DERS: NESNEYE YÖNELİMLİ ANALİZ VE TASARIM

AD – SOYAD: SERCAN ÖZER

NUMARA: B201210080

SINIF/ŞUBE: 2.SINIF/1A

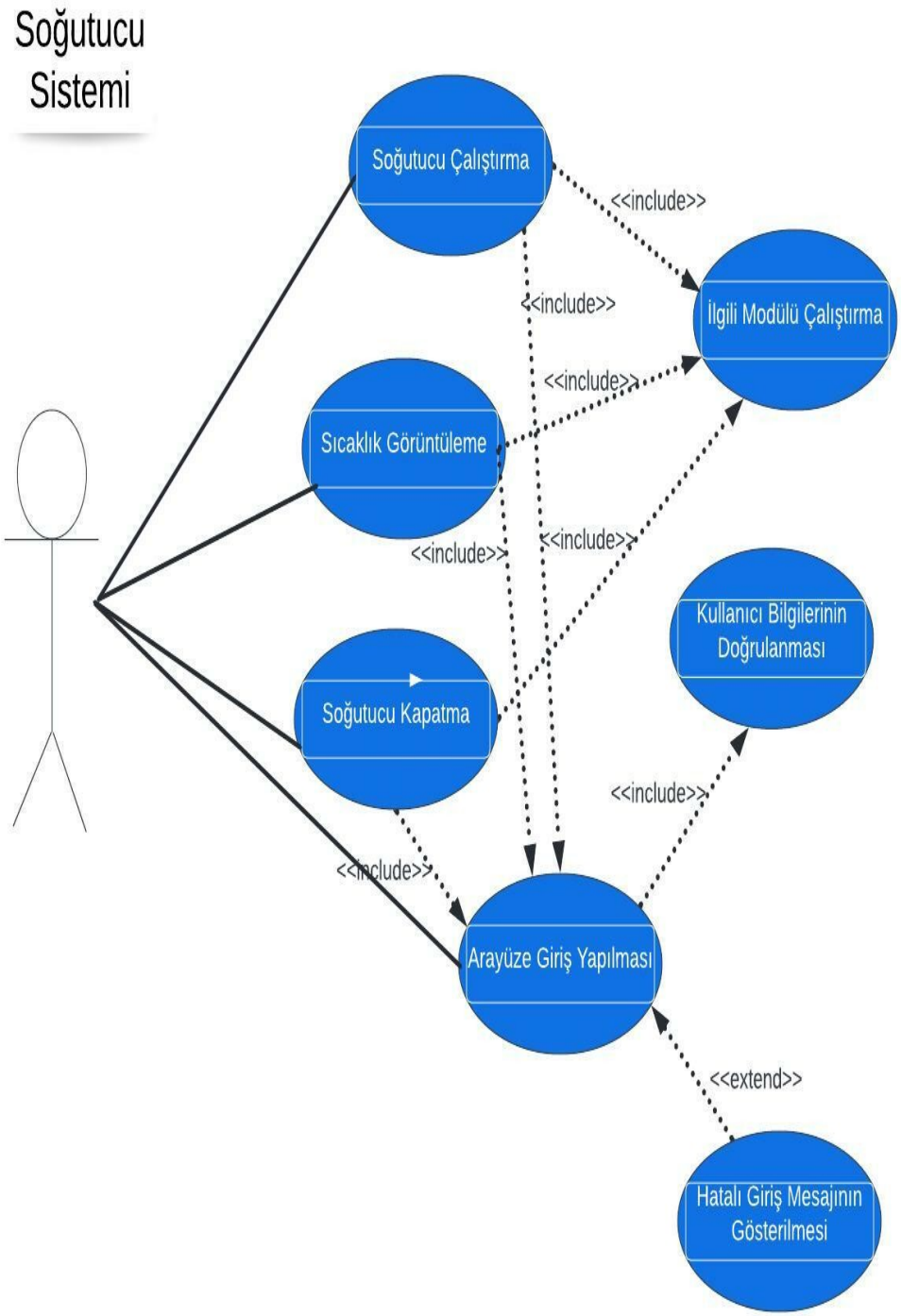
AD – SOYAD: MUSTAFA ÜNLÜ

NUMARA: B201210392

SINIF/ŞUBE: 2.SINIF/1A

GİTHUB LİNKLERİ:

# KULLANIM DURUMU (USE CASE)



## Sıcaklığın Görüntülenmesi

- Kullanım durumunun adı: Arayüz Kullanılarak Sıcaklığın Görüntülenmesi
- Hazırlayanlar: Mustafa Ünlü – Sercan Özer
- Sürüm: v1.1.1
- Tarih: 02.05.2022 – 07.05.2022
- İlgili Aktörler: İnternet kullanıcısı
- Giriş Koşulu: Sisteme kayıtlı kullanıcı arayüze bağlanarak sıcaklığı görüntülüme seçeneğini seçer
- Çıkış Koşulu: Kullanıcı işlemi tamamlayıp çıkış seçeneğini seçer.
- Özel Gereksinim: Arayüz tasarımına sahip olması, elektrik gücü, internet bağlantısı

### Ana Olay Akışı

1. Kullanıcı arayüze bağlanır.
2. Arayüz kullanıcıya giriş ekranını getirir.
3. Kullanıcı arayüz yardımıyla giriş bilgilerini girer.
4. Sistem veritabanına bağlanır.
5. Veritabanı yardımı ile kullanıcı doğrulanır.
6. Arayüz kullanıcıya menüyü getirir.
7. Merkezi sistem kullanıcının isteğine karşılık sıcaklık algılayıcıyı çalıştırır.
8. Sıcaklık görüntülenir.

### Alternatif Olay Akışı

A1. Kullanıcının bilgileri doğrulanamadı. (5)
6. Ekrana kullanıcının doğrulanamadığı yazdırılır.
7. İşlem sonlandırılır.

