

**NESNE YÖNELİMLİ ANALİZ VE TASARIM**

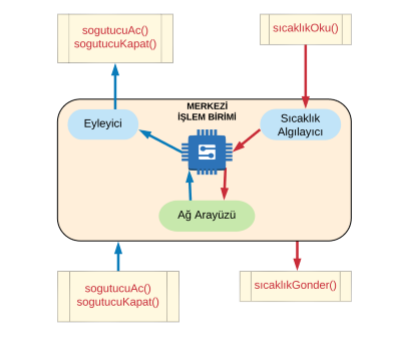
CELAL ÇEKEN

**G191210309**

**MUSTAFA KELEŞ**

**İKİNCİ ÖĞRETİM B GRUBU**

[**mustafa.keles4@ogr.sakarya.edu.tr**](mailto:mustafa.keles4@ogr.sakarya.edu.tr)



**Akıllı Cihazın Tanıtımı**

**• Akıllı cihaz, şekilde görüldüğü gibi, merkezi işlem birimi ile çevresel birimlerden (eyleyici, sıcaklık algılayıcısı ve ağ arayüzü) oluşmaktadır.**

**• Merkezi birim geliştirilecek yazılımın çalıştırılacağı birimdir. Çevre birimleri ile şekilde verilen arayüzler üzerinden haberleşmektedir.**

**• Sıcaklık algılayıcı modül ortam sıcaklığını ölçerek merkezi işlem birimine gönderir.**

**• Eyleyici modül soğutucunun açılması ve başlatılması işlemlerini yerine getirir.**

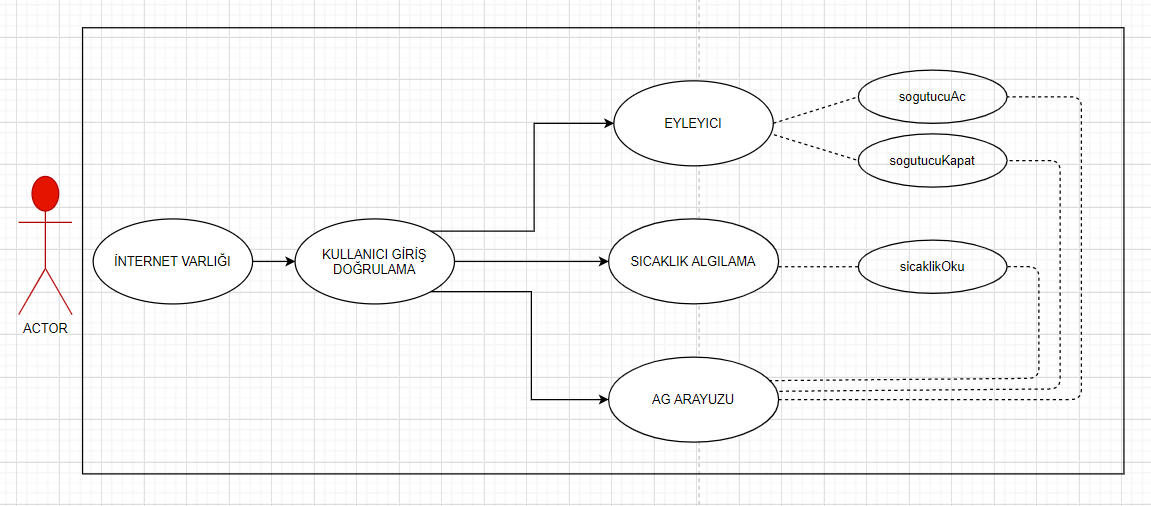
**• İnternet kullanıcıları ağ ara yüzünü kullanarak; sıcaklık görüntüleme, soğutucuyu açma ve soğutucuyu kapatma işlemlerini yerine getirebilirler.**

**• Akıllı cihaz; bekleme, kapalı, algılama ve kontrol gibi durumlara sahiptir.**

GÖREV A: Use Case Diagram

# Use Case Nedir?

Use Case, bir sistem aracılığı ile sunulan veya sunulacak **tek bir fonksiyonu** tanımlamaktadır. Diğer bir deyişle, kullanıcıların bir fonksiyon aracılığı ile sistem üzerinde gerçekleştirmek istedikleri işlemleri tanımlamaktadır.



**GÖREV B: İnternet üzerinden “sıcaklığın görüntülenmesi” ve “soğutucunun çalıştırılması” kullanım durumlarını metinsel olarak tanımlayınız.**

**AĞ ARAYÜZÜ**

**Tanım:** Sıcaklık görüntüleme, soğutucu açma ve soğutucu kapatma gibi kontrol işlemlerini gerçekleştirir.

**Ad**: Ağ Ara yüzü

**Hazırlayan**: Mustafa KELEŞ

**Sürüm**: M.3.1

**Tarih:** 04/03/2020

**İlgili Aktör:** Merkezi İşlem Birimi

**Giriş Koşulu**: Bir kullanıcı girişi doğrulanması gerekir, internet varlığı ve sıcaklık, soğukluk kontrolleri varlığı.

**Çıkış Koşulu: S**ıcaklık okunur geriye gönderir ve soğutucu istenen talebe göre işlemini yapar.

**Özel Gereksinim**: İnternetin varlığı, sıcaklık duruma cihazı sıcaklık ve soğukluk olma durumları.

**Olay Akışı:**

1.) Kullanıcı giriş doğrulanır.

2.) İnternet varlığı kontrol edilir.

3.) Merkezi İşlem Birimi talebi alır.

4.) Ağ ara yüzü istenen talebi alır.

5.) Ağ ara yüzü istenen talebi yapacak olana yönlendirir.

