

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

NESNE YÖNELİMLİ ANALİZ VE TASARIM DERSİ
YIL İÇİ PROJE ÇALIŞMASI RAPORU

SOĞUTUCU DENETİMİ İÇİN

AKILLI CİHAZ TASARIMI

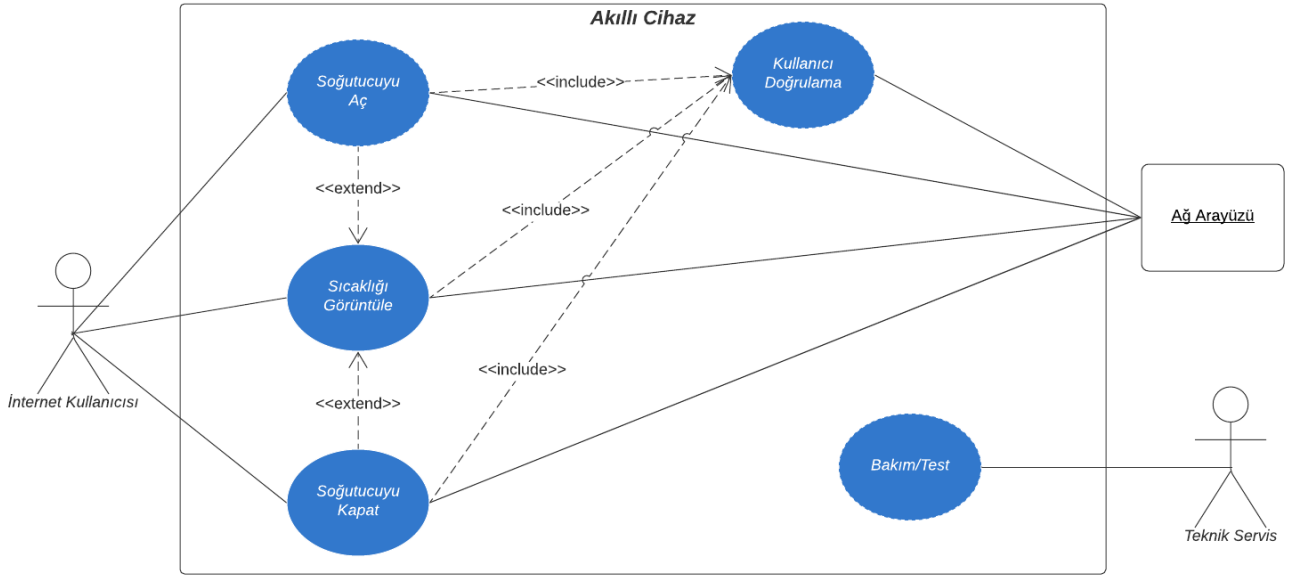
İlker KÜÇÜCÜK

B191210351

ilker.kucucuk@ogr.sakarya.edu.tr

1.Öğretim A Grubu

İnternet Kullanıcısı İçin Kullanıcı Durum Diyagramı



Kullanım Durumları

a) Akıllı Cihaz ile Sıcaklığın Görüntülenmesi Kullanım Durumu

Hazırlayan: İlker Küçücük

Sürüm: 1.0

Tarih: 04/17/2021

İlgili Aktörler: İnternet Kullanıcısı

Use Case - “Sıcaklığın Görüntülenmesi” Olay Akışı (Ana Senaryo Başarılı)

1. Cihaz çalıştırılır ve arayüze bağlanılır. Arayüzün tasarım ve eklentileri yüklenir. Kullanıcı adı ve şifre istenir
2. Kullanıcı adı ve şifre girdisi yapılır
3. Kimlik, veritabanı tarafında kontrol edilir.
4. Sisteme giriş sağlanır. Kontrol paneli yüklenir.
5. Kullanıcıya yapılabilecek işlemleri sunar
6. Sıcaklığı görüntülenebilmesi için sıcaklık görüntüle seçeneği seçilir ve işlem birimine istek gönderilir.
7. Sıcaklık sensörü ölçtüğü değeri işlem birimine iletir.
8. Değerler arayüzde kullanıcıya yazdırılır.

Use Case - “Sıcaklığın Görüntülenmesi” Olay Akışı (Alternatif Akışlar)

A1(3) – Kimlik Bilgileri Hatalı!

- Kullanıcı adı ve şifre veritabanında herhangi bir kayıt ile eşleşmezse arayüze uyarı mesajı gönderilir ve tekrar giriş denemesi yapılması istenir.

A2(6) – Oturum Zaman Aşımına Uğradı!

- Kullanıcı panelde yapmasını istediği işlemi seçmezse ve bir süre böyle beklerse zaman aşımına uğradığına dair bilgi alır ve oturumu sonlandırılır. Kullanıcının devam edebilmesi için tekrar giriş yapması gerekir.

Özel Gereksinimler: Arayüz gereksinimleri, sıcaklık değeri bağlantı aktif olduğu sürece gösterilmeli ve anlık olarak getirilmelidir.

b) Akıllı Cihaz ile Soğutucu Kontrolü Kullanım Durumu

Hazırlayan: İlker Küçücük

Sürüm: 1.0

Tarih: 04/17/2021

İlgili Aktörler: Akıllı Cihaz ve Kullanıcısı

Use Case - “Soğutucunun Çalıştırılması” Olay Akışı (Ana Senaryo Başarılı)

1. Cihaz çalıştırılır ve arayüze bağlanır. Arayüzün tasarım ve eklentileri yüklenir. Kullanıcı adı ve şifre istenir.
2. Kullanıcı adı ve şifre girdisi yapılır.
3. Kimlik, veritabanı tarafında kontrol edilir.
4. Sisteme giriş sağlanır. Kontrol paneli yüklenir.
5. Kullanıcıya yapılabilecek işlemleri sunar.
6. Kullanıcı soğutucunun çalıştırılabilmesi için soğutucuyu çalıştır seçeneğini seçer ve arayüz, isteği işlem birimine iletir.
7. İşlem birimi isteği eyleyiciye iletir ve soğutucu çalıştırılır. Kullanıcı soğutucunun çalıştığına dair bilgilendirme mesajı alır.

Use Case - “Soğutucunun Çalıştırılması” Olay Akışı (Alternatif Akışlar)

A1(3) – Kimlik Bilgileri Hatalı!