## Soğutucunun Çalıştırılması

- Kullanım durumunun adı: Arayüz Kullanılarak Soğutucunun Çalıştırılması
- Hazırlayanlar: Mustafa Ünlü – Sercan Özer
- Sürüm: v1.1.1

- Tarih: 02.05.2022 – 07.05.2022
- İlgili Aktörler: İnternet kullanıcısı
- Giriş Koşulu: Sisteme kayıtlı kullanıcı arayüze bağlanarak sıcaklığı görüntülüme seçeneğini seçer
- Çıkış Koşulu: Kullanıcı işlemi tamamlayıp çıkış seçeneğini seçer.
- Özel Gereksinim: Arayüz tasarımına sahip olması, elektrik gücü, internet bağlantısı

Ana Olay Akışı

1. Kullanıcı arayüze bağlanır.
2. Arayüz kullanıcıya giriş ekranını getirir.
3. Kullanıcı arayüz yardımıyla giriş bilgilerini girer.
4. Sistem veritabanına bağlanır.
5. Veritabanı yardımı ile kullanıcı doğrulanır.
6. Arayüz kullanıcıya menüyü getirir.
7. Merkezi sistem kullanıcının seçimine karşılık soğutucuyu açma isteğini alır.
8. Soğutucu önceden açık değilse merkezi sistem soğutucuyu açma isteğini eyleyiciye iletir.
9. Eyleyici soğutucuyu açma işlemini gerçekleştirir.

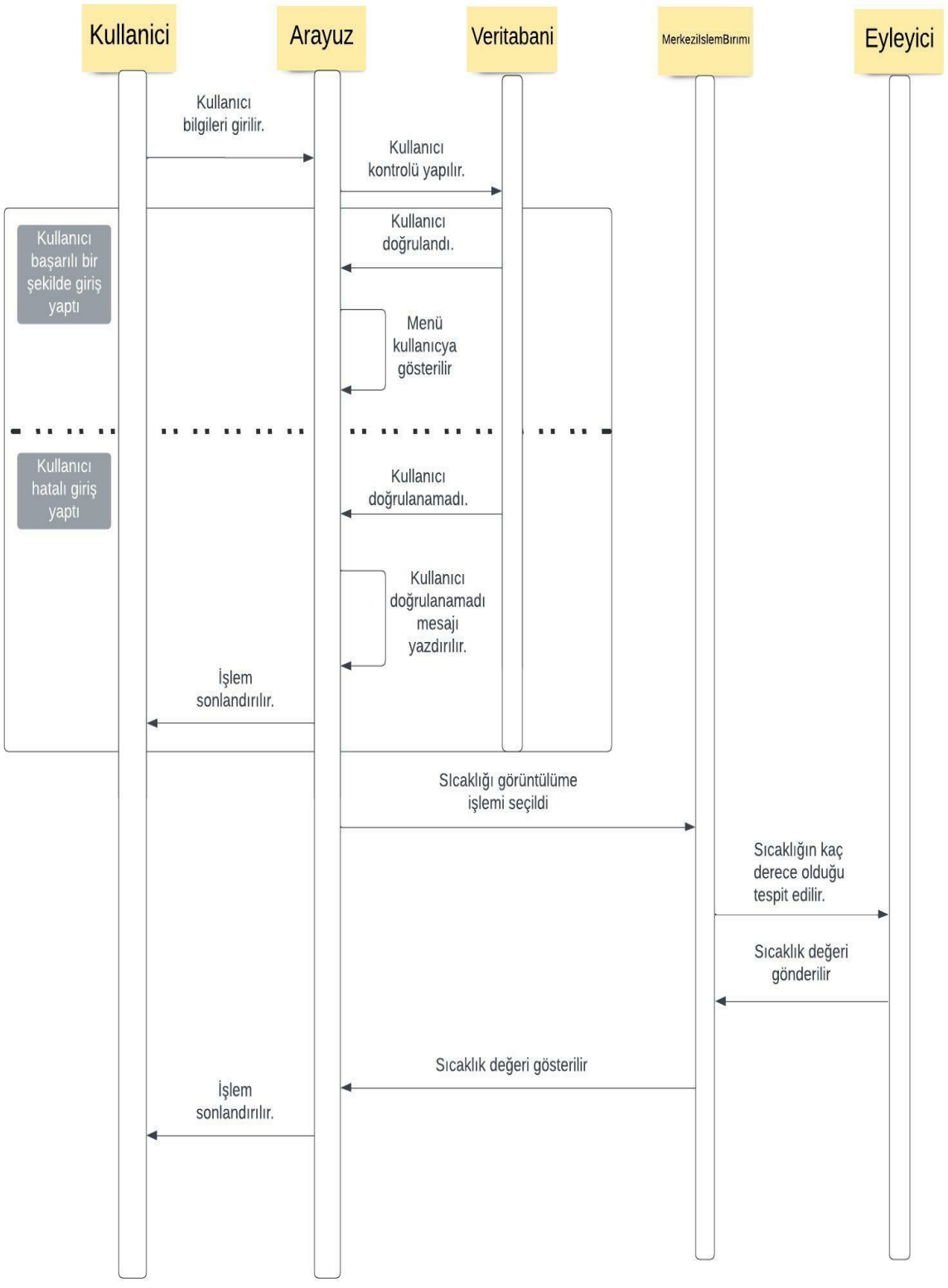
Alternatif Olay Akışı

A1. Kullanıcının bilgileri doğrulanamadı. (5)
6. Ekrana kullanıcının doğrulanamadığı yazdırılır.
7. İşlem sonlandırılır.

