

```
import java.time.LocalDateTime;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.Scanner;
import java.sql.*;

public class Bekleme {
    private static final int sure = 100;
    public static void bekle(int sure){
        try
        {
            Thread.sleep(sure);
        }
        catch(InterruptedException ex)
        {
            Thread.currentThread().interrupt();
        }
    }
    public static void bekle(){
        try
        {
            Thread.sleep(sure);
        }
        catch(InterruptedException ex)
        {
            Thread.currentThread().interrupt();
        }
    }
}
```

```
public class Ekran implements IEkran{  
    public void mesajGoruntule(String mesaj){System.out.println(mesaj);}  
}
```

```
public class ekranUyari implements IObservable{  
  
    @Override  
    public void update(String uyari){System.out.println("Ekran'da" + uyari + "mesajı gösterildi.");}  
}
```

```
public interface IEkran {  
    public void mesajGoruntule(String mesaj);  
}
```

```
public interface IIslem {  
    public void islemYap();  
}
```

```
public interface IObservable {  
    public void update(String uyari);  
}
```

```
public interface ISubject {  
    public void attach(IObservable observer);  
    public void detach(IObservable observer);  
    public void notify(String uyari);  
}
```

```
public interface ITusTakimi {  
    public int veriAl();  
}
```

```
}
```

```
public class Kullanici{  
    private int kullaniciTuru;  
    private String kullaniciAdi;  
    private int kullaniciNumarasi;  
    private int sifre;  
    private IEkran ekran;  
  
    public Kullanici(String kullaniciAdi){  
        this.kullaniciAdi = kullaniciAdi;  
    }  
    public Kullanici(int kullaniciTuru,String kullaniciAdi,int kullaniciNumarasi,int sifre){  
        this.kullaniciTuru = kullaniciTuru;  
        this.kullaniciAdi = kullaniciAdi;  
        this.kullaniciNumarasi = kullaniciNumarasi;  
        this.sifre = sifre;  
    }  
    public Kullanici(String kullaniciAdi,int kullaniciTuru,int sifre){  
        this.kullaniciAdi = kullaniciAdi;  
        this.kullaniciTuru = kullaniciTuru;  
        this.sifre = sifre;  
    }  
    public int getKullaniciNumarasi(){  
        return kullaniciNumarasi;  
    }  
    public String getKullaniciAdi(){  
        return kullaniciAdi;  
    }  
    public int getKullaniciTuru(){  
        return kullaniciTuru;
```

```

    }

    public int getSifre(){
        return sifre;
    }

    @Override
    public String toString(){
        return "Kullanici Bilgileri {" +
            "kullaniciAdi:" + kullaniciAdi + "\" +
            ", kullaniciTuru:" + kullaniciTuru + "}";
    }

}

public class LogUyari implements IObservable{
    @Override
    public void update(String uyari){LogYoneticisi.getInstance("Log.txt").dosyayaYaz(uyari);}
}

public class LogYoneticisi{
    private static LogYoneticisi instance;
    PrintWriter out;

    private LogYoneticisi(String logDosyasi){
        try{
            out = new PrintWriter(new FileWriter(logDosyasi,true), true);
        }catch (IOException e){e.printStackTrace();}
    }

    public static synchronized LogYoneticisi getInstance(String logDosyasi){
        if(instance ==null)

```

```

        instance = new LogYoneticisi(logDosyasi);
        return instance;
    }
    public void dosyayaYaz(String mesaj){
        out.println(LocalDate.now() + ":" + mesaj);
    }
}

```

```

public class Publisher implements ISubject{
    private List<IObserver> observers = new ArrayList<>();

    @Override
    public void attach(IObserver observer){observers.add(observer);}

    @Override
    public void detach(IObserver observer){observers.remove(observer);}

    @Override
    public void notify(String uyari){
        for(IObserver observer: observers){
            observer.update(uyari);
        }
    }
}

```

```

public class SicaklikBelirleme {
    private int sicaklik;
    private int deger;
    ISubject publisher;

    public int SicaklikBelirleme(){

```

```
Random rastgele = new Random();  
sicaklik = rastgele.nextInt((60 - 0) + 1);  
return sicaklik;  
}
```

```
public SicaklikBelirleme(ISubject publisher, int deger){  
    this.deger = deger;  
    this.publisher = publisher;  
}
```

```
public int getDeger() {  
return deger;  
}  
  
public void setDeger(int eskiDeger) {  