- Kullanıcı adı ve şifre veritabanında herhangi bir kayıt ile eşleşmezse arayüze uyarı mesajı gönderilir ve tekrar giriş denemesi yapması istenir.

A2(6) – Oturum Zaman Aşımına Uğradı!

- Kullanıcı panelde yapmasını istediği işlemi seçmezse ve bir süre böyle beklerse zaman aşımına uğradığına dair bilgi alır ve oturumu sonlandırılır. Kullanıcının devam edebilmesi için tekrar giriş yapması gerekir.

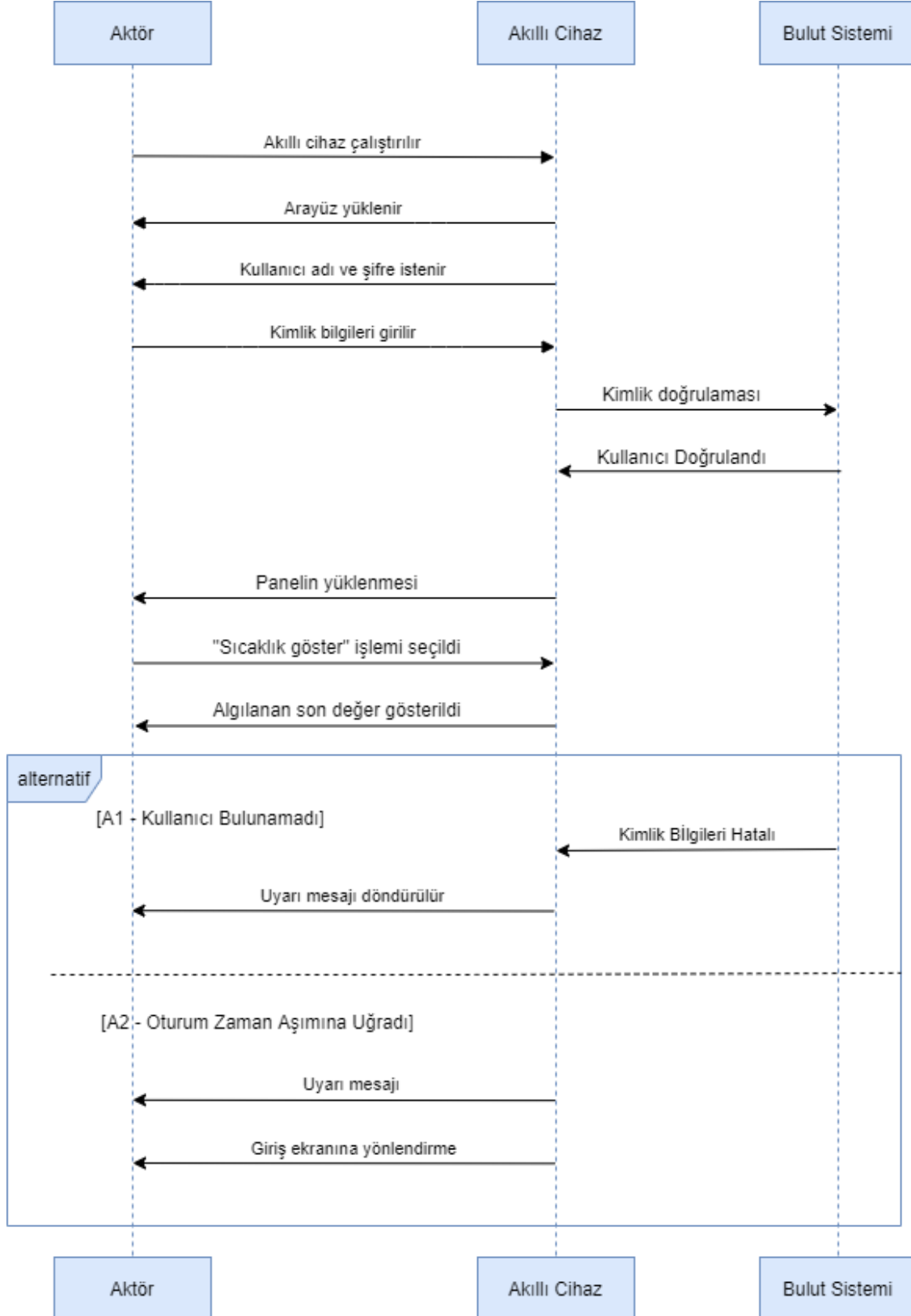
A3(7) – Soğutucu Zaten Açık!

- Soğutucu açıldıktan sonra arayüzdeki soğutucuyu açma seçeneği kullanılmaz duruma gelir. Soğutucu açıkken sadece kapatma işlemi gerçekleştirilebilir. Kullanıcı tekrar soğutucu açmaya çalıştığında uyarı mesajı alır.