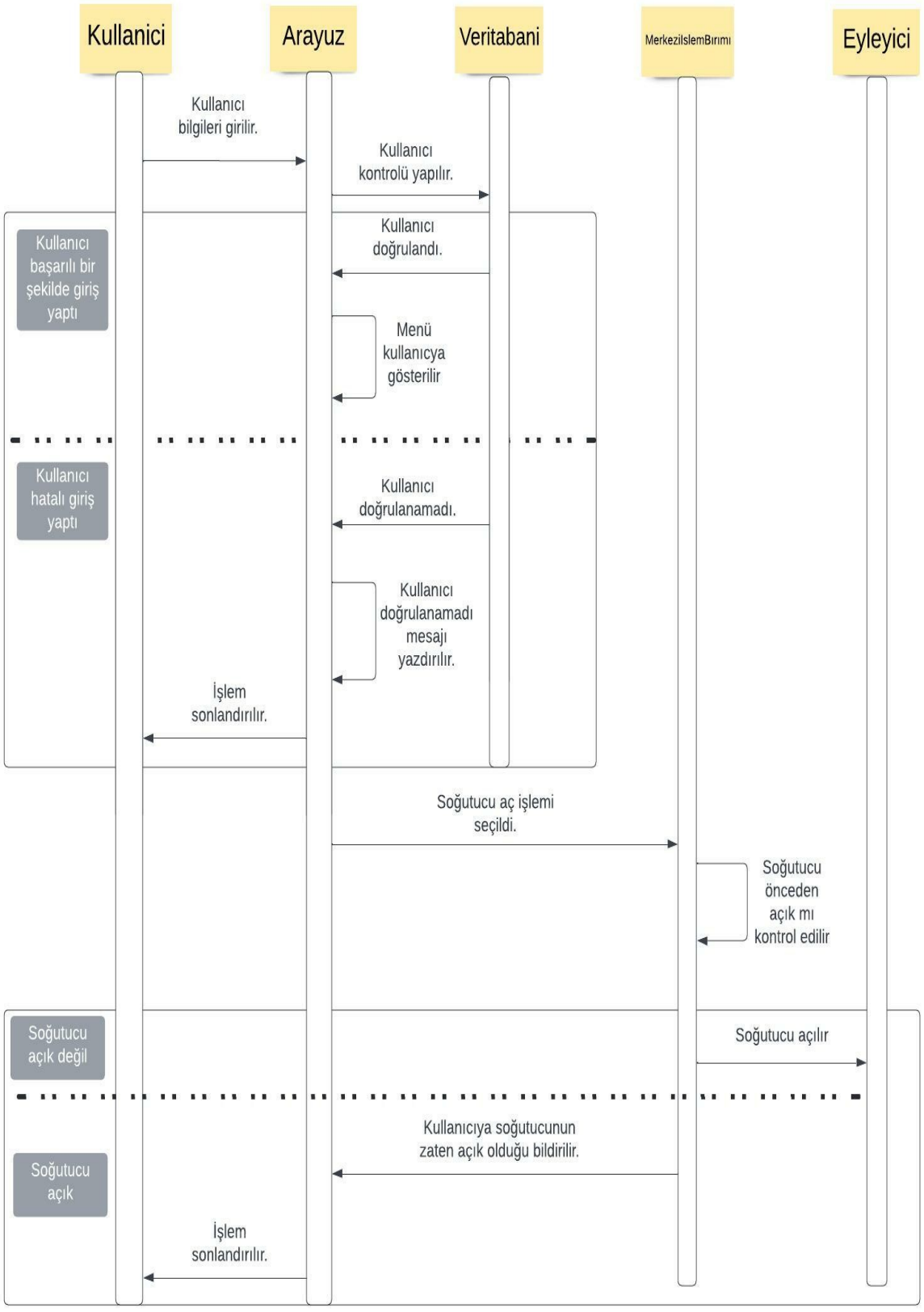
A2. Soğutucu önceden açılmış. (8)
9. Merkezi sistem kullanıcının isteğini eyleyiciye iletmez.
10. Kullanıcıya soğutucunun zaten açık olduğu mesajla bildirilir.
11. İşlem sonlandırılır.