6.) Eğer yönlendirilen yer sıcaklık gör ise sıcaklık okunur.

7.) Sıcaklık okuma sonucu ağ ara yüzüne bildirir.

8.) Ağ ara yüzde Merkezi İşlem Birimi ne bildirir.

9.) Eğer yönlendirilen yer soğurucu açma ise soğutucu aç talebi alır.

10.) Talebi gerçekleştirip Ağ ara yüzüne bildirir.

11.) Ağ ara yüzde Merkezi İşlem Birimi ne bildirir.

12.) Eğer yönlendirilen yer soğutucu kapat ise soğutucu kapat talebi alır.

13.) Talebi gerçekleştirip Ağ ara yüzüne bildirir.

14.) Ağ ara yüzde Merkezi İşlem Birimi ne bildirir.

**Ağ Ara yüzü Alternatif Yolları:**

**C1)** Girilen kullanıcıyok ise (1)

2.) Hata mesajı verilir.

3.) İşlem sonlanır.

**C2)** İnternet yok ise (2)

3). İnternet ayarını düzenlemesi için geri bildirim yollanır.

4.) İşlem sonlandırılır.

**C3)** Sıcaklık okuyucu yok ise (6)

7.) Sıcaklık okuyucu yokluğu geri bildirilir.

8.) MİB okuyucu olmadığını ve sıcaklık okunamadığını bildirir.

9.) İşlem sonlandırılır.

**C4)** Açılacak soğutucu yok ise (9)

9.) Soğutucunun olmadığı geri bildirilir.

10.) MİB kendini sıcaklığa karşı korur.

11.) İşlem sonlandırılır.

**C5)** Kapatılacak bir soğutucu yok ise (12)

13.) Soğutucunun olmadığı geri bildirilir.

14.) MİB kendini sıcaklığa karşı korur.

15.) İşlem sonlandırılır.

**--------------------------------------------------------------------------------**

**EYLEYİCİ**

**Tanım:** Soğutucu kontrolü ve çalıştırılmasını sağlar.

**Ad**: Eyleyici

**Hazırlayan**: Mustafa KELEŞ

**Sürüm**: M.1.1

**Tarih:** 02/03/2020

**İlgili Aktör:** Merkezi İşlem Birimi

**Giriş Koşulu:** Bir kullanıcı girişi doğrulanması gerekir ve bir soğutucu vardır açılıp kapanabilir.

**Çıkış Koşulu**: Soğutucu açılıp kapanır.

**Özel Gereksinimler:** Soğutucuların varlığı, sürekli çalışması.

**Olay Akışı:**

1. Kullanıcı giriş doğrulanır.
2. Merkezi İşlem Birim talebi alır.
3. Merkezi İşlem Birimi eyleyiciye talebi bildirir.
4. Merkezi İşlem Birimi soğutucunun varlığını kontrol eder.
5. Eyleyici yapılacak işlemi yönlendirir.
6. Soğutucu gelen talebi alır.
7. Gelen talebe göre soğutucuyu açar veya kapatır.
8. Açılıp kapanma durumunu eyleyiciye bildirir.
9. Eyleyicide Merkezi İşlem Birimine bildirir.

**Alternatif Yollar:**

**A1)** Girilen kullanıcıyok ise (1)

2.) Hata mesajı verilir.

3.) İşlem sonlanır.

**A2)** Soğutucu yok ise (4)

5.) Soğutucunun yokluğunu geri bildirir.

6.) Merkezi İşlem Birimi kendini sıcaklığa karşı korur.

7.) İşlem sonlanır.

**SICAKLIK ALGILAMA**

**Tanım:** Var olan sıcaklığı algılar ve sıcaklığı bildirir.

**Ad**: Sıcaklık Algılama

**Hazırlayan**: Mustafa KELEŞ

**Sürüm**: M.2.1

**Tarih:** 03/03/2020

**İlgili Aktör:** Merkezi İşlem Birimi

**Giriş Koşulu:** Bir kullanıcı girişi doğrulanması gerekir ve var olan bir sıcaklık vardır bu okunur.

**Çıkış Koşulu:** Sıcaklık okunur ve geriye bildirilir.

**Özel Gereksinim:** Sıcaklık okuma cihazının varlığı ve sıcaklık

**Olay Akışı:**

1.) Kullanıcı giriş doğrulanır.

2.) Merkezi İşlem Birimi talebi alır.

3.) Sıcaklık algılamaya talep bildirilir.

4.) Sıcaklık okuyucu varlığı kontrol edilir.

5.) Sıcaklık okuyucu birimi yapılacak işleme yönlendirilir.

6.) Sıcaklık okuyucu talebi alır ve işler.

7.) Sıcaklık okunur ve geriye bildirilir.

8.) Sıcaklık okuyucu sonucu sıcaklık algılanmaya bildirilir

9.) Sıcaklık algılama Merkezi İşlem Birimi ne durumu bildirir.

**Alternatif Yollar:**

**B1)** Girilen kullanıcıyok ise (1)

2.) Hata mesajı verilir.

3.) İşlem sonlanır.

**B2)** Sıcaklık okuyucu yok ise (4)

5.) Sıcaklık okuyucu yokluğu geri bildirilir.

6.) Merkezi İşlem Birimi okuyucu olmadığını ve sıcaklık okumadığını bildirir.

7.) İşlem sonlanır.

**B3)** Sıcaklık çok yüksek değerde ise veya çok düşük değerde ise (7)

8.) Sıcaklık algılanıyorsa kritik durum bildirilir.