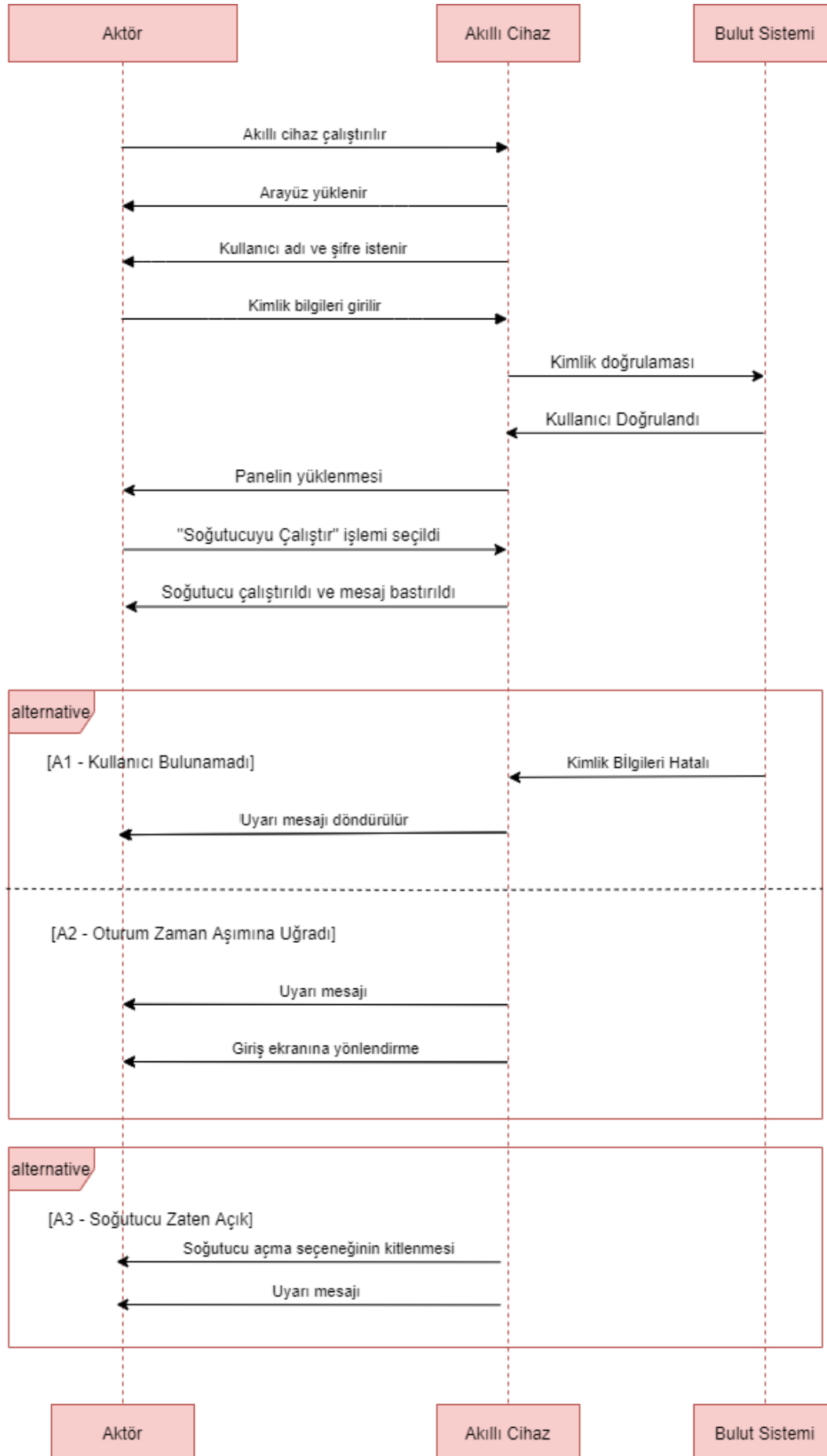
Özel Gereksinimler: Arayüz gereksinimleri, kullanıcının aktif seçenekleri tekrar kullanması engellenmelidir.

Sıralama Şemaları

a) Sıcaklığın Görüntülenmesi Kullanım Durumuna Ait Sıralama Şeması(Sequence Diagram)

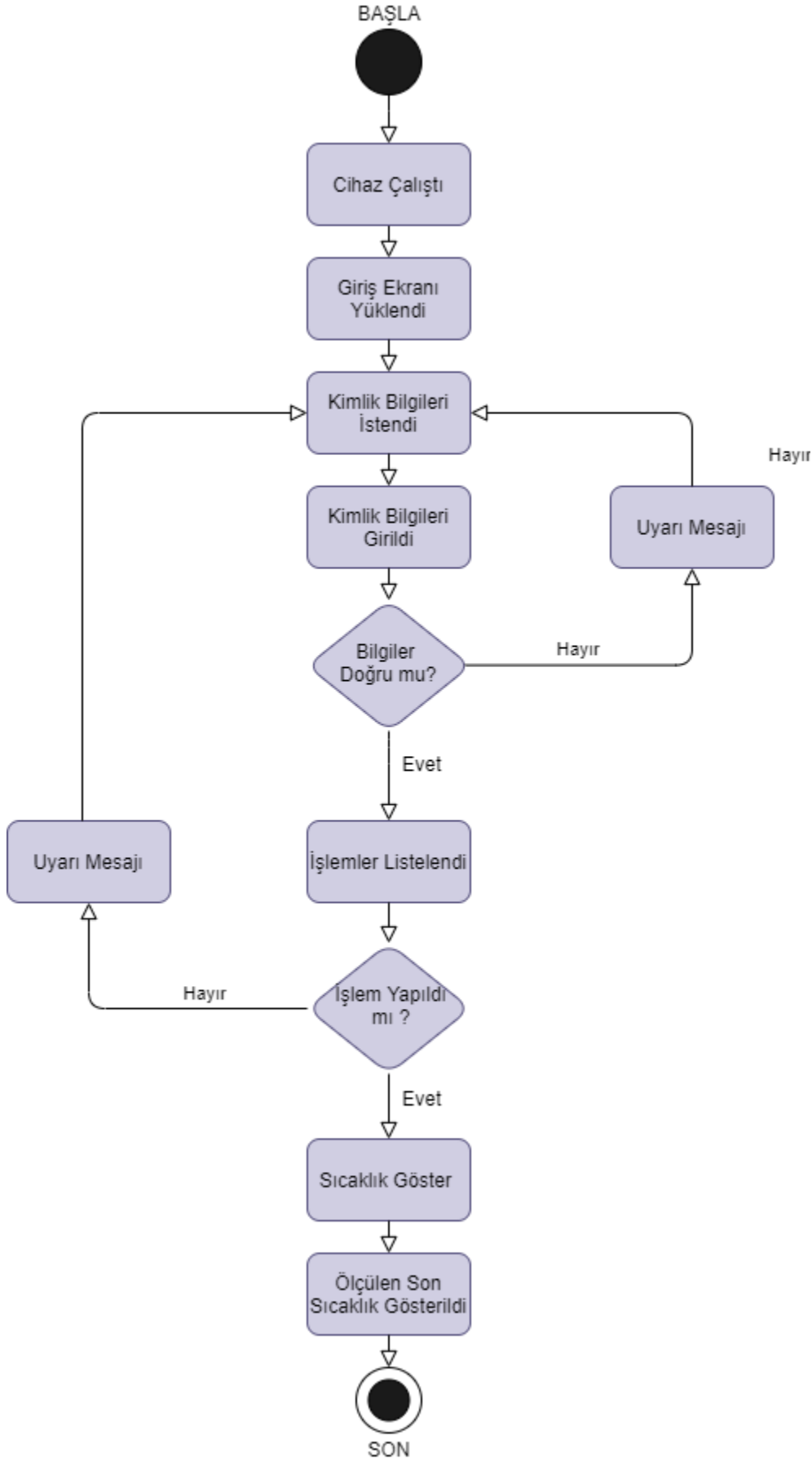


b) Soğutucun Çalıştırılması Kullanım Durumuna Ait Sıralama Şeması(Sequence Diagram)