# SIRALAMA ŞEMALARI (SEQUENCE DİAGRAM)

## Sıcaklığın Görüntülenmesi

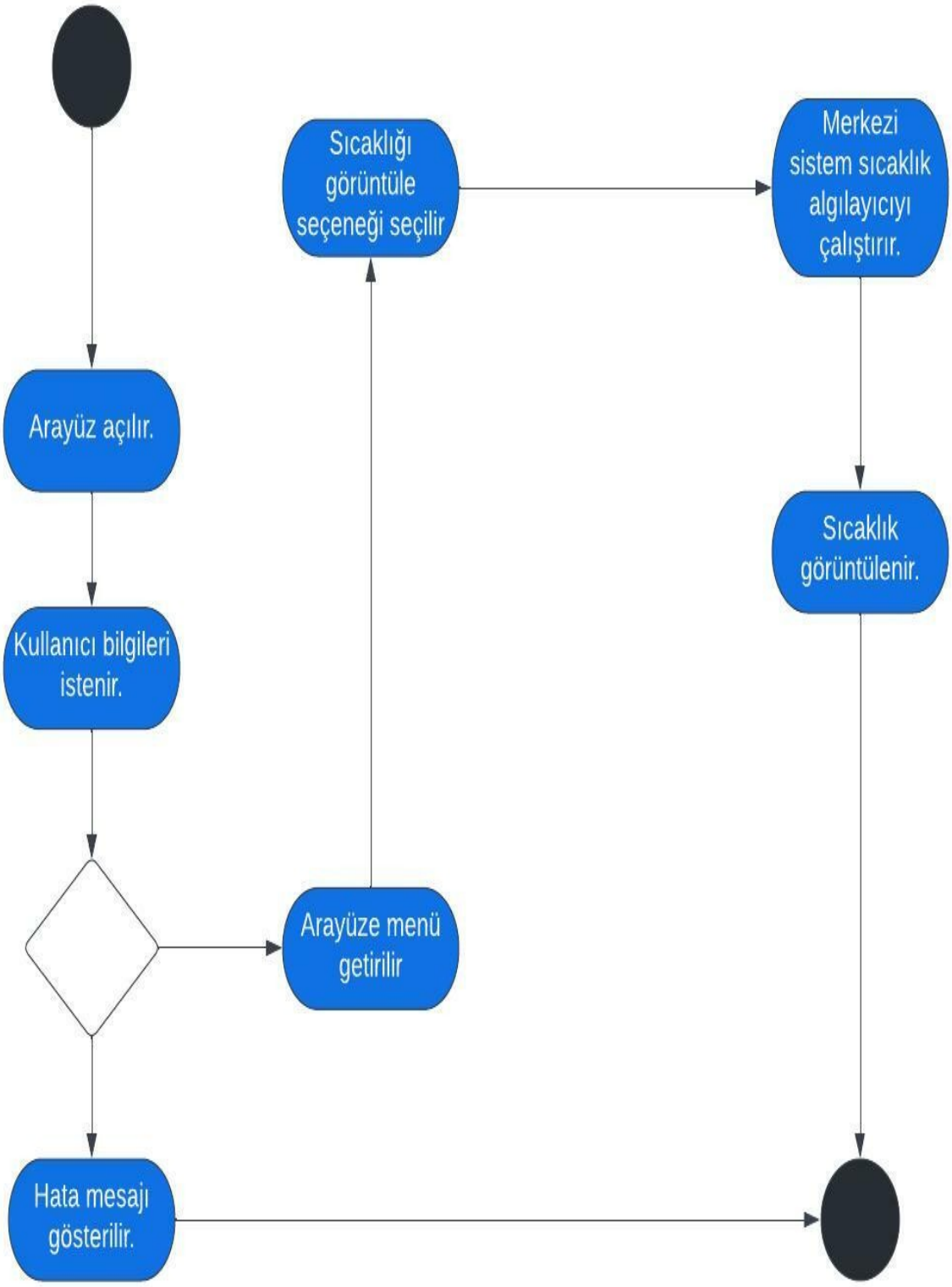


# Soğutucunun Çalıştırılması

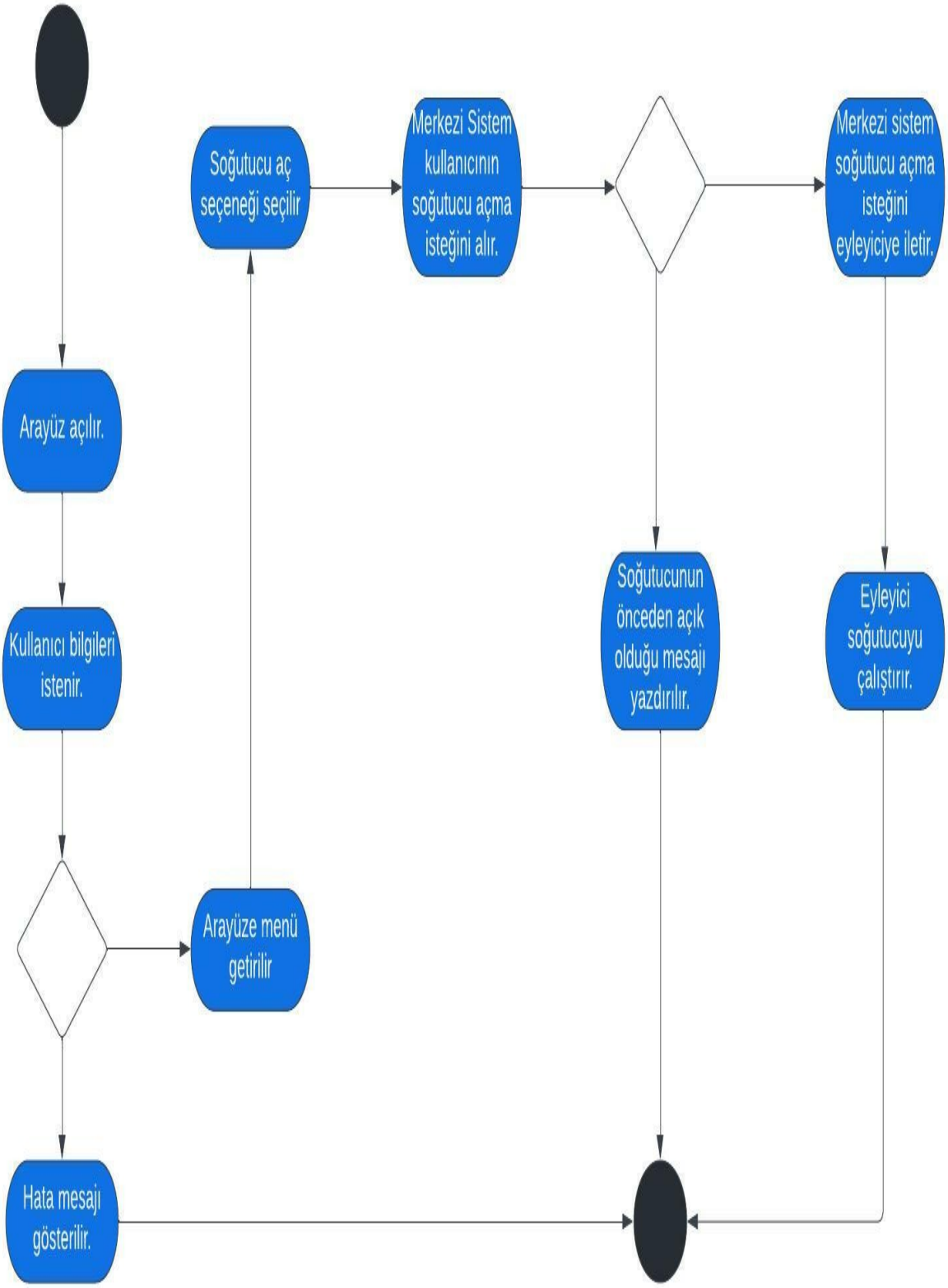