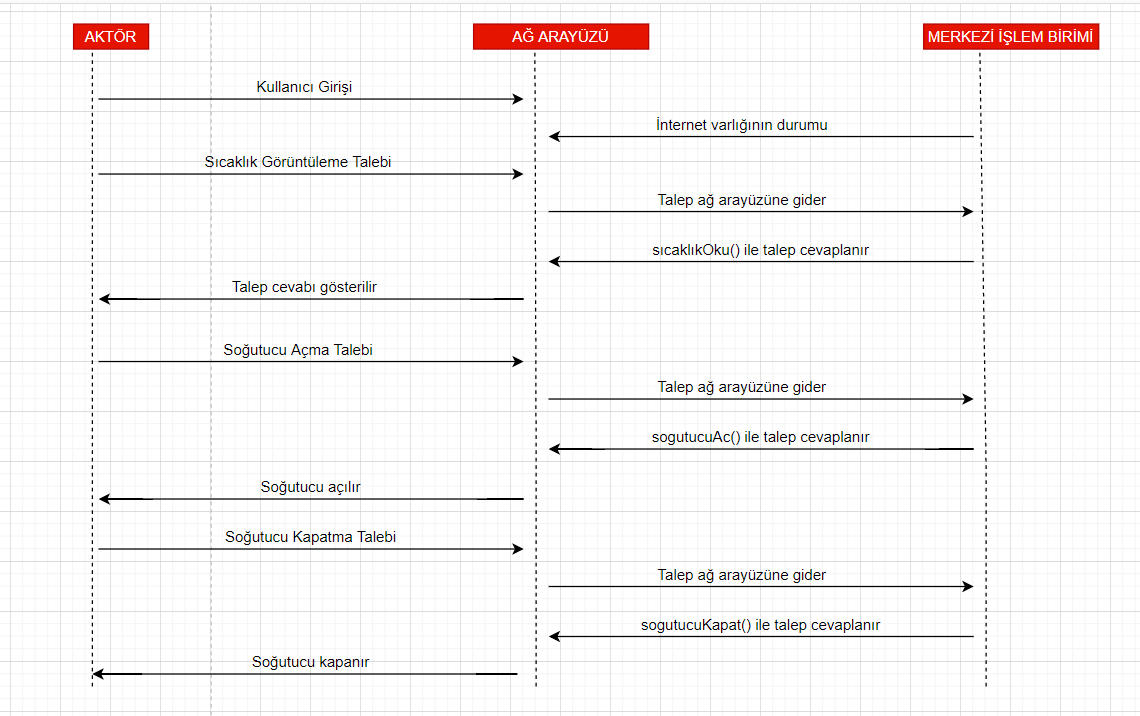
9.) Merkezi İşlem Birimi kritik duruma göre kendini korur.

10.) İşlem sonlanır.

**GÖREV C ve D: İnternet üzerinden “sıcaklığın görüntülenmesi” ve “soğutucunun çalıştırılması” kullanım durumlarına ait; sıralama şemasını (sequence diagram) ve etkinlik şemasını (activity diagram) çiziniz.**

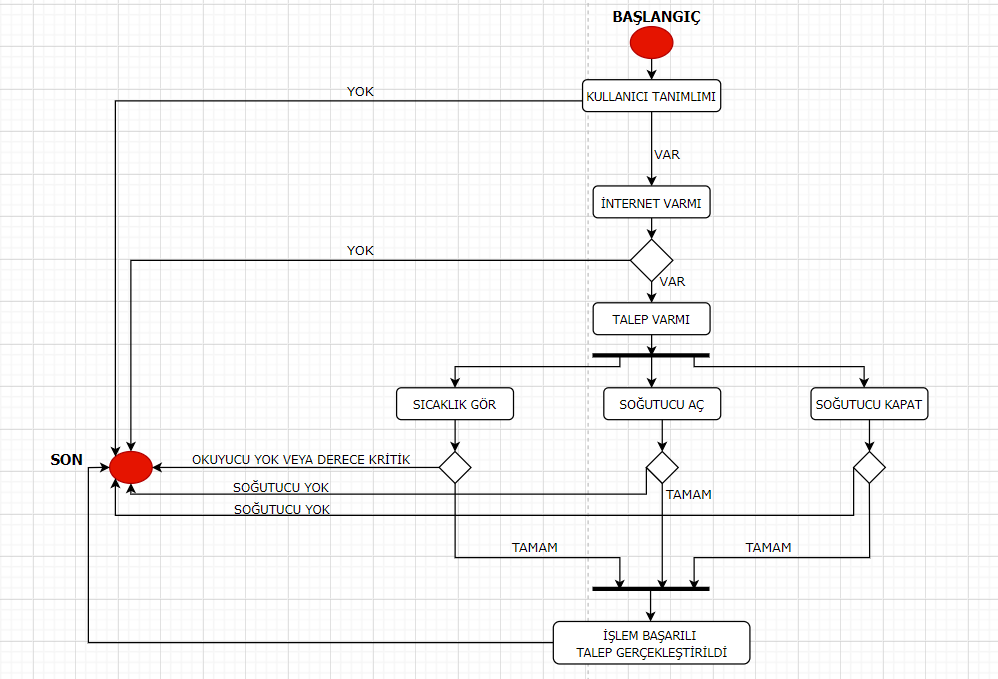
**Sequence Diagram Nedir?**

**Uml** dilinde birçok diyagram bulunmaktadır. Bu diyagramlar kullanış yerlerine ve amaçlarına göre farklılık göstermektedirler. **Sequence Diagram (Sıralama Diyagramı)** daha çok nesnelerin birbirleriyle olan iletişimlerini sırasıyla göstermek için kullanılır.