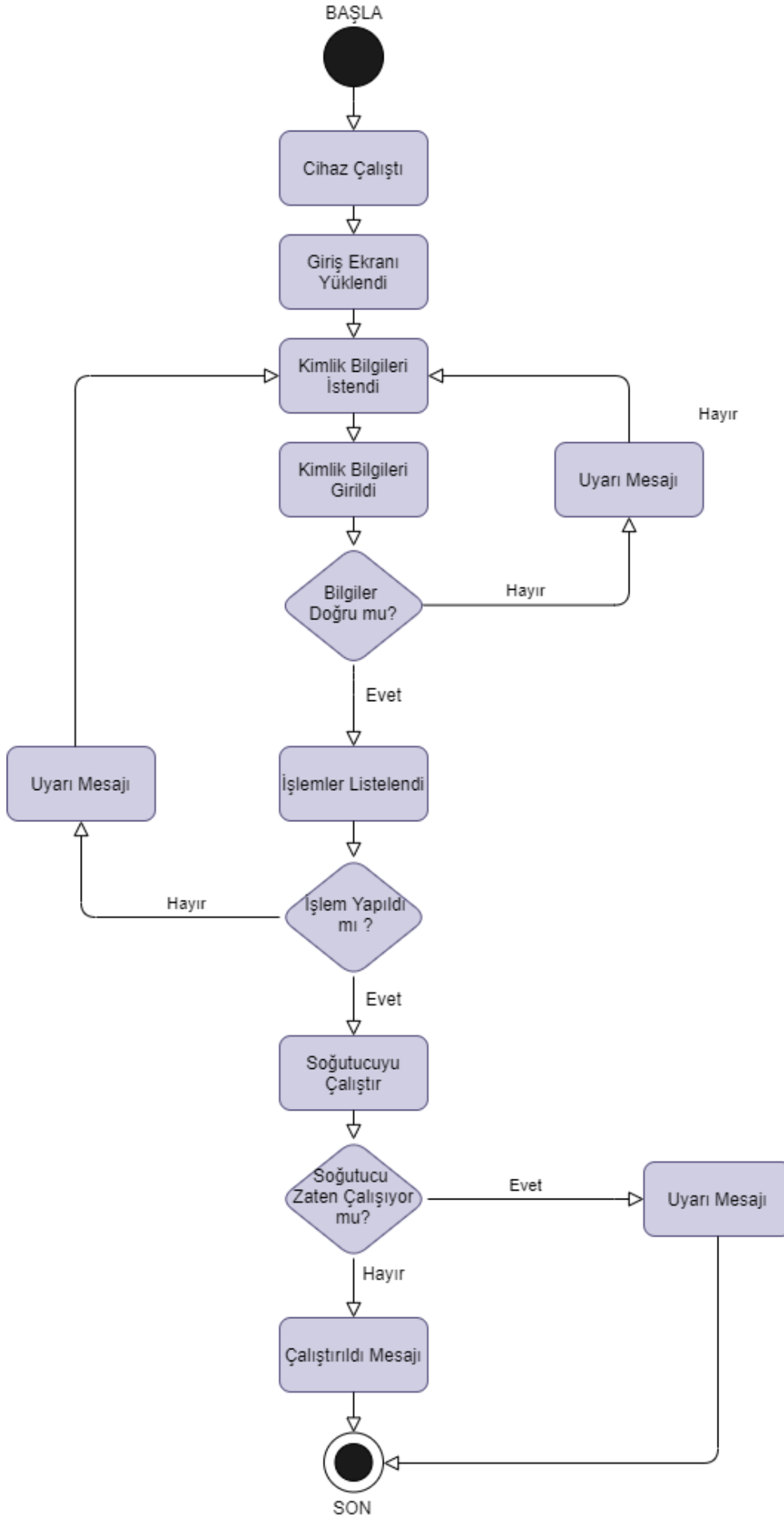


Etkinlik Şemaları

a) Sıcaklığın Görüntülenmesi Etkinlik Şeması (Activitiy Diagram)