# ETKİNLİK ŞEMALARI (ACTİVİTY DİAGRAM)

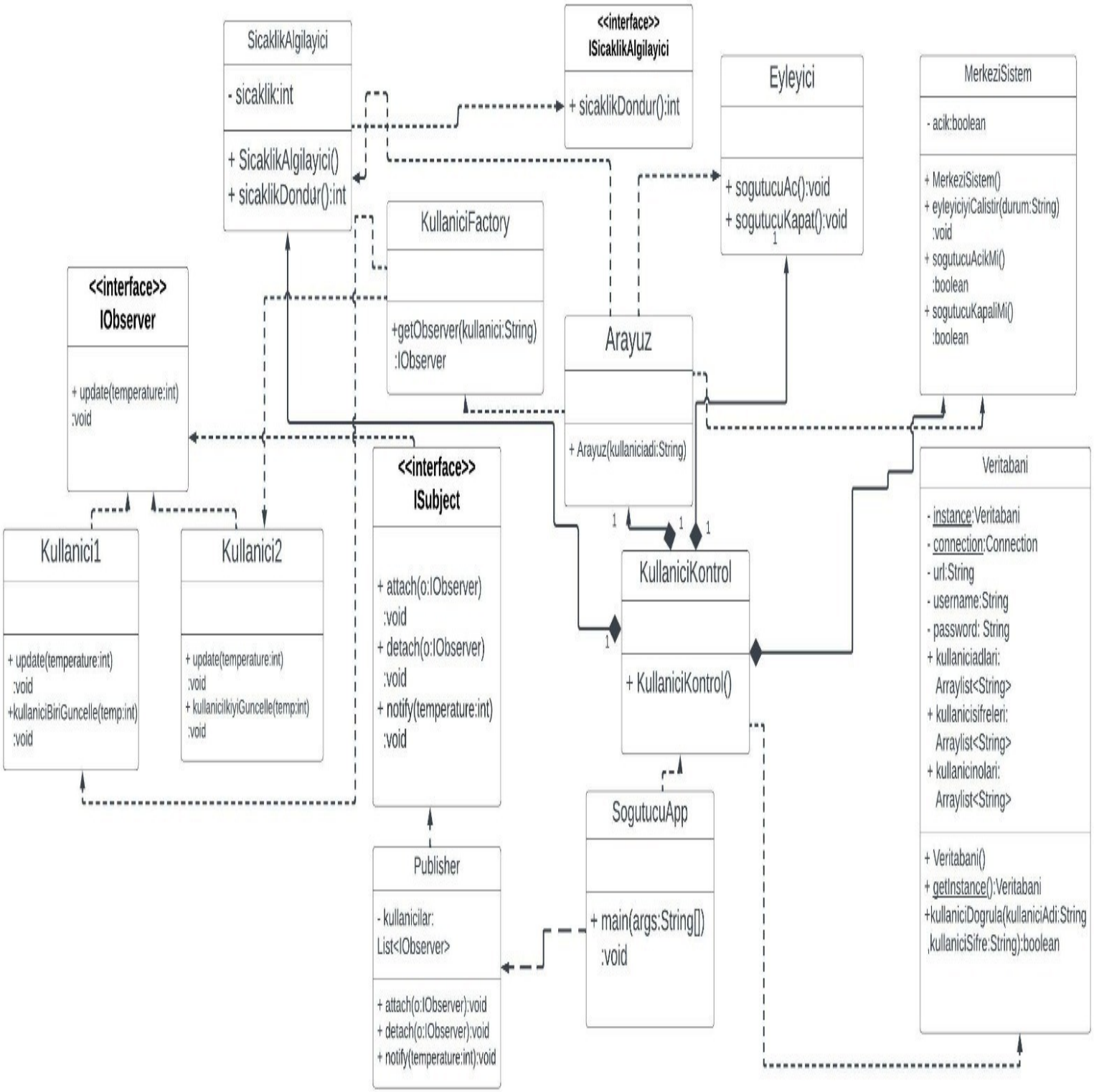
## Sıcaklığın Görüntülenmesi



Soğutucunun Çalıştırılması



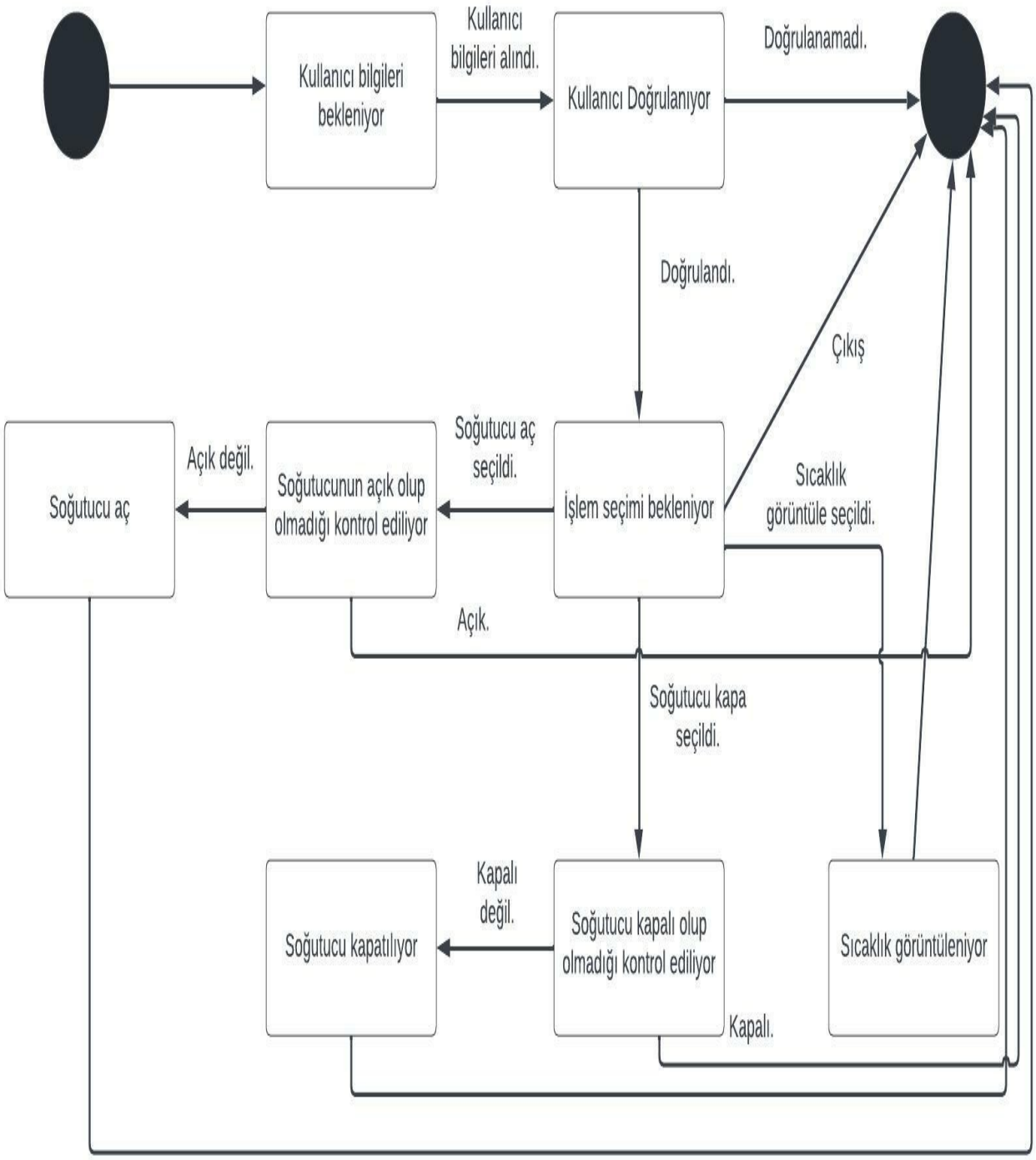
# SINIF ŞEMASI (CLASS DİAGRAM)