### Etkinlik (Activity) Diyagramı Nedir?

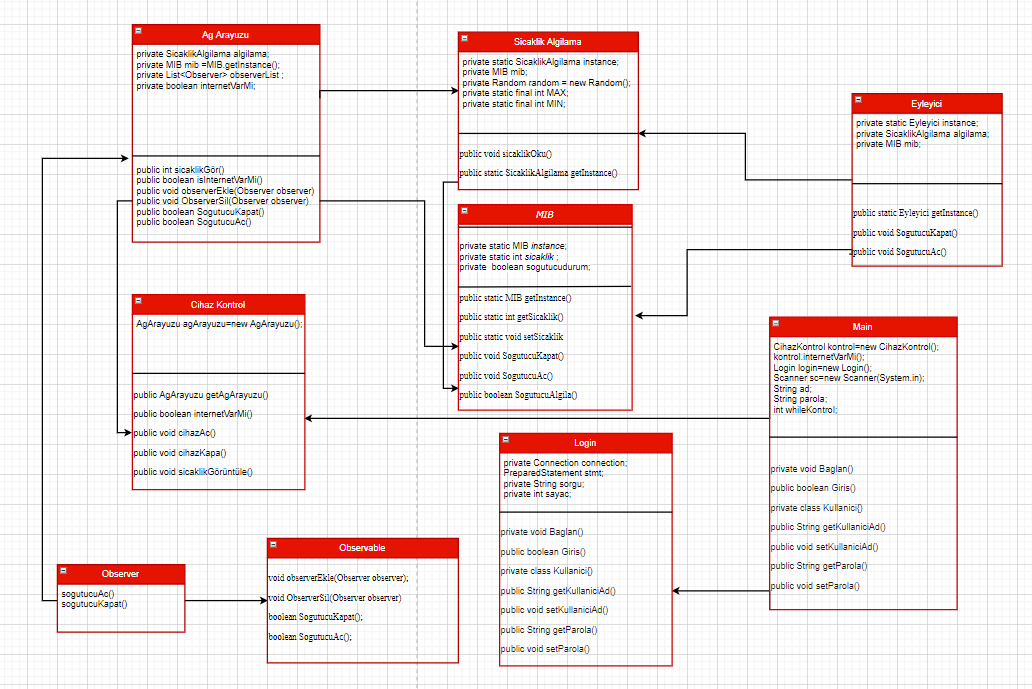
Etkinlik diyagramları, sistemin akış yönünden davranışını betimler. Etkinlik diyagramları bir şeyin durumlarını temsil etmesi niteliği ile durum diyagramlarına benzer. Fakat durum diyagramları gerçekleşen eylemler neticesinde oluşan durumları nitelerken etkinlik diyagramları ise koşullu ya da paralel ilerleyen eylemleri betimler.



**GÖREV E:** Geliştireceğiniz sistemin sınıf şemasını oluşturunuz.

**Sınıf Şeması Nedir?**

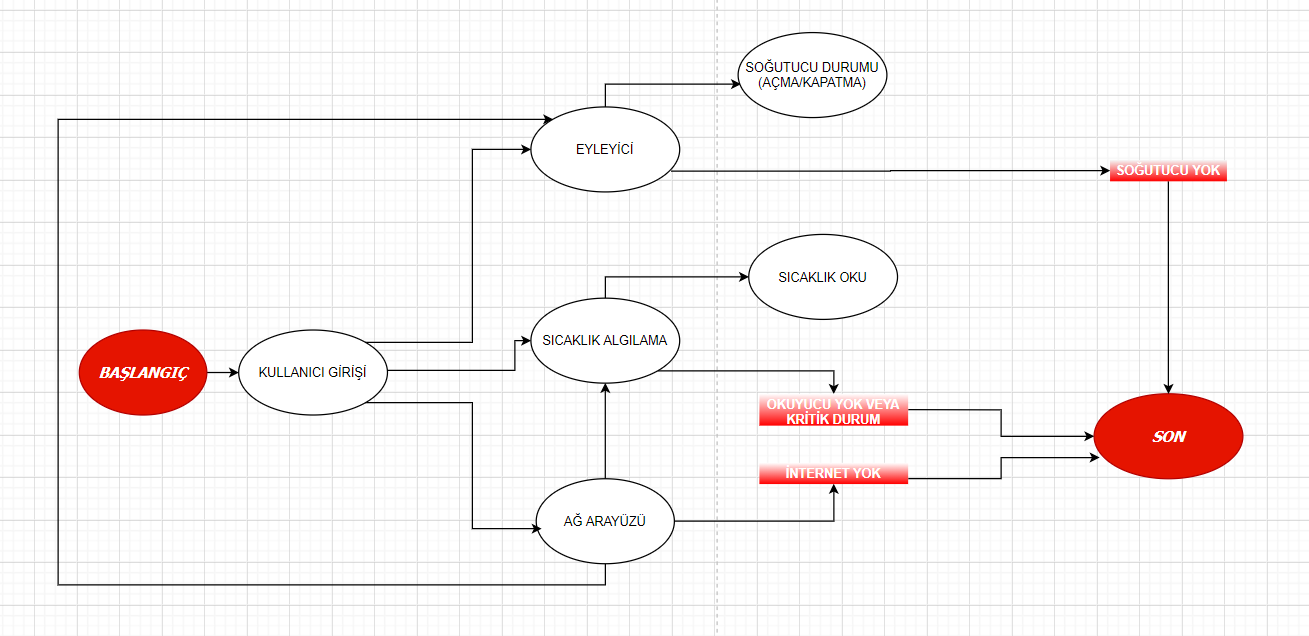
Amaç yazılımımız içindeki sınıflar ve aralarındaki ilişkileri tanımlamaktır.