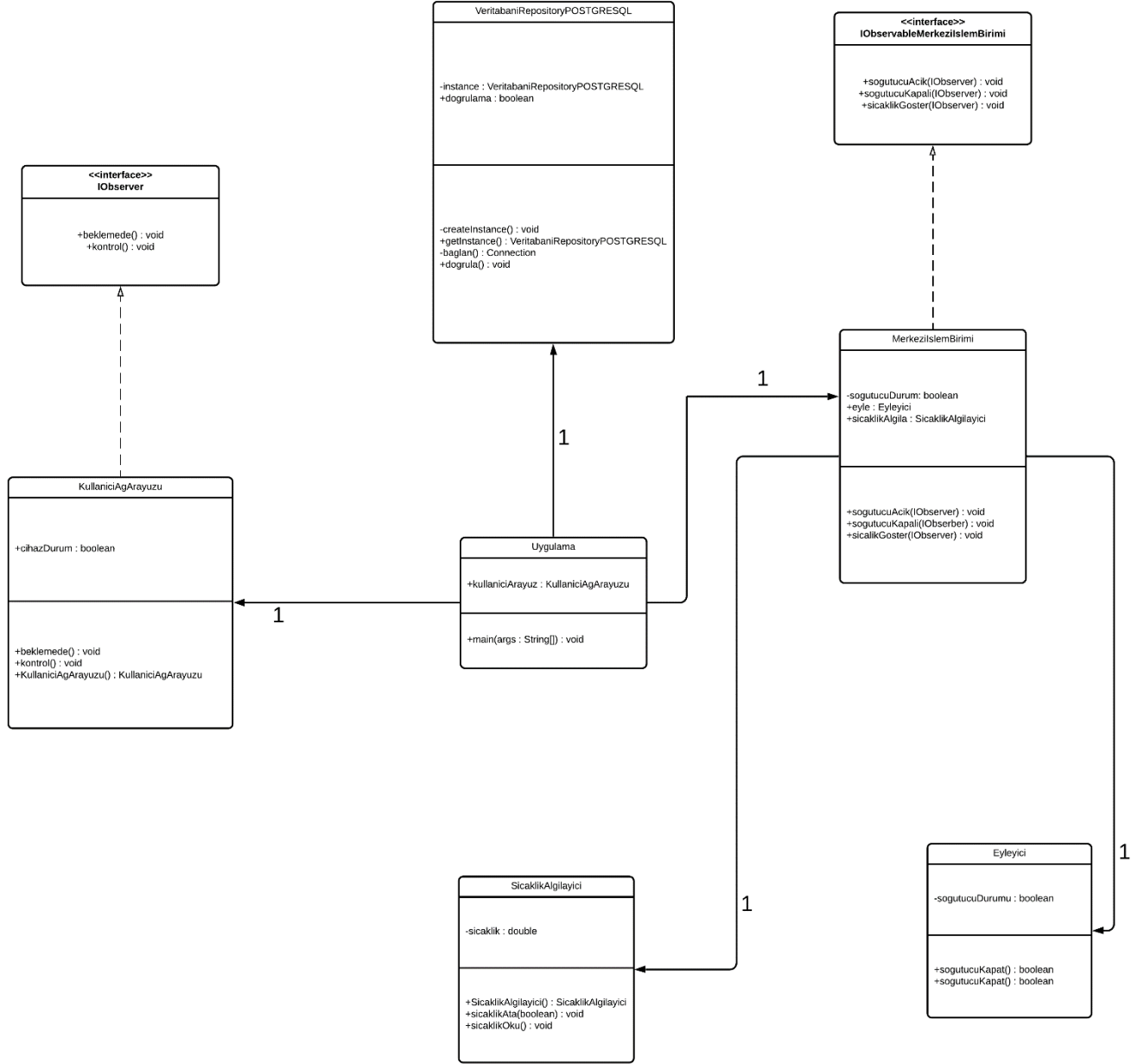


b) Soğutucun Çalıştırılması Etkinlik Şeması (Activity Diagram)



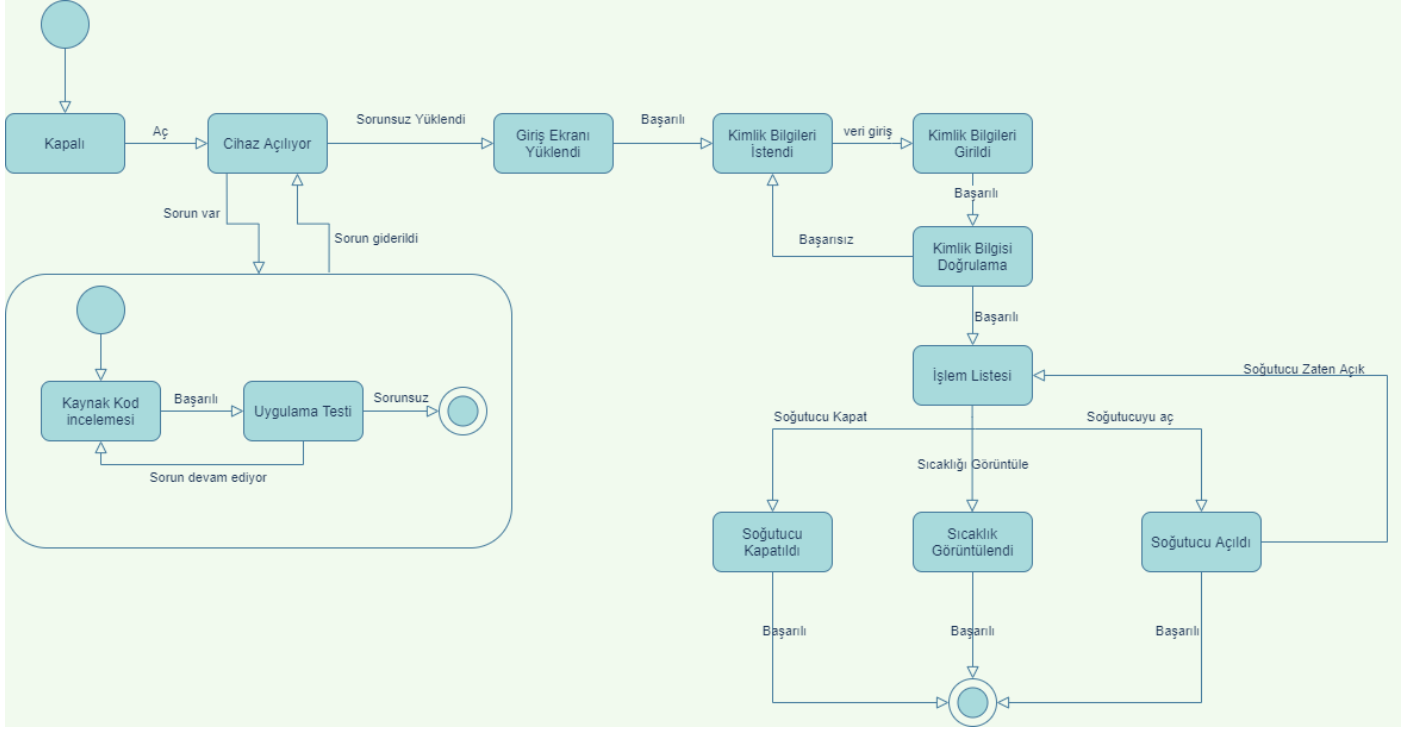
Sınıf Şeması

Sistemin Sınıf Şeması



Durum Diyagramı

Sistemin Sınıf Şeması



Kod İçeriği

Akıllı Cihazın Çalıştırılması ve Kullanıcı Girişi

Program çalıştırıldığında "Akıllı cihaz çalıştırıldı" yazısı ağ arayüzünde gösterilir. Veritabanı bağlantısı gerçekleştirilir ve daha sonrasında kullanıcıdan kullanıcı adı ve şifresinin girilmesi istenir.

```
Akıllı cihaz çalıştırıldı
```

```
-----  
Veritabanına bağlandı!
```

```
Kullanıcı adı: ilkerkek
```

```
Şifrenizi gir:
```