this.deger = eskiDeger;  
publisher.notify("{ \"Sıcaklık\": \" " + eskiDeger + \"°C yapıldı.\" + \"}\"");  
}  
}
```

```
public class SicaklikGoruntuleme implements IIslem{  
    private IEkran ekran;  
    private ITusTakimi tusTakimi;  
    private Kullanici kullanici;  
    private SicaklikBelirleme sicaklik;  
  
    public SicaklikGoruntuleme(IEkran ekran, ITusTakimi tusTakimi, Kullanici kullanici, SicaklikBelirleme  
sicaklik)  
    {  
        this.sicaklik = sicaklik;  
        this.ekran = ekran;  
        this.tusTakimi = tusTakimi;
```

```
        this.kullanici = kullanici;
    }

    @Override
    public void islemYap(){
        ekran.mesajGoruntule("Sıcaklığı görüntüleme işlemini seçtiniz.");
        ekran.mesajGoruntule("Sıcaklık değeri:..." + " " + sicaklik.SicaklikBelirleme() + "°C" );
    }
}
```

```
public class SogutucuDenetleyisi {
    private IEkran ekran;
    private ITusTakimi tusTakimi;
    private ISubject publisher;

    private static final int SICAKLIK_GORUNTULEME = 1;
    private static final int SOGUTUCUYU_ACMA = 2;
    private static final int SOGUTUCUYU_KAPATMA = 3;
    private static final int CIKIS = 4;

    public SogutucuDenetleyisi(){
        ekran = new Ekran();
        tusTakimi = new TusTakimi();
    }

    public void Basla(){
        int kullaniciNumarasi = 0;
        int sifre = 0;
        String kullaniciAdi = null;
```

```
Kullanici kullanici = null;
```

```
Kullanici kullanicci = null;
```

```
VeriTabaniYonetimSistemi veritabani = new VeriTabaniYonetimSistemi();
```

```
ekran.mesajGoruntule("Giriş ekranına yönlendiriliyorsunuz..");
```

```
Bekleme.bekle(2000);
```

```
ekran.mesajGoruntule("Kullanıcı numaranızı giriniz: ");
```

```
kullaniciNumarasi = tusTakimi.veriAl();
```

```
ekran.mesajGoruntule("Şifrenizi giriniz: ");
```

```
sifre = tusTakimi.veriAl();
```

```
kullanici = veritabani.kullaniciKontrol(kullaniciNumarasi,sifre);
```

```
Bekleme.bekle(1000);
```

```
if(kullanici != null){
```

```
    ekran.mesajGoruntule("Kullanıcı doğrulandı." + kullanici);
```

```
    ekran.mesajGoruntule("Başarılı bir şekilde giriş yaptınız. Menü açılıyor..");
```

```
    Bekleme.bekle(1000);
```

```
    islemSecimi(kullanici);
```

```
}
```

```
else
```

```
    ekran.mesajGoruntule("Kullanıcı doğrulanamadı.");
```

```
}
```

```
private void islemSecimi(Kullanici kullanici){
```

```
    int secim;
```

```
    do{
```

```
    secim=anaMenuyuGoster();
```

```
    switch (secim){
```



```
case SICAKLIK_GORUNTULEME:
    LogYoneticisi.getInstance("Log.txt").dosyayaYaz("Sicaklik goruntulendi.");
    SicaklikBelirleme sicaklik = new SicaklikBelirleme(publisher,0);
    IIslem sicaklikgoruntuleme = new SicaklikGoruntuleme(ekran, tusTakimi, kullanıcı,
sicaklik);
    sicaklikgoruntuleme.islemYap();
break;
```

```
case SOGUTUCUYU_ACMA:
```

```
    LogYoneticisi.getInstance("Log.txt").dosyayaYaz("Sogutucu acildi.");
    Scanner giris = new Scanner(System.in);
```

```
    ekranUyari uyariEkran = new ekranUyari();
```

```
    Yoneticici yoneticici = new Yoneticici();
```

```
    LogUyari log = new LogUyari();
```

```
    Publisher publisher = new Publisher();
```

```
    publisher.attach(log);
```

```
    publisher.attach(uyariEkran);
```

```
    publisher.attach(yoneticici);
```

```
    SicaklikBelirleme degisenSicaklik = new SicaklikBelirleme(publisher,0);
```

```
    IIslem sogutucuyuAcma = new SogutucuyuAcma(ekran, tusTakimi, kullanıcı,publisher);
```

```
    sogutucuyuAcma.islemYap();
```

```
    ekran.mesajGoruntule("Soğutucuyu ayarlamak istediğiniz sıcaklığı giriniz.");
```

```
    int deger = tusTakimi.veriAl();
```

```
    ekran.mesajGoruntule("Sıcaklık ayarlanıyor...");
```

```
Bekleme.bekle(1000);
```

```
degisenSicaklik.setDeger(deger);
```

```
break;
```

```
case SOGUTUCUYU_KAPATMA:
```

```
LogYoneticisi.getInstance("Log.txt").dosyayaYaz("Sogutucu kapatildi.");
```

```
Islem sogutucuyuKapatma = new SogutucuyuKapatma(ekran, tusTakimi, kullanıcı);
```

```
sogutucuyuKapatma.islemYap();
```

```
break;
```

```
case CIKIS:
```

```
LogYoneticisi.getInstance("Log.txt").dosyayaYaz("Cikis yapildi.");
```

```
ekran.mesajGoruntule("Çıkılıyor..");
```

```
Bekleme.bekle(1000);
```

```
break;
```

```
default:
```

```
ekran.mesajGoruntule("1-4 arasında bir sayı girmelisiniz");
```

```
}
```

```
    }while(secim!=4);