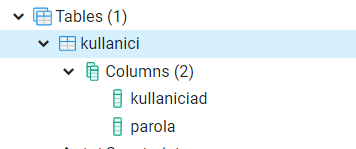
**GÖREV F:** Sistemin durum diyagramını çiziniz.

**Durum Diyagramı Nedir?**

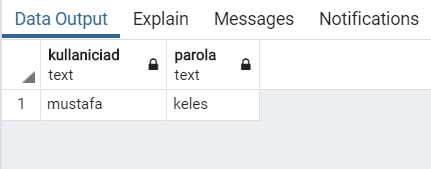
Durum diyagramları bir sistemin davranışlarını modeller ve bir olay gerçekleştiğinde olası tüm durumları tanımlar. Her bir diyagram bir sınıfın tek bir nesnesini ele alır ve sistem içerisindeki farklı durumlarını irdeler.



**GÖREV G:** Kullanıcı doğrulama ekranı ve açıklaması.



Tablo



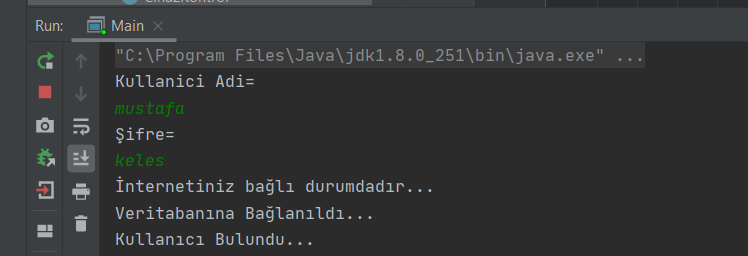
Tablodaki Veriler

Akıllıcihaz sistemine giriş yapılabilmesi için veri tabanındaki kullanici tablomuzda kayıtlı olan;

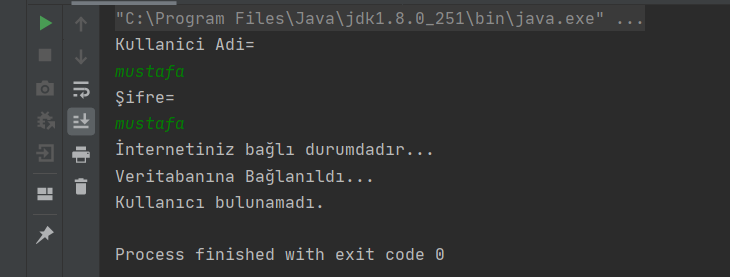
“ Kullanıcı Ad: mustafa

Parola: keles “ giriş bilgileri kullanılmalıdır.

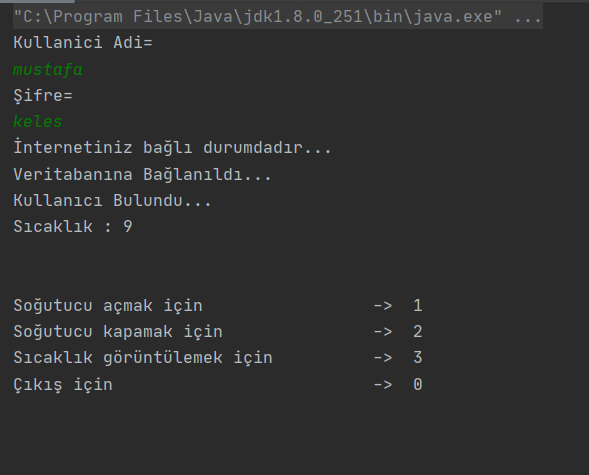
Bu kullanıcı adı ve şifresi doğru bir şekilde girilirse veri tabanı ile başarı bir şekilde bağlantı kurulduktan sonra “Veritabanına Bağlanıldı...” ve kullanıcı bulunduysa “Kullanıcı Bulundu… ”mesajı verilmektedir. Aşağıdaki görselde başarılı bir giriş bulunmaktadır.



Eğer kullanıcı yanlış bir ad veya parola girerse program veritabanına bağlanır, kullanıcıyı arar ve kullanıcı yok ise kullanıcıya “ Kullanıcı bulunamadı.” Hata mesajını vererek programı sonlandırır. Aşağıdaki görselde yanlış şifre girmiş kullanıcının alacağı mesajlar gösterilmiştir.



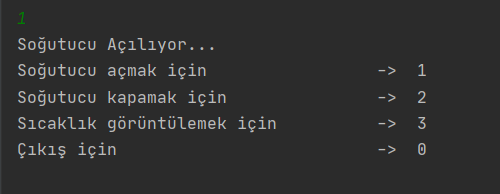
Görev H: Sıcaklığın görüntülenmesi ve soğutucunun açılıp kapatılmasıyla ilgili ekran görüntüleri ve açıklaması.



Yukarıdaki görselde eğer kullanıcı doğru giriş yaptı ise ve interneti bağlı durumda ise kullanıcının önüne gelecek olan menü gözükmektedir. Sıcaklık değeri olarak gözüken değer Random olarak atanmıştır ve girişte gözükmektedir.

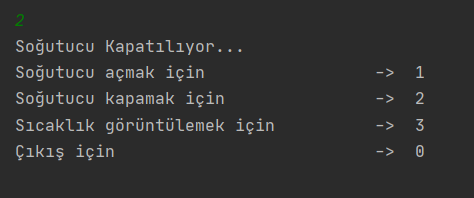
Durum 1:Soğutucu Açma Talebi

Kullanıcı eğer konsol ekranına 1 değerini girerse soğutucu açma işlemini gerçekleştirmektedir. Eğer soğutucu açık ise tekrar açılması engellenmiştir.