Eğer Kullanıcı Girişi Doğrulırsa

Eğer girilen kullanıcı adı ve şifre değerleri veritabanında "kullanici" tablosundaki bir kullanıcı kaydı ile eşleşirse doğrulama başarılı olur. Ağ arayüzünde "Giriş Başarılı!" mesajı kullanıcıya gösterilir ve hemen ardından kullanıcının yapabileceği işlemler listelenir.

```
Veritabanına bağlandı!
```

```
Kullanıcı adı: ilkerkek
```

```
Şifrenizi gir: 159753
```

```
Giriş başarılı!
```

```
Hoş geldin ilker
```

```
Lütfen İşlem seçiniz
```

```
1-Soğutucu Aç
```

```
2-Soğutucu Kapat
```

```
3-Sıcaklık Görüntüle
```

```
4-Çıkış Yap
```

```
İşleminiz: |
```

Eğer Girilen Bilgiler Doğrulanamazsa

Eğer girilen kullanıcı adı ve şifre değerleri veritabanında "kullanici" tablosundaki bir kullanıcı kaydı ile eşleşmez ise doğrulama başarısız olur ve "Kullanıcı adı ve şifre kombinasyonu yanlış! Kullanıcı bilgilerinizi lütfen tekrar giriniz." mesajı kullanıcıya gösterilir. Kullanıcı adı ve kullanıcı şifresi bilgileri kullanıcıdan tekrar istenir.

```
public static void KullaniciGiris() throws SQLException {
    Connection dogrulamaBaglantisi = baglan();
    String vtKullaniciAdi = null, vtSifre = null, vtIsim;
    Statement stmt = dogrulamaBaglantisi.createStatement();
    String sorgu = "SELECT * FROM \"kullanici\"";
    while (dogrulama == false) {
        ResultSet rs = stmt.executeQuery(sorgu);
        Scanner veriCek = new Scanner(System.in);
        System.out.print("Kullanıcı adı: ");
        String kAdi = veriCek.next();
        System.out.print("Şifrenizi gir: ");
        String sifre = veriCek.next();
        System.out.println("\n");
        // Kullanıcı adı ve şifre kontrolü
        while (rs.next()) {
            vtKullaniciAdi = rs.getString( columnLabel: "kullaniciadi");
            vtSifre = rs.getString( columnLabel: "kullanicisifre");
            vtIsim = rs.getString( columnLabel: "isim");
            if (kAdi.equals(vtKullaniciAdi) && sifre.equals(vtSifre)) {
                System.out.println(String.format("Giriş başarılı!\nHoş geldin %s\n", vtIsim));
                dogrulama = true;
                break;
            } else {
                System.out.println("Kullanıcı adı ve şifre kombinasyonu yanlış!\nKullanıcı bilgilerinizi lütfen tekrar giriniz.\n");
                dogrulama = false;
                break;
            }
        }
    }
    dogrulamaBaglantisi.close();
}
```

Sıcaklığın Görüntülenmesi ve Soğutucunun Açıp Kapatılması

Giriş işlemi başarıyla gerçekleştirdikten sonra kullanıcının karşısına ağ arayüzü aracılığıyla yapabileceği işlemler listelenir.

Soğutucunun Açılması

-Eğer Kullanıcı 1 Numaralı İşlemi Seçer İse Soğutucu Açılır

Cihaz durumu kontrol durumuna geçer soğutucu açılır ve arayüze "Soğutucu Açıldı!" mesajı gönderilir. Bu esnada ortamın sıcaklığı düşmeye başlar. Daha sonra cihaz tekrar bekleme moduna döner ve tekrar işlem menüsü kullanıcıya gösterilir.

```
Cihaz kontrol ediliyor...
```

```
-----
```

```
*** Soğutucu Açıldı ! ***
```

```
Cihaz işlem bekliyor...
```

```
-----
```

```
Lütfen İşlem seçiniz
```

```
1-Soğutucu Aç
```

```
2-Soğutucu Kapat
```

```
3-Sıcaklık Görüntüle
```

```
4-Çıkış Yap
```

```
İşleminiz: |
```

-Eğer Kullanıcı Soğutucu Açık olduğu halde 1 Numaralı İşlemi Seçer İse

Cihaz durumu tekrar kontrol durumuna geçer ve soğutucu açık mı diye kontrol edilir az önceki işlemde soğutucu açıldığı ve tekrar açılmaya çalışıldığı için arayüze "Soğutucu zaten açık tekrar açamazsınız!" mesajını gönderir. Daha sonra tekrar cihaz tekrar bekleme durumuna geçer ve kullanıcıya işlem menüsü tekrar gösterilir.

```
Cihaz kontrol ediliyor...
```

```
-----
```

```
*** Soğutucu zaten açık ! ***
```

```
Cihaz işlem bekliyor...
```

```
-----
```

```
Lütfen İşlem seçiniz
```

```
1-Soğutucu Aç
```

```
2-Soğutucu Kapat
```

```
3-Sıcaklık Görüntüle
```

```
4-Çıkış Yap
```

```
İşleminiz: |
```

Soğutucunun Kapatılması

-Eğer Kullanıcı 2 Numaralı İşlemi Seçer İse Soğutucu Açılır

Cihaz durumu kontrol durumuna geçer soğutucu kapatılır ve arayüze "Soğutucu Kapatıldı!" mesajı gönderilir. Bu esnada ortamın sıcaklığı artmaya başlar. Daha sonra cihaz tekrar bekleme moduna döner ve tekrar işlem menüsü kullanıcıya gösterilir. Bu işlemin gerçekleştirilebilmesi için öncelikle soğutucu açık durumda olmalıdır.

```
Cihaz kontrol ediliyor...
```

```
-----
```

```
*** Soğutucu Kapalı! ***
```

```
Cihaz işlem bekliyor...
```

```
-----
```

```
Lütfen İşlem seçiniz
```

```
1-Soğutucu Aç
```

```
2-Soğutucu Kapat
```

```
3-Sıcaklık Görüntüle
```

```
4-Çıkış Yap
```

```
İşleminiz: |
```