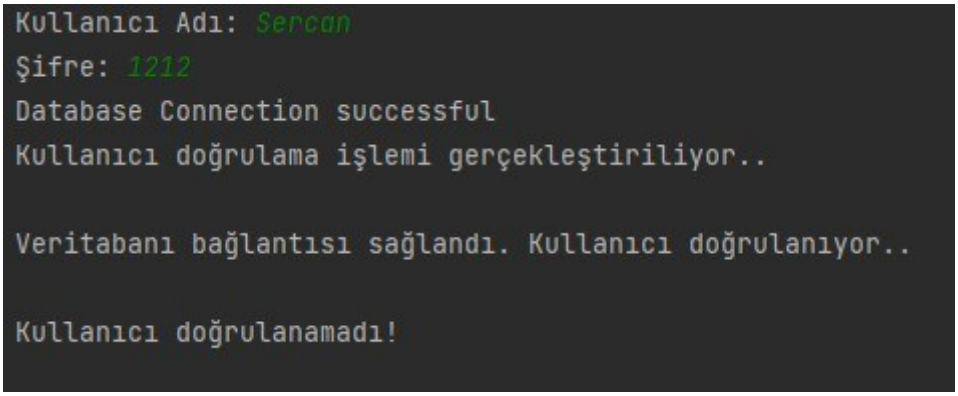
CRC KART

| Soğutucu Uygulaması | |
|---|---|
| Sorumluluk | İş birliği Yapılan Sınıf |
| Uygulamanın başlatılması | SogutucuApp |
| Kullanıcı giriş bilgilerini iste | KullaniciKontrol |
| Kullanıcı giriş bilgilerini gir | KullaniciKontrol |
| Kullanıcı doğrula | KullaniciKontrol,Veritabani |
| Menü işlemleri | Arayüz |
| Soğutucu işlemleri | MerkeziSistem,Eyleyici |
| Sıcaklık işlemleri | SicaklikAlgilayici |
| Kullanıcı işlemleri | KullaniciFactory, Publisher, Kullanici1, Kullanici2 |
| Uygulamanın kapatılması | SogutucuApp |

