SAKARYA
ÜNİVERSİTESİ

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

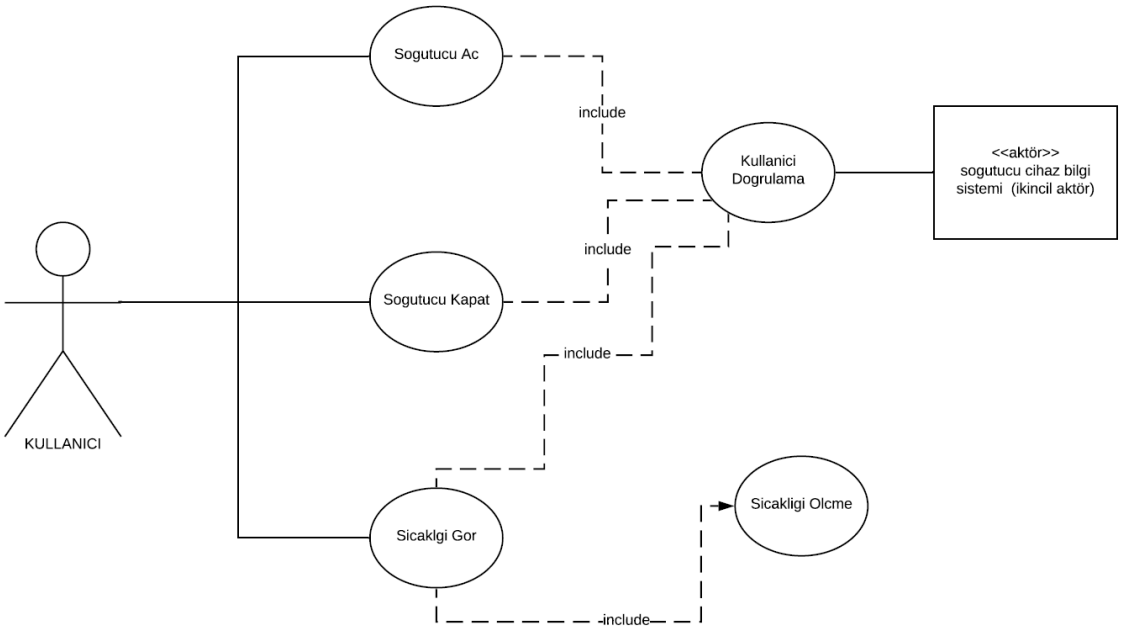
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
Nesne Yönelimli Analiz ve Tasarım

PROJE/TASARIM

Hazırlayan:
B171210082 - Emre Akcan 2. Sınıf 2. Öğretim A grubu

SAKARYA
Nisan 2020

USE CASE DIAGRAM



"SOĞUTUCUNUN ÇALIŞTIRILMASI DURUMU"

Eşsiz bir ad "Soğutucunun çalıştırılması"

- Soğutucunun başlatılması sürecini tanımlar
- 20.04.2020 v1.1 Emre Akcan

İlgili Aktörler: Kullanıcı, Cihaz aygıtları
Giriş Koşulu: Soğutucu başlatılma isteği
Çıkış Koşulu: Soğutucunun başlatılması
Olay Akışı:

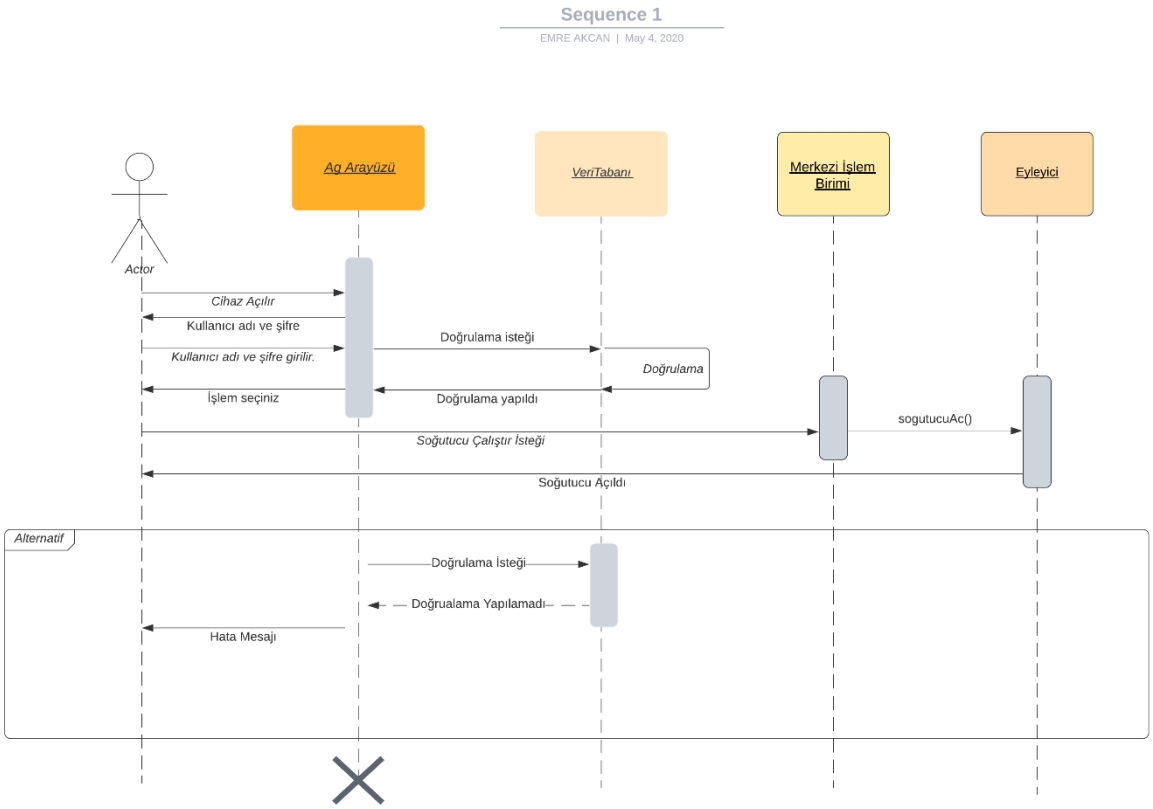
1. Kullanıcı soğutucu cihazı çalıştırır.
2. Ekrana kullanıcı adı ve şifre girilmesini istenen bir mesaj gönderilir.
3. Kullanıcı, kullanıcı adı ve şifresini girer.
4. Cihaz klavyeden girilen değerleri alır.
5. Kullanıcı doğrulama yapmak için cihaza istekte bulunur.
6. Soğutucu isteği kabul eder ve gerekli işlemleri başlatır.
7. Kullanıcı soğutucuyu başlatma isteği gönderir.
8. Soğutucu başlatılır.

Alternatif Olay Akışı:

A1.Kullanıcı adı ve şifre yanlış

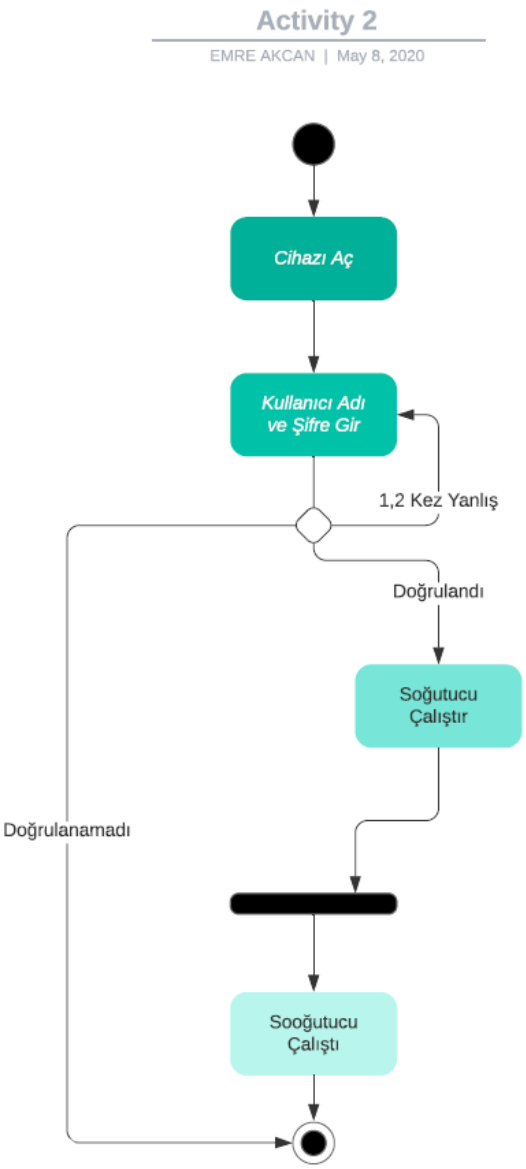
7. 3 kez yanlış girildi.
8. Çıkış.

SOĞUTUCU ÇALIŞTIRILMASI SEQUENCE DIAGRAM



SAKARYA
Nisan 2020

SOĞUTUCUNUN ÇALIŞTIRILMASI AKTİVİTE DİYAGRAMI



SAKARYA
Nisan 2020

“SICAKLIGIN GÖRÜNTÜLENMESİ DURUMU”

Eşsiz bir ad “Sıcaklığın görüntülenmesi”

- Sıcaklığın görüntülenme sürecini tanımlar
- 20.04.2020 v1.0 Emre Akcan

İlgili Aktörler: Kullanıcı, Fiziksel Ortam(Hava), Cihaz aygıtları
Giriş Koşulu: Sıcaklık görüntüleme isteği
Çıkış Koşulu: Sıcak görüntülenmesi
Olay Akışı:

