			<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Yes
 	kullaniciAdi	character varying	50		<input type="checkbox"/> No	<input type="checkbox"/> No
 	sifre	character varying	50		<input type="checkbox"/> No	<input type="checkbox"/> No

```
--
-- PostgreSQL database dump
--

-- Dumped from database version 13.1
-- Dumped by pg_dump version 13.1

-- Started on 2021-05-09 02:20:00

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

SET default_tablespace = '';

SET default_table_access_method = heap;

--
-- TOC entry 200 (class 1259 OID 42028)
-- Name: KullaniciGiris; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public."KullaniciGiris" (
    id integer NOT NULL,
```

```

    "kullaniciAdi" character varying(50),
    sifre character varying(50)
);

ALTER TABLE public."KullaniciGiris" OWNER TO postgres;

--
-- TOC entry 2980 (class 0 OID 42028)
-- Dependencies: 200
-- Data for Name: KullaniciGiris; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY public."KullaniciGiris" (id, "kullaniciAdi", sifre) FROM stdin;
1      akyol61      12345
\.

--
-- TOC entry 2849 (class 2606 OID 42032)
-- Name: KullaniciGiris KullaniciGiris_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public."KullaniciGiris"
    ADD CONSTRAINT "KullaniciGiris_pkey" PRIMARY KEY (id);

-- Completed on 2021-05-09 02:20:00

--
-- PostgreSQL database dump complete
--

```

## DEPENDENCY INVERSION DESIGN

Üst seviyeli işlem yapan sınıflar(High Level Class – Yüksek Dereceli Sınıf), alt seviyeli işlem yapan sınıflara(Low Level Class – Düşük Dereceli Sınıf) bağımlı olmaktadır. Bir başka deyimle üst seviyeli işlem yapan metodlar, alt seviyeli işlem yapan metodları kullanmaktadırlar. Haliyle alt seviye metodlarda olası her değişikliğin üst seviye metodlarda değişikliğe sebep olması üst seviyenin alt seviyeye bağımlılığını göstermektedir. Dependency Inversion Design bize bu bağımlılığın ters çevrilmesini prensip edinmemizi önermektedir.



İlk olarak ac() ve kapat() şeklinde metodların olduğu IDip isminde bir interface tanımlanır. Araya eklenen bu arayüz bağımlılığı azaltır.

```
package proje21;

public class Dip {
    private IDip dip;

    public Dip(IDip dip) { this.dip = dip; }

    public void ac() { dip.ac(); }

    public void kapat() { dip.kapat(); }
}
```

Dip şeklinde bir class tanımlayıp hiçbir değişiklik yapmadan başka işlemlerin kontrolünde kullanılabilir (Eyleyici, Televizyon, Su ısıtıcısı, Lamba vb.)

```

1 package proje21;
2
3 public class Eyleyici implements IDip{
4     private AkilliCihaz cihaz;
5     @Override
6     public void ac() {
7         System.out.println("Soğutucu Açılıyor!");
8         cihaz.bekle( sure: 1000);
9         System.out.println("Lütfen Bekleyiniz.");
10        cihaz.bekle( sure: 1000);
11        System.out.println("SOĞUTUCU : AÇIK");
12    }
13
14    @Override
15    public void kapat() {
16        System.out.println("Soğutucu Kapatılıyor!");
17        cihaz.bekle( sure: 1000);
18        System.out.println("Lütfen Bekleyiniz.");
19        cihaz.bekle( sure: 1000);
20        System.out.println("SOĞUTUCU : KAPALI");
21    }
22 }