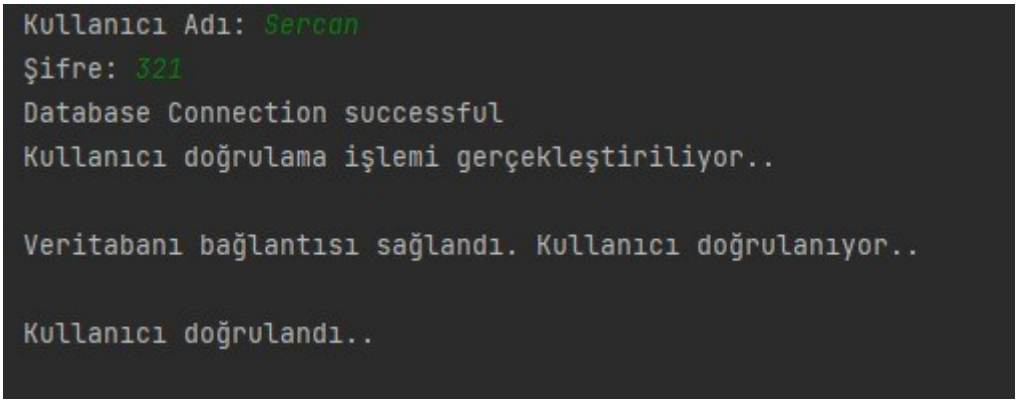
# DURUM MAKİNE (STATE MACHİNE) DİAGRAM



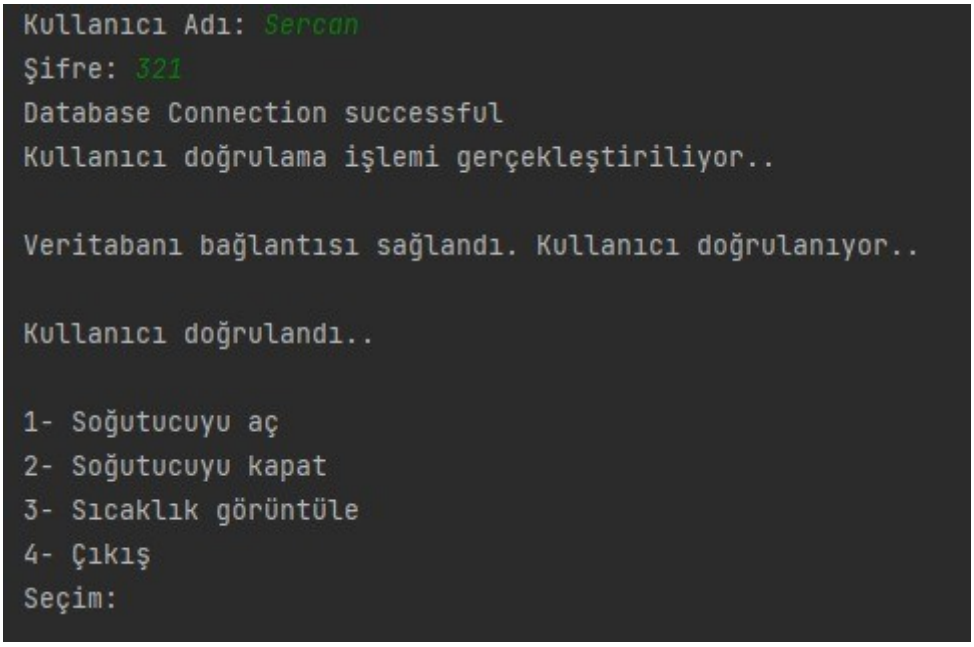
# KULLANICI DOĞRULAMA EKRANI




Kullanıcıdan kullanıcı adı ve şifre bilgileri girilmesi istenir. Kullanıcı bilgileri yanlış veya eksik girerse “Kullanıcı doğrulanamadı!” mesajı ile karşılaşır ve program sonlandırılır.

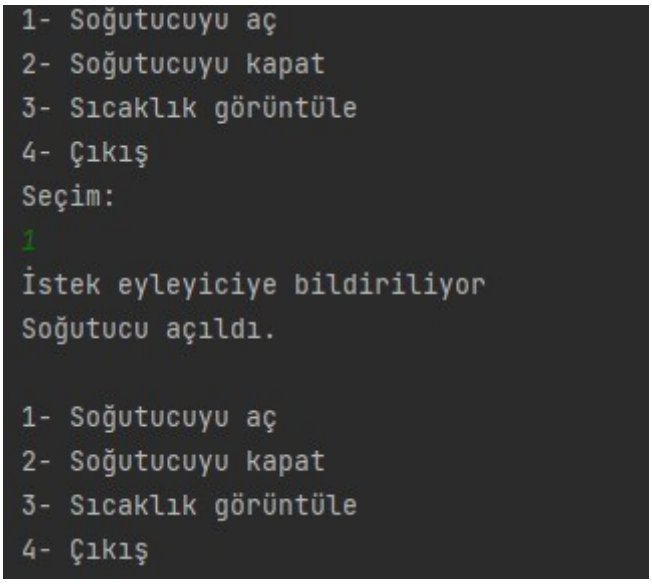



Kullanıcı eğer bilgilerini eksiksiz ve doğru bir şekilde girerse “Kullanıcı doğrulandı..” mesajı ile karşılaşır.



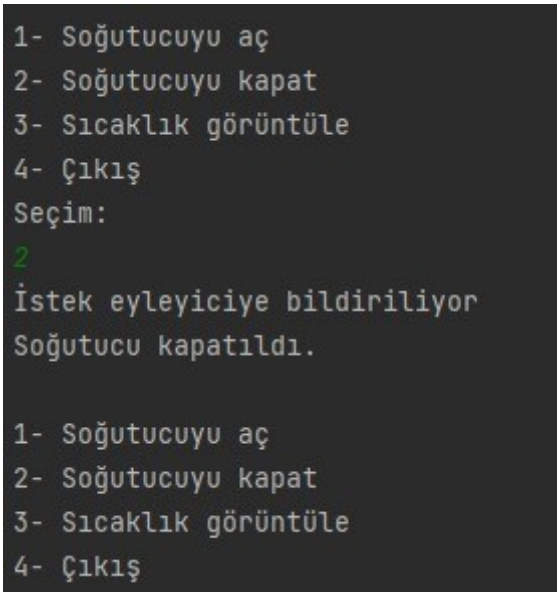

Kullanıcı sisteme giriş yaptıktan sonra içinde soğutucuyu aç, soğutucuyu kapat, sıcaklık görüntüle ve çıkış işlemleri içeren bir menü ile karşılaşır.

# SOĞUTUCUNUN AÇILMASI



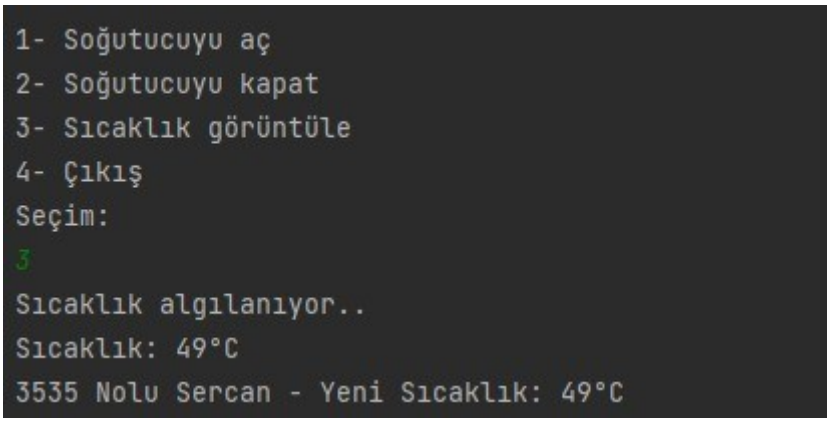

Kullanıcı menüden 1. Seçeneği seçerse program soğutucunun açık olup olmadığını kontrol eder. Açıksa ekrana “Soğutucu açık durumda!”, açık değilse ekrana “Soğutucu açıldı.” mesajı çıkartır.