```

```
}
```

```
private int anaMenuyuGoster()
```

```
{
```

```
    ekran.mesajGoruntule("*****");
```

```
    ekran.mesajGoruntule("ANA MENÜ");
```

```
        ekran.mesajGoruntule("1-Sıcaklığı Görüntüle");
        ekran.mesajGoruntule("2-Soğutucuyu Aç");
        ekran.mesajGoruntule("3-Soğutucuyu Kapat");
        ekran.mesajGoruntule("4-Çıkış");
        ekran.mesajGoruntule("Seçiminiz: ");
        ekran.mesajGoruntule("*****");
        return tusTakimi.veriAl();
    }
}
```

```
public class SogutucuyuAcma implements IIslem{
    private IEkran ekran;
    private ITusTakimi tusTakimi;
    private Kullanici kullanıcı;
    private ISubject publisher;
    private int deger;

    public SogutucuyuAcma(IEkran ekran,ITusTakimi tusTakimi,Kullanici kullanıcı,ISubject publisher )
    {
        this.ekran = ekran;
        this.kullanıcı = kullanıcı;
        this.tusTakimi = tusTakimi;
        this.publisher = publisher;
    }

    public int getDeger()
    {
        return deger;
    }

    public void setDeger(int deger)
```

```
{
    this.deger = deger;
    publisher.notify("Sıcaklık" + " " + deger + "°C'ye ayarlandı.");
}

@Override
public void islemYap(){
    ekran.mesajGoruntule("Soğutucuyu Açma İşlemini Seçtiniz.");
}
}

public class SogutucuyuKapatma implements IIslem{
    private IEkran ekran;
    private ITusTakimi tusTakimi;
    private Kullanici kullanici;

    public SogutucuyuKapatma(IEkran ekran, ITusTakimi tusTakimi, Kullanici kullanici )
    {
        this.ekran = ekran;
        this.kullanici = kullanici;
        this.tusTakimi = tusTakimi;
    }

    @Override
    public void islemYap(){
        ekran.mesajGoruntule("Soğutucuyu Kapatma İşlemini Seçtiniz.");
        ekran.mesajGoruntule("Soğutucu kapatılıyor...");
        Bekleme.bekle(1000);
    }
}
```

```
public class TusTakimi implements ITusTakimi {  
    public int veriAl()  
    {  
        Scanner input = new Scanner(System.in);  
        return input.nextInt();  
    }  
}
```

```
public class VeriTabaniYonetimSistemi {
```

```
    private Connection baglan(){  
  
        Connection conn=null;  
  
        try {  
            conn =  
DriverManager.getConnection("jdbc:postgresql://localhost:5432/KullanniciDogrulama",  
                "postgres", "hllmrvt");  
            if (conn != null){  
                System.out.println("Veritabanına bağlandı. Kullanıcı bilgileri doğrulanıyor..");  
                Bekleme.bekle(2000);  
            }  
            else  
                System.out.println("Bağlantı girişimi başarısız.");  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
        return conn;  
    }
```

```
}
```

```
public Kullanici kullaniciKontrol(int kullaniciNo, int girilenSifre){
```

```
    System.out.println("Kullanıcı numarası kontrol ediliyor...");
```

```
    Bekleme.bekle(2000);
```

```
    Kullanici kullanici=null;
```

```
    String sql= "SELECT * FROM \"Kullanici\" WHERE \"kullaniciNumarasi\"="+kullaniciNo + "AND  
sifre=" +girilenSifre ;
```

```
    Connection conn=this.baglan();
```

```
    try{
```

```
        Statement stmt = conn.createStatement();
```

```
        ResultSet rs = stmt.executeQuery(sql);
```

```
        //***** Bağlantı sonlandırma *****
```

```
        conn.close();
```

```
        int kullaniciTuru;
```

```
        String kullaniciAdi;
```

```
        int kullaniciNumarasi;
```

```
        while(rs.next())
```

```
        {
```

```
            kullaniciAdi = rs.getString("kullaniciAdi");
```

```
            kullaniciNumarasi = rs.getInt("kullaniciNumarasi");
```

```
            kullaniciTuru = rs.getInt("kullaniciTuru");
```

```
            kullanici = new Kullanici(kullaniciAdi ,kullaniciNumarasi, kullaniciTuru );
```

```
}
```

```
rs.close();
```

```
stmt.close();
```

```
} catch (Exception e) {
```

```
    e.printStackTrace();
```

```
}
```

```
return kullanıcı;
```

```
}
```

```
}
```

```
public class Yönetici implements IObservable{
```

```
    @Override
```

```
    public void update(String uyarı){System.out.println("Yöneticiye bildirilen mesaj:" + " " + uyarı);}
```

```
}
```

```
public class SoğutucuDenetleyisiSistemi {
```

```
    public static void main(String[] args) {
```

```
        SoğutucuDenetleyisi denetleyici = new SoğutucuDenetleyisi();
```

```
        denetleyici.Basla();
```

```
    }
```

```
}
```