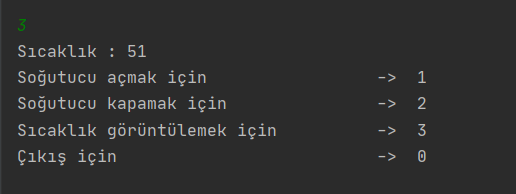
Durum 2:Soğutucu Kapatma Talebi

Kullanıcı eğer konsol ekranına 2 değerini girerse soğutucu kapatma işlemini gerçekleştirmektedir. Eğer soğutucu kapalı ise tekrar kapanması engellenmiştir.



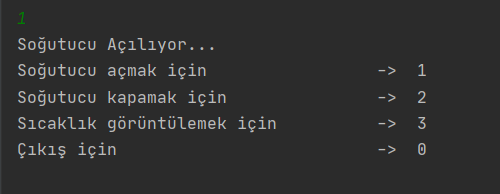
Durum 3: Sıcaklık Görüntüleme Talebi

Kullanıcı eğer konsol ekranına 3 değerini girerse sıcaklık bilgisi görüntüleme işlemini gerçekleştirmektedir. Sıcaklık değeri program tarafından Random olarak atanmıştır.



Durum 0: Çıkış İşlemi

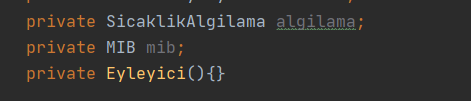
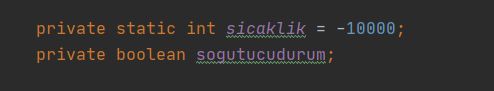
Kullanıcı eğer konsol ekranına 0 değerini girerse program kapanma işlemini gerçekleştirmektedir.



Görev İ: “Open/Closed” ilkesinin ne olduğu ve uygulama içerisinde nasıl gerçeklendiği.

**“Open/Closed” Nedir?**

Bu prensip, değişiklik yapmadan genişleyebilen modüller geliştirmeyi amaçlar. Mevcut kodu değiştirmeden yeni kodlar ekleyerek, yeni özellikler eklemeyi sağlar. Yeni bir gereksinim geldiğinde mevcut kod üzerinde herhangi bir değişiklik yapıyorsanız, open/closed prensibine ters düşüp düşmediğinizi kontrol etmenizde yarar var.



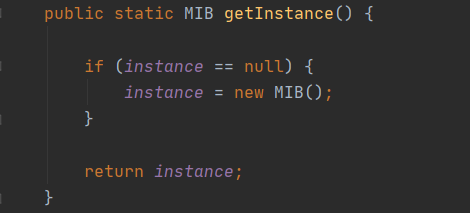
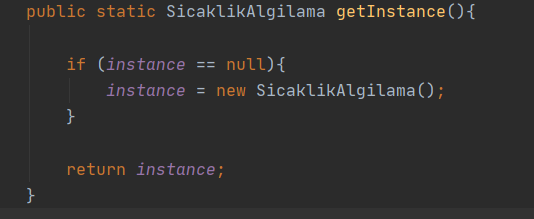
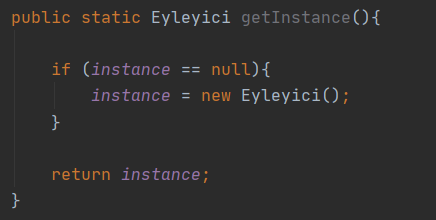
Hazırladığım Akıllı Cihaz uygulamasında yapılan yazılımın tümü open/closed ilkesine göre yazılmıştır. Yani projemizdeki tüm yapılar değişime kapalı fakat gelişime açıktır. Yukarıda birkaç örnek verilmiştir.

**GÖREV J)** “SİNGLETON” ve “OBSERVER” Desenleri Ve Uygulama İçerisinde Nasıl Gerçekleştiği.

**Singleton Nedir?**

Bir sınıfın yalnızca tek nesnesi olmasını ve bu nesneye global olarak erişilmesini sağlamak. Nesne oluşturmayla (creational) ilgili desenlerden biridir. Nesne yoksa oluşturulur döndürülür, varsa olan döndürülür. Nesnenin new komutu ile dışarıdan oluşturulabilmesini engellemek için yapıcı yöntem “private” yapılır. Yapıcı gibi çalışan “static” bir yöntem tanımlanır. Nesne oluşturma görevi bu yöntemindir. Oluşturulan nese “static” bir üyede saklanır.

Gerçeklemesi:



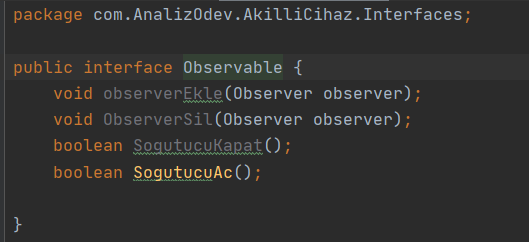
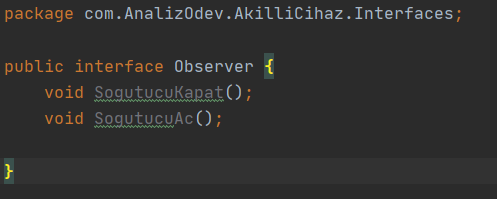
gngngnnvbn

**Observer Nedir?**

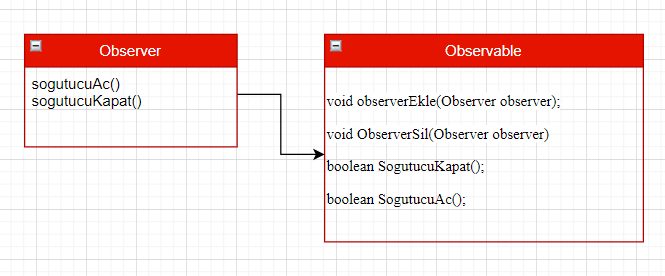
Çok sayıda nesneye, gözlemledikleri nesnede meydana gelen değişikli veya olayı tüm elamanlara bildirmeye yarar.