# SOĞUTUCUNUN KAPATILMASI



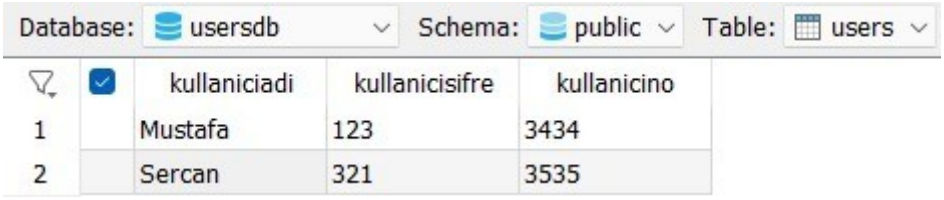

Kullanıcı menüden 2. Seçeneği seçerse program soğutucunun kapalı olup olmadığını kontrol eder. Kapalıysa ekrana “Soğutucu kapalı durumda!”, kapalı değilse ekrana “Soğutucu kapatıldı.” mesajı çıkartır.

# SICAKLIĞIN GÖRÜNTÜLENMESİ




Kullanıcı menüden 3. Seçeneği seçerse sıcaklık değerini rastgele seçip ekrana “Sıcaklık: 49°C” mesajı çıkartır.

# VERİTABANI EKRAN GÖRÜNTÜSÜ



Database: usersdb Schema: public Table: users

| | kullaniciadi | kullanicisifre | kullanicino |
|---|---|---|---|
| 1 | Mustafa | 123 | 3434 |
| 2 | Sercan | 321 | 3535 |

# DEPENDENCY INVERSİON

SOLİD yazılım prensibinin son ilkesi olan dependency inversion bize sınıflar arasındaki bağlantıyı minimuma indirerek üst seviye sınıfların alt seviye sınıflara bağımlılıklarını yok etmeyi amaçlar. Yani alt seviye sınıfta yapılan bir değişiklik doğrudan üst seviye sınıfa etki etmemelidir. Aksi takdirde üst seviye sınıfın davranışında bir bozulma meydana gelebilir. Sınıflar arasındaki yukarıda bahsedilen bu bağımlılık interface yardımıyla yok edilebilir. Yeni bir interface oluşturulup alt seviye sınıfa implement edilir. Alt seviye sınıf sahip olduğu methodu interface içinde çağırır. Üst seviye sınıfta interface alt seviye sınıfa göre başlatılır ve ilgili method çağrılır.

```java
package nesneProject;

public interface IObserver {
    public void update(int temperature);
}
```

```java
package nesneProject;

public class Kullanici1 implements IObserver{

    @Override
    public void update(int temperature) {
        kullaniciBiriGuncelle(temperature);
    }
    public void kullaniciBiriGuncelle(int temp){
        System.out.println("3535 Nolu Sercan - Yeni Sıcaklık: "+temp+"°C");
    }
}
```

```java
package nesneProject;

public class Kullanici2 implements IObserver{

    @Override
    public void update(int temperature) {
        kullaniciIkiyiGuncelle(temperature);
    }
    public void kullaniciIkiyiGuncelle(int temp){
        System.out.println("3434 Nolu Mustafa - Yeni Sıcaklık: "+temp+"°C");
    }
}
```

IObserver Kullanici1 ve Kullanici2 sınıflarına implement edilir ve update metodu sınıfı her iki sınıfa da eklenir. Sınıfların kendi methodları yani kullaniciBiriGuncelle ve kullaniciIkiyiGuncelle methodları update metodu içerisinde çağrılarak koruma altına alınmış olur.

```java
KullaniciFactory kullaniciFactory=new KullaniciFactory();
IObserver k1=kullaniciFactory.getObserver( kullanici: "k1");
IObserver k2=kullaniciFactory.getObserver( kullanici: "k2");
```

```java
if(kullaniciadi.equals("Mustafa"))
    k2.update(sicaklik);
else
    k1.update(sicaklik);
```

IObserver arayüzünü factory kullanarak  başlatmış olduk. Sonrasında ilgili yerimizde update metodlarını çağırdık.

# FACTORY METHOD

```java
package nesneProject;

public interface IObserver {
    public void update(int temperature);
}
```

```java
package nesneProject;

public class Kullanici1 implements IObserver{

    @Override
    public void update(int temperature) {
        kullaniciBiriGuncelle(temperature);
    }
    public void kullaniciBiriGuncelle(int temp){
        System.out.println("3535 Nolu Sercan - Yeni Sıcaklık: "+temp+"°C");
    }
}
```

```java
package nesneProject;

public class Kullanici2 implements IObserver{

    @Override
    public void update(int temperature) {
        kullaniciIkiyiGuncelle(temperature);
    }
    public void kullaniciIkiyiGuncelle(int temp){
        System.out.println("3434 Nolu Mustafa - Yeni Sıcaklık: "+temp+"°C");
    }
}
```

Kullanici1 ve Kullanici2 sınıflarını başlatmak için KullaniciFactory adında bir sınıf oluşturuyoruz ve getObserver metodu ile bu işlemi gerçekleştiriyoruz.

```java
KullaniciFactory kullaniciFactory=new KullaniciFactory();
IObserver k1=kullaniciFactory.getObserver( kullanici: "k1");
IObserver k2=kullaniciFactory.getObserver( kullanici: "k2");
```

# OBSERVER

Observer tasarım bir nesne üzerinde değişiklik yapıldığında bu nesneye bağlı tüm birimlere haber verilmesine dayanan tasarımdır. Projemde bu deseni IObserver, ISubject interfaceleri ve Publisher sınıfı yardımı ile gerçekleştirdim. IObserver interface'i kullanıcılara bağladım böylelikle gözlemlenen nesnede bir olay geldiğinde bağlı olan kullanıcılara Publisher sınıfından nesne oluşturarak ve bu kullanıcı nesnelerini publishera ekledim. Publisher'ın notify metodunu çağırarak kullanıcılara mesaj iletilmesini sağladım.