-Eğer Kullanıcı Soğutucu Kapalı olduğu halde 2 Numaralı İşlemi Seçer İse

Cihaz durumu tekrar kontrol durumuna geçer ve soğutucu kapalı mı diye kontrol edilir az önceki işlemde soğutucu kapandığı ve tekrar kapatılmaya çalışıldığı için arayüze "Soğutucu zaten kapalı tekrar kapatamazsınız!" mesajını gönderir. Daha sonra tekrar cihaz tekrar bekleme durumuna geçer ve kullanıcıya işlem menüsü tekrar gösterilir.

```
Cihaz kontrol ediliyor...
```

```
-----
```

```
*** Soğutucu zaten kapalı !***
```

```
Cihaz işlem bekliyor...
```

```
-----
```

```
Lütfen İşlem seçiniz
```

```
1-Soğutucu Aç
```

```
2-Soğutucu Kapat
```

```
3-Sıcaklık Görüntüle
```

```
4-Çıkış Yap
```

```
İşleminiz:
```

Sıcaklığın Görüntülenmesi

-Eğer Kullanıcı 3 Numaralı İşlemi Seçer İse Rastgele Bir Oda Sıcaklığı Görüntülenir:

Cihaz durumu kontrol durumuna geçer sıcaklık değeri arayüzde gösterilir. Daha sonra cihaz tekrar bekleme moduna döner ve tekrar işlem menüsü kullanıcıya gösterilir.

```
Cihaz kontrol ediliyor...
```

```
Sıcaklık Değeri: 34.29698770951863
```

```
Cihaz işlem bekliyor...
```

```
Lütfen İşlem seçiniz
```

```
1-Soğutucu Aç
```

```
2-Soğutucu Kapat
```

```
3-Sıcaklık Görüntüle
```

```
4-Çıkış Yap
```

```
İşleminiz:
```

Eğer Kullanıcı Önce Soğutucuyu Açıp Sonra 3 Numaralı İşlemi Seçer İse Daha Düşük Aralıkta Rastgele Bir Sıcaklık Görüntülenir

Eğer Kullanıcı Açık Soğutucuyu Kapatıp Sonra 3 Numaralı İşlemi Seçer İse Daha Yüksek Aralıkta Rastgele Bir Sıcaklık Görüntülenir:

Soğutucunun Açılmasını Sağlayan Kodlar

Eyleyici.java

Soğutucu durumunu true atayıp bu değeri geri döndürür.

```
@Override
public boolean sogutucuAc() {
    this.sogutucuDurumu = true;
    return sogutucuDurumu;
}
```

MerkezIslem.java

Eğer soğutucu açık ise tekrar açılmaya çalışıldığında uyarı mesajı hazırlanıyor.

Eğer soğutucu açık değil ise soğutucu açılıyor ve sıcaklık değeri buna bağlı olarak düşüyor. Kullanıcı arayüzünde kullanıcı bilgilendiriliyor.

```
@Override
public void sogutucuAcik(IObserver observer) {
    if (this.sogutucuDurum) //true ise
    {
        System.out.println("\n*** Soğutucu zaten açık ! ***");
    } else {
        this.sogutucuDurum = islem.sogutucuAc();
        sıcaklikAlgila.sicaklikAta(sogutucuDurum);

        System.out.println("\n*** Soğutucu Açıldı ! ***");
    }
}
```

AgArayuzu.java

Kullanıcı eğer 1 numaralı işlemi seçer ise merkezi işlem birimi aracılığı ile sogutucuAcik() fonksiyonu çalıştırılır. O da eyleyicideki sogutucuAc() fonksiyonunu çağırır. Soğutucu açılmış olur ve soğutucunun durumu açık olarak değiştirilmiş olur, sıcaklık düşmeye başlar.

```
@Override
public void islem() {

    Scanner islemSecim = new Scanner(System.in);

    while (true) {
        System.out.println("Lütfen İşlem seçiniz");
        System.out.println("1-Soğutucu Aç");
        System.out.println("2-Soğutucu Kapat");
        System.out.println("3-Sıcaklık Görüntüle");
        System.out.println("4-Çıkış Yap");
        System.out.print("İşleminiz: ");
        int secim = islemSecim.nextInt();
        switch (secim) {
            case 1:
                kontrol();
                merkezIslem.sogutucuAcik( observer: this);
                beklemede();
                break;
        }
    }
}
```

Soğutucunun Kapanmasını Sağlayan Kodlar

Eyleyici.java

Soğutucu durumunu false atayıp bu değeri geri döndürür.

```
@Override
public boolean sogutucuKapat() {
    this.sogutucuDurumu = false;
    return sogutucuDurumu;
}
```

MerkezIslem.java

Eğer soğutucu kapalı ise tekrar kapatılmaya çalışıldığında uyarı mesajı hazırlanıyor. Eğer soğutucu kapalı değil ise soğutucu kapatılıyor ve sıcaklık değeri buna bağlı olarak artıyor. Kullanıcı arayüzünde kullanıcı bilgilendiriliyor.

```
@Override
public void sogutucuKapali(IObserver observer) {
    if (!this.sogutucuDurum) //false ise
    {
        System.out.println("\n*** Soğutucu zaten kapalı !***");
    } else {
        this.sogutucuDurum = islem.sogutucuKapat();
        sıcaklikAlgila.sicaklikAta(sogutucuDurum);

        System.out.println("\n*** Soğutucu Kapalı! ***");
    }
}
```


AgArayuzu.java

Kullanıcı eğer 2 numaralı işlemi seçer ise merkezi işlem birimi aracılığı ile sogutucuKapali() fonksiyonu çalıştırılır. O da eyleyicideki sogutucuKapat() fonksiyonunu çağırır. Soğutucu kapanmış olur ve soğutucunun durumu kapalı olarak değiştirilmiş olur, sıcaklık artmaya başlar.

```
case 2:
    kontrol();
    merkezIslem.sogutucuKapali( observer: this);
    beklemede();
    break;
```