Gerçeklemesi:

Kod Olarak Görünüm:



Class Diagram Görüntüsü:



Görev K: Uygulamanızın kaynak kodları.

package com.AnalizOdev.AkilliCihaz.Interfaces;  
  
public interface Observable {  
 void observerEkle(Observer observer);  
 void ObserverSil(Observer observer);  
 boolean SogutucuKapat();  
 boolean SogutucuAc();  
  
}

package com.AnalizOdev.AkilliCihaz.Interfaces;  
  
public interface Observer {  
 void SogutucuKapat();  
 void SogutucuAc();  
  
}

package com.AnalizOdev.AkilliCihaz.MerkeziIslemBirimi;  
  
import com.AnalizOdev.AkilliCihaz.Interfaces.Observable;  
import com.AnalizOdev.AkilliCihaz.Interfaces.Observer;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class MIB {  
 private static MIB *instance*;  
  
 private static int *sicaklik* = -10000;  
 private boolean sogutucudurum;  
  
 private MIB() {  
 }  
  
 public static MIB getInstance() {  
  
 if (*instance* == null) {  
 *instance* = new MIB();  
 }  
  
 return *instance*;  
 }  
  
 public static int getSicaklik() {  
 return *sicaklik*;  
 }  
  
 public static void setSicaklik(int sicaklik) {  
 MIB.*sicaklik* = sicaklik;  
 }  
  
 public void SogutucuKapat() {  
 this.sogutucudurum = false;  
 }  
  
 public void SogutucuAc() {  
 this.sogutucudurum = true;  
 }  
  
 public boolean SogutucuAlgila() {  
 return this.sogutucudurum;  
 }  
}

package com.AnalizOdev.AkilliCihaz;  
  
import com.AnalizOdev.AkilliCihaz.Interfaces.Observable;  
import com.AnalizOdev.AkilliCihaz.Interfaces.Observer;  
import com.AnalizOdev.AkilliCihaz.MerkeziIslemBirimi.MIB;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class AgArayuzu implements Observable {  
 private SicaklikAlgilama algilama=SicaklikAlgilama.*getInstance*();  
 private MIB mib =MIB.*getInstance*();  
 private List<Observer> observerList = new ArrayList<>();  
 private boolean internetVarMi;  
 public int sicaklikGör(){  
 algilama.sicaklikOku();  
 if(mib.*getSicaklik*()!=-10000){  
 return mib.*getSicaklik*();  
 }else{  
 System.*out*.println("Sıcaklık Algılanamadı. Yeniden Deneyiniz.");  
 return mib.*getSicaklik*();  
 }  
 }  
  
 public boolean isInternetVarMi() {  
 return internetVarMi;  
 }  
  
 public void setInternetVarMi(boolean internetVarMi) {  
 this.internetVarMi = internetVarMi;  
 }  
  
 @Override  
 public void observerEkle(Observer observer) {  
 observerList.add(observer);  
 }  
  
 @Override  
 public void ObserverSil(Observer observer) {  
 observerList.remove(observer);  
 }  
 @Override  
 public boolean SogutucuKapat() {  
 if(mib.SogutucuAlgila()==true){  
 mib.SogutucuKapat();  
 return true;}  
 else{ return false;}  
 }  
  
 @Override  
 public boolean SogutucuAc() {  
 if(mib.SogutucuAlgila()==false){  
 mib.SogutucuAc();  
 return true;  
 }else{  
 return true;  
 }  
 }  
}

package com.AnalizOdev.AkilliCihaz;  
  
import com.AnalizOdev.AkilliCihaz.Interfaces.Observer;  
import com.AnalizOdev.AkilliCihaz.MerkeziIslemBirimi.MIB;  
  
public class Eyleyici implements Observer {  
 private static Eyleyici *instance*;  
 private SicaklikAlgilama algilama;  
 private MIB mib;  
 private Eyleyici(){}  
  
 public static Eyleyici getInstance(){  
  
 if (*instance* == null){  
 *instance* = new Eyleyici();  
 }  
  
 return *instance*;  
 }  
 @Override  
 public void SogutucuKapat() {  
 System.*out*.println("Soğutucu Kapatılıyor...");  
 mib.SogutucuKapat();  
 }  
  
 @Override  
 public void SogutucuAc() {  
 System.*out*.println("Soğutucu Açılıyor...");  
 mib.SogutucuAc();  
 }  
}

package com.AnalizOdev.AkilliCihaz;  
  
import com.AnalizOdev.AkilliCihaz.MerkeziIslemBirimi.MIB;  
  
import java.util.Random;  
  
public class SicaklikAlgilama {  
 private static SicaklikAlgilama *instance*;  
 private MIB mib;  
 private Random random = new Random();  
 private static final int *MAX*=100;  
 private static final int *MIN*=-20;  
  
  
 public void sicaklikOku() {  
 int derece = random.nextInt(*MAX*-*MIN*)+*MIN*;  
 mib.*setSicaklik*(derece);  
 }  
  
 public static SicaklikAlgilama getInstance(){  
  
 if (*instance* == null){  
 *instance* = new SicaklikAlgilama();  
 }  
  
 return *instance*;  
 }  
}

package com.AnalizOdev;  
  
import com.AnalizOdev.AkilliCihaz.AgArayuzu;  
  
import java.io.IOException;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.net.URLConnection;  
  
public class CihazKontrol {  
 AgArayuzu agArayuzu=new AgArayuzu();  
  
 public AgArayuzu getAgArayuzu() {  
 return agArayuzu;  
 }  
  
 public boolean internetVarMi() {  
 try {  
 URL url = new URL("http://www.google.com");  
 URLConnection connection = url.openConnection();  
 connection.connect();  
 agArayuzu.setInternetVarMi(true);  
 return true;  
 } catch (MalformedURLException e) {  
 return false;  
 } catch (IOException e) {  
 return false;  
 }  
 }  
  
 public void cihazAc(){  
 if(agArayuzu.SogutucuAc()){  
 System.*out*.println("Soğutucu Açılıyor...");  
 }else{  
 System.*out*.println("Soğutucu Zaten Açık...");  
 }  
 }  
 public void cihazKapa(){  
 if(agArayuzu.SogutucuAc()){  
 System.*out*.println("Soğutucu Kapatılıyor...");  
 }else{  
 System.*out*.println("Soğutucu Zaten Kapalı...");  
 }  
 }  
 public void sicaklikGörüntüle(){  
 System.*out*.println("Sıcaklık : "+agArayuzu.sicaklikGör());  
 }  
   
}

package com.AnalizOdev;  
  
import java.sql.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class Login {  
 private Connection connection=null;  
 PreparedStatement stmt= null;  
 private String sorgu="Select \* from kullanici where kullaniciad=? and parola=?";  
 private int sayac=0;  
 private void Baglan(){  
 try {  
 Class.*forName*("org.postgresql.Driver");  
 connection= DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/analız\_odev","postgres", "Mustafa");  
 }catch (Exception e){  
 System.*out*.println("Veritabanına Bağlanılamadı...");  
 sayac=1;  
 }  
 if (sayac==0) {  
 System.*out*.println("Veritabanına Bağlanıldı...");  
 }  
 }  
 public boolean Giris(String ad,String parola){  
 this.Baglan();  
 ResultSet rs=null;  
 List<Kullanici> kullanicilar=new ArrayList<>();  
 try {  
 stmt= connection.prepareStatement(sorgu);  
 stmt.setString(1,ad);  
 stmt.setString(2,parola);  
 rs =stmt.executeQuery();  
 while (rs.next()){  
 kullanicilar.add(new Kullanici(  
 rs.getString("kullaniciad"),  
 rs.getString("parola")  
 ));  
 }  
 rs.close();  
 stmt.close();  
 connection.close();  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
  
 }finally {  
 try { rs.close(); } catch (Exception e) { /\* ignored \*/ }  
 try { stmt.close(); } catch (Exception e) { /\* ignored \*/ }  
 try { connection.close(); } catch (Exception e) { /\* ignored \*/ }  
 }  
 if(kullanicilar.isEmpty() || kullanicilar.size()>1){  
 return false;  
 }else {  
 return true;  
 }  
  
 }  
  
 private class Kullanici{  
 private String kullaniciAd;  
 private String parola;  
  
 public Kullanici() {  
 }  
  
 public Kullanici(String kullaniciAd, String parola) {  
 this.kullaniciAd = kullaniciAd;  
 this.parola = parola;  
 }  
  
 public String getKullaniciAd() {  
 return kullaniciAd;  
 }  
  
 public void setKullaniciAd(String kullaniciAd) {  
 this.kullaniciAd = kullaniciAd;  
 }  
  
 public String getParola() {  
 return parola;  
 }  
  
 public void setParola(String parola) {  
 this.parola = parola;  
 }  
  
 }  
}

package com.AnalizOdev;  
  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 CihazKontrol kontrol=new CihazKontrol();  
 kontrol.internetVarMi();  
 Login login=new Login();  
 Scanner sc=new Scanner(System.*in*);  
 String ad;  
 String parola;  
 int whileKontrol=-1;  
 if(kontrol.getAgArayuzu().isInternetVarMi()) {  
 System.*out*.println("Kullanici Adi=");  
 ad=sc.nextLine();  
 System.*out*.println("Şifre=");  
 parola=sc.nextLine();  
 System.*out*.println("İnternetiniz bağlı durumdadır...");  
 if (login.Giris(ad, parola)) {  
 System.*out*.println("Kullanıcı Bulundu...");  
 kontrol.sicaklikGörüntüle();  
 System.*out*.println("\n");  
 while (whileKontrol != 0) {  
 System.*out*.println("Soğutucu açmak için \t->\t1\n" +  
 "Soğutucu kapamak için \t->\t2\n" +  
 "Sıcaklık görüntülemek için \t->\t3\n" +  
 "Çıkış için \t->\t0\n"  
 );  
 whileKontrol = sc.nextInt();  
 switch (whileKontrol) {  
 case 1:  
 kontrol.cihazAc();  
 break;  
 case 2:  
 kontrol.cihazKapa();  
 break;  
 case 3:  
 kontrol.sicaklikGörüntüle();  
 break;  
 }  
 }  
 }  
 else {  
 System.*out*.println("Kullanıcı bulunamadı.");  
 }  
 }  
 else {  
 System.*out*.println("İNTERNET BAĞLI DEĞİL , LÜTFEN İNTERNETİNİZİ KONTROL EDİNİZ.");  
 }  
 }  
}

**VİDEO LİNKİ:**

<https://www.youtube.com/watch?v=5gvSBc9ARaQ>