```

Eyleyici aldığı parametreyle soğutucuyu açıp kapatabiliyor. Örneğin Eyleyici Class 1 gibi Televizyon için aç kapa işlemleri IDip ve Dip modülünü değiştirmeden yeni bir class açıp ac() kapat() metodlarıyla işlem yapmamızı sağlar.

```

MerkeziIslemBirimi.java x
1  package proje21;
2
3  import java.lang.reflect.Method;
4  import java.util.Random;
5  import java.util.concurrent.Callable;
6  public class MerkeziIslemBirimi implements IMerkeziIslemBirimi{
7      private ISicaklikAlgilayici sicaklikAlgilayici;
8      public MerkeziIslemBirimi(){
9          sicaklikAlgilayici=SicaklikAlgilayici.getInstance(new Publisher());
10         sicaklikAlgilayici.aboneEkle(new Subscriber1());
11         sicaklikAlgilayici.aboneEkle(new Subscriber2());
12     }
13     @Override
14     public void eyleyiciyeGonder(){
15         IDip eylç=new Eyleyici();
16         Dip dip=new Dip(eylç);
17         dip.ac();
18     }
19     @Override
20     public void eyleyiciyeGonder2(){
21         IDip eylç=new Eyleyici();
22         Dip dip=new Dip(eylç);
23         dip.kapat();
24     }
25     @Override
26     public void sicaklikAlgilayiciyaGonder() { sicaklikAlgilayici.sicaklikOku(); }
27
28
29 }

```

Açma ve kapama işlemleri için eyleyiciyeGonder() ve eyleyiceyeGonder2() metodlarındaki gibi çağrılırsa açma ve kapama işlemleri gerçekleştirilir.

Bu şekilde yeni bir işlem yapacağımız zaman kodları değiştirmemize gerek kalmaz ve maliyet tasarrufu sağlanmış ve işlemlerimiz kolaylaşmış olur.



## BUILDER DESIGN

Nesne tabanlı programlamanın özü sınıflara dayanır. Sınıflardan nesneler yaratırız. Bunu yapmak için de constructorları kullanırız. Sınıfımızda bulunan field sayısı fazla olursa bundan dolayı birden çok constructora ihtiyaç duyabiliriz. Haliyle her bir field eklendiğinde yeni bir constructor ekleme ihtiyacı hissedebiliriz. Çünkü nesneyi oluştururken hangi field başta atama yapılacak ya da yapılmayacak bilemeyebiliriz. İşte bu uzayıp giden parametre sayısından, karmaşık constructorlardan kurtarmak için Builder Pattern güzel bir çözüm sunmaktadır.

```
Builder.java x
1  package proje21;
2
3  public class Builder {
4      private String author, date, version;
5
6      @private Builder(ClassBuilder builder) {
7          this.author = builder.author;
8          this.date = builder.date;
9          this.version = builder.version;
10     }
11     public String getAuthor() { return author; }
12
13     public String getDate() { return date; }
14
15     public String getVersion() { return version; }
16
17     @Override
18     public String toString() {
19         return "-----" +
20             "\nAuthor =" + author + "\n" +
21             "Date =" + date + "\n" +
22             "Version =" + version + "\n" +
23             "-----";
24     }
25
26     public static class ClassBuilder
27     {
28         private String author, date, version;
29         public ClassBuilder author (String author) {
30             this.author = author;
31             return this;
32         }
33         public ClassBuilder date (String date) {
34             this.date = date;
35             return this;
36         }
37         public ClassBuilder version (String version) {
38             this.version = version;
39             return this;
40         }
41         public Builder build() { return new Builder(this); }
42     }
43 }
44
45
46
47
48
```

Bu şekilde bir builder sınıfımız olduğunu varsayalım. İçinde author, date ve version değerlerini tutuyor.

Bu sınıftan bir nesne oluşturmamız için 3 alanı da constructor içinde göndermemiz gerekmekte. Yalnızca author ve date başta atama yapılacaksa, version bilgisine gerek yoksa author ve date alanlarını parametre olarak alan ayrı bir constructor yazarak sorunu çözerdik. Peki ya bu gibi 10 tane alan olsa ve nesneyi oluştururken hangi alanların başta atanacağını bilmiyorsa Her bir durum için constructor yazardık ancak bu içinden çıkılmaz bir hal alırdı. Builder Pattern bu gibi durumlara çözüm sunabiliyor.

Üstte Builder Pattern'nin sunduğu çözümü görebiliriz. Builder sınıfımızın içinde static bir inner class var ve bunun üstünden asıl nesnemizi oluşturuyoruz. Nesne oluştururken yalnızca istediğimiz alanlar ile birlikte oluşturabiliyoruz.

