T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

NESNE YÖNELİMLİ ANALİZ VE TASARIM PROJE RAPORU

B181210052 - ARSLANCAN SARIKAYA

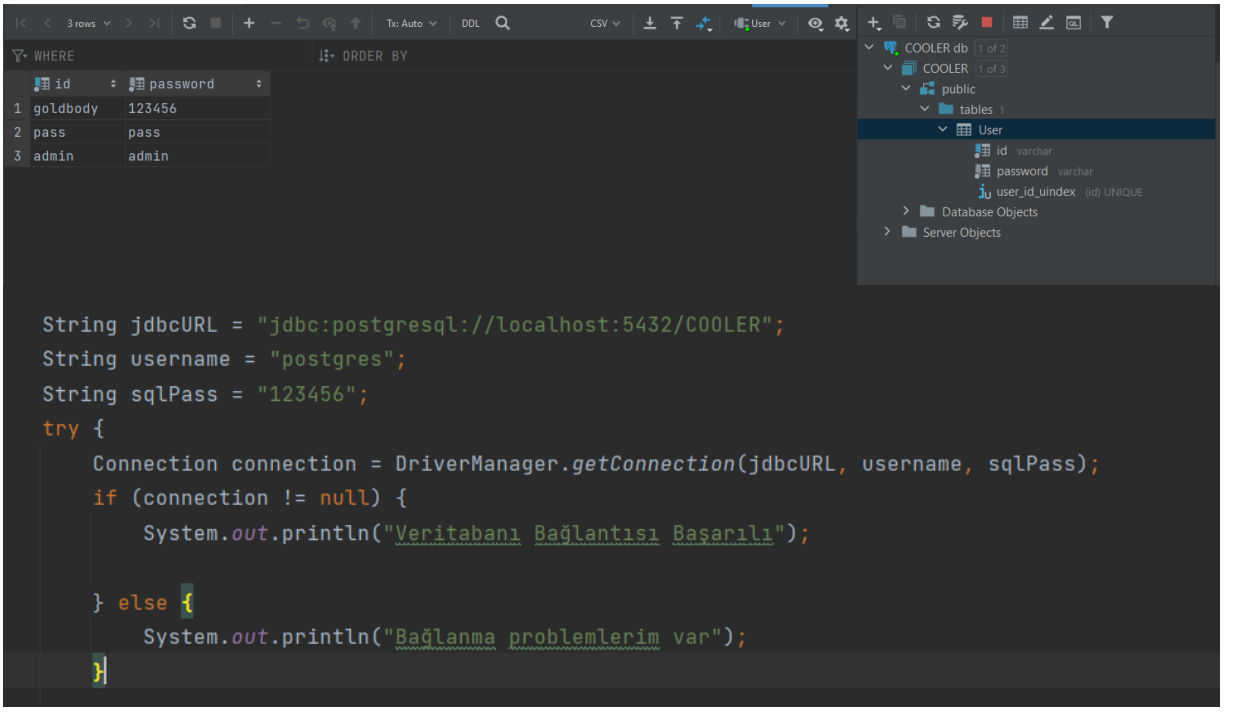
SAKARYA

MAYIS, 2021

Nesneye Yönelimi Analiz ve Tasarım Dersi

1- Veri Tabanı

Veri tabanı olarak PostgreSQL kullandım.



2- Uygulama Giriş Ekranı

a- Başarılı Giriş

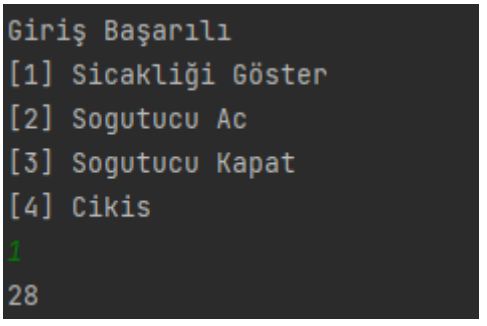
```
Veritabanı Bağlantısı Başarılı
Kullanici Adi..:admin
Sifre..:admin
Kullanıcı doğrulanıyor...
Veritabanına bağlandı ve kullanıcı hesap bilgilerini aldı...
Giriş Başarılı
[1] Sicakliği Göster
[2] Sogutucu Ac
[3] Sogutucu Kapat
[4] Cikis
```

b- Başarısız Giriş

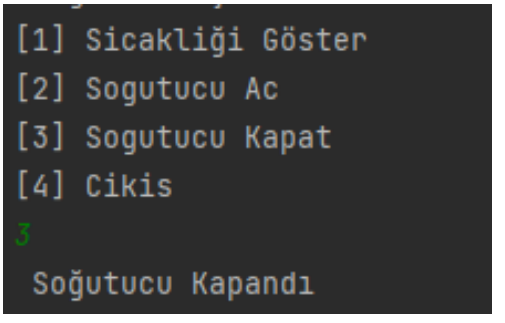
```
Veritabanı Bağlantısı Başarılı
Kullanici Adi..:celal
Sifre..:ceken
Kullanıcı doğrulanıyor...
Veritabanına bağlandı ve kullanıcı hesap bilgilerini aldı...
Giriş Başarısız

Process finished with exit code 0
```

3- Yazılım Menüsü ve İşlem Sonuçları

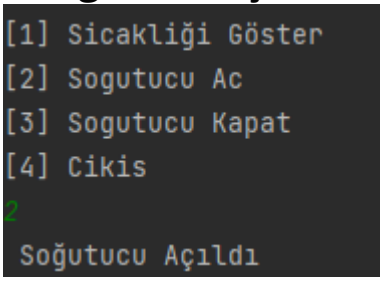
a-Sıcaklığı Göster



c- Soğutucu Kapat



c-Soğutucu Aç



4- Dependency Inversion Nedir?

Dependency Inversion, Classlar arasındaki bağımlılığı en aza indirmeyi esas alır. Bir alt sınıfa yapılan değişiklikler bir üst sınıfı etkilememelidir. Bunu sağlamak için Interface kullanımı önerilir.

Hem Observer yapısını gerçeklerken hem de diğer Classlarımı oluştururken Interface kullanarak dependency Inversionu gerçekledim.

5- Builder ve Observer Tasarım Desenleri

Builder Nedir?

Amaç, karmaşık bir nesnenin tanımını parçalara ayırmaktır. 10 tane fielda sahip bir classtan nesne oluştururken her özelliği girmek istemeyebiliriz ve bunu sağlamak için her bir olasılığa özgü kurucu fonksiyon yazmamız gerekir. İşte bu sorunu Builder ile ortadan kaldırabiliriz.

Bu ödevde Builder deseninin User classı içinde kullandım.

```java
public class User {
    private final String id;
    private final String password;
    private final String name;
    private final String surname;
```

```java
public User(Builder builder){
    this.id = builder.id;
    this.password= builder.password;
    this.name = builder.name;
    this.surname = builder.surname;
}

String getId() { return id; }

String getPassword() { return password; }

public static class Builder{
    private String id;
    private String password;
    private String name;
    private String surname;

    public Builder(){}

    public Builder id(String id){
        this.id = id;
        return this;
    }

    public Builder password(String password){
        this.password = password;
        return this;
    }

    public Builder name(String name){
        this.name = name;
        return this;
    }
    public Builder surname(String surname){
        this.surname = surname;
        return this;
    }
    public User build() { return new User( builder: this); }
```

Observer Nedir?

Amaç, Gözlenen nesnelerdeki değişiklikleri nesnelere bildirmektir.
Bu tasarım desenini projede User classını gözlemleyerek gerçekleştirdim.
Kullanıcı nesnesinde herhangi bir değişiklik olduğu zaman Screen classına bildirir ve notify methodu çalışır.

Youtube Link => https://www.youtube.com/watch?v=4vT6yYRR5ow