Sıcaklığın Görüntülenmesini Sağlayan Kodlar

SicaklikAlgilyici.java

Soğutucunun açık veya kapalı olma durumuna göre ortamın sıcaklığına rastgele bir değer ataması yapan sıcaklikAta() ve bu değer geri döndürülmesini sağlayan sıcaklikOku() fonksiyonlarını içerir. Cihaz ilk çalıştırıldığında kurucu fonksiyon ile 18-28 derece arasında rastgele bir double sıcaklık değeri ataması gerçekleştirilir. Ayrıca bu sınıftan nesne üretilirken Builder modeli kullanılmıştır

```
package SogutucuKontrolCihazı;

public class SicaklikAlgilyici implements ISicaklikAlgilyici {
    private double sıcaklik;

    private SicaklikAlgilyici() { sıcaklik = (Math.random() * 10) + 28; //18~28 }

    @Override
    public void sıcaklikAta(boolean sogutucuAcikMi) {
        if (sogutucuAcikMi) //true ise
        {
            this.sıcaklik = (Math.random() * 8) + 10; //10~18
        } else {
            this.sıcaklik = (Math.random() * 10) + 25; //25~35
        }
    }

    @Override
    public double sıcaklikOku() { return this.sıcaklik; }

    public static class SicaklikAlgilyiciBuilder {
        private double sıcaklik;

        public SicaklikAlgilyici build() { return new SicaklikAlgilyici(); }
    }
}
```

Observer Tasarım Deseni:

Çok sayıda nesneye , gözlemledikleri nesnede meydana gelen olayı bildirme amacını içermektedir. Bir nesne durumu değiştirildiğinde ona bağlı diğer nesnelerin tümü uyarılır ve otomatik olarak güncellenir.

(Yararlanılan Kaynak:

<https://github.com/celalceken/NesneYoneliMliAnalizVeTasarimDersiUygulamalari/blob/master/Ders10/TasarimDesenleriSingletonObserver.pdf>)

IObservableMerkezIslem.java

```
package SogutucuKontrolCihazı;  
  
public interface IObservableMerkezIslem {  
    void sogutucuAcik(IObserver observer);  
  
    void sogutucuKapali(IObserver observer);  
  
    void sicaklikGoster(IObserver observer);  
}
```

IObserver.java

```
package SogutucuKontrolCihazı;  
  
public interface IObserver {  
    void beklemede();  
  
    void kontrol();  
  
    void islem();  
}
```

Builder Tasarım Deseni:

Karmaşık nesnelerin (içerisinde çok sayıda üye değişken ve üye nesne olan) oluşturulması için kullanılır. Karmaşık bir nesnenin yapımını, temsilinden (sunumundan) ayırır. Böylece, aynı yapım süreci farklı temsiller oluşturabilir. Nesnelerin farklı temsillerinin (sunumlarının) her biri için ayrı ayrı yapıcı tanımlamak yerine, nesne oluşturma işini adım adım gerçekleştiren “builder” deseni kullanılabilir.

(Yararlanılan Kaynak:

https://dl.sabis.sakarya.edu.tr/p/ac2Hcw6fOhVHnRGzTTq_3oUMw4GFje8kIeKs2c-oRfwhJCGb9xkpe2MqG4Oj-uX00)

Eyleyici.java

```
package SogutucuKontrolCihazı;  
  
public class Eyleyici implements IEyleyici {  
    private boolean sogutucuDurumu;  
  
    private Eyleyici(EyleyiciBuilder builder) {  
  
    }  
  
    @Override  
    public boolean sogutucuAc() {  
        this.sogutucuDurumu = true;  
        return sogutucuDurumu;  
    }  
  
    @Override  
    public boolean sogutucuKapat() {  
        this.sogutucuDurumu = false;  
        return sogutucuDurumu;  
    }  
  
    public static class EyleyiciBuilder {  
  
        public Eyleyici build() { return new Eyleyici( builder, this); }  
    }  
}
```

SicaklikAlgılayıcı.java

```
package SogutucuKontrolCihazı;  
  
public class SicaklikAlgılayıcı implements ISicaklikAlgılayıcı {  
    private double sicaklik;  
  
    private SicaklikAlgılayıcı() { sicaklik = (Math.random() * 10) + 28; //18~28 }  
  
    @Override  
    public void sicaklikAta(boolean sogutucuAcikMi) {  
        if (sogutucuAcikMi) //true ise  
        {  
            this.sicaklik = (Math.random() * 8) + 10; //10~18  
        } else {  
            this.sicaklik = (Math.random() * 10) + 25; //25~35  
        }  
    }  
  
    @Override  
    public double sicaklikOku() { return this.sicaklik; }  
  
    public static class SicaklikAlgılayıcıBuilder {  
        private double sicaklik;  
  
        public SicaklikAlgılayıcı build() { return new SicaklikAlgılayıcı(); }  
    }  
}
```

Dependency Inversion Prensipleri:

Nesneler arasındaki bir bağlantıda, yüksek seviyeli modül ile düşük seviyeli modül (her ikisi birden) soyutlamaya bağlı olmalı. Her ikisi birbirine doğrudan değil arayüz üzerinden bağlanmalı. Böylece modülde yapılacak değişiklik diğer modülleri etkilememiş olur. Değişiklik yapmak kolaylaşır. Yeni özellik eklemek kolaylaşır (OCP). Modüllerin tekrar kullanım oranı artar. OCP ve LSP uygulandığında, aynı zamanda DIP uygulanmış olur.

(Yararlanılan Kaynak:

<https://dl.sabis.sakarya.edu.tr/p/ m8LA9r6aVzalbQ VQKjAvvZBA23N2PPH2K JlrKMpawCqXAIqRMQ3WQxPy4F yZ0>)

```
public class MerkezIslem implements IObservableMerkezIslem {
    private boolean sogutucuDurum;

    /* Dependency Inversion */
    IEyleyici islem = new Eyleyici.EyleyiciBuilder()
        .build(); //builder kullanımı






    ISicaklikAlgilyici sicaklikAlgila = new SicaklikAlgilyici.SicaklikAlgilyiciBuilder()
        .build(); //builder kullanımı
```





- IEyleyici
- IObservableMerkezIslem
- IObserver
- ISicaklikAlgilyici

Gerekli birimleri soyutlama yaparak Merkez İşlem'e bağladık

Veritabanı Görüntüleri:

Databases	Schemas	Tables	Fields
Sogutucu...olCihazı	public	kullanici	isim kullaniciadi kullanicino kullanicisifre

Database:  SogutucuKontrolCihazı  Schema:  public  Table:  kullanici

		kullanicino	kullaniciadi	kullanicisifre	isim
1		1	ilkrkck	159753	ilker
2		2	yetkili1	yetkiliKisi	celal

Video Sunum Link:

<https://youtu.be/PT6EGyC11os>

Github:

https://github.com/ilkrkck/NesneYonelimliAnalizveTasarim_Proje