Nesnemizi istediğimiz alanlar ile oluşturmak artık bu kadar kolay. Builder Pattern kullanmadan Builder sınıfını default constructor ile oluşturarak, daha sonra fieldları setter metodları ile atama yapabiliriz. Ancak oluşturduğumuz her nesne için bunu yapmak kodun okunabilirliğini düşürdü ve bizim de gereksiz vaktimizi alırdı.

```
1 package proje21;
2
3 public class Main {
4     public static void main(String[] args) {
5         Builder builder1 = new Builder.ClassBuilder().author(" Raif Akyol").version(" 1.0").date(" 08.05.2001").build();
6         System.out.println(builder1);
7         AkilliCihaz akilliCihaz= new AkilliCihaz();
8         akilliCihaz.giris();
9     }
10 }
11 }
12
```

Main içinde author, version ve date String ifadelerine bilgiler yazdığımızda aşağıdaki gibi bilgi ekranı çıkar.

```
Run: Main x
C:\Users\akyol61\.jdk\openjdk-14.0.1\bin\java.exe
-----
Author = Raif Akyol
Date = 08.05.2001
Version = 1.0
-----
Kullanıcı adı ve şifre girin.
Kullanıcı Adı:
```

## OBSERVER DESIGN

Observer adından da anlaşılacağı üzere gözlemci, izleyici, gözcü yahut gözetmen diye nitelendirilen, anlamı gibi işlev gören bir tasarım desendir. Elimizdeki mevcut nesnenin durumunda herhangi bir değişiklik olduğunda, bu değişikliklerden diğer nesneleri haberdar eden sisteme Observer tasarım deseni denir.

Bir duyuru sistemimiz var ve kayıt olan kullanıcılara duyuru yapıldığında mesaj gönderilecek.

```
IObserver.java x
1 package proje21;
2 public interface IObserver {
3     public void update(String m);
4 }
```

yukarı da IObserver interface'i tanımladık duyurular bu interface üzerinden sağlanacak.

```
ISubject.java x
1 package proje21;
2 public interface ISubject {
3     public void attach(IObserver o);
4     public void detach(IObserver o);
5     public void notify(String m);
6 }
```

ISubject interface'i ise duyuru sisteminde ki kullanıcıların temsili olarak düşünebiliriz.

```
Publisher.java x
1 package proje21;
2 import java.util.ArrayList;
3 import java.util.List;
4
5 public class Publisher implements ISubject {
6     private List<IObserver> subscribers = new ArrayList<>();
7     @Override
8     public void attach(IObserver subscriber) {
9         subscribers.add(subscriber);
10    }
11    @Override
12    public void detach(IObserver subscriber) {
13        subscribers.remove(subscriber);
14    }
15    @Override
16    public void notify(String mesaj) {
17        for(IObserver subscriber: subscribers) {
18            subscriber.update(mesaj);
19        }
20    }
21 }
```

Publisher sınıfı ile ISubject interface'ini genişlettik ve gerekli özelleştirmeleri yaptık.

Attach ile kullanıcıları duyuruya eklemek için kullanılır.

Detach ile kullanıcıları duyurudan silmek için kullanılır.

Notify ile Duyuru ya kayıtlı kullanıcılara mesaj göndermek için kullanılır.

```
Subscriber1.java x
1 package proje21;
2
3 public class Subscriber1 implements IObserver {
4     @Override
5     public void update(String mesaj) {
6         System.out.println("Abone1 e gelen mesaj->" + mesaj);
7     }
8 }

Subscriber2.java x
1 package proje21;
2
3 public class Subscriber2 implements IObserver {
4     @Override
5     public void update(String mesaj) {
6         System.out.println("Abone2 ye gelen mesaj->" + mesaj);
7     }
8 }
```

Yukarıda ki iki tane kullanıcı sınıfımız var ikisi de IObserver interface'ini implement etmiş yani artık ikisi de birer gözlemci ve duyuru yapıldığında update() metoduna gelen parametre ile duyuru okunacaktır.

## **GİTHUB ADRESİ**

[https://github.com/raifakyol/AkilliCihaz\\_CoolerSystem](https://github.com/raifakyol/AkilliCihaz_CoolerSystem)

## **YOUTUBE ADRESİ**

<https://www.youtube.com/watch?v=BTn9AkHl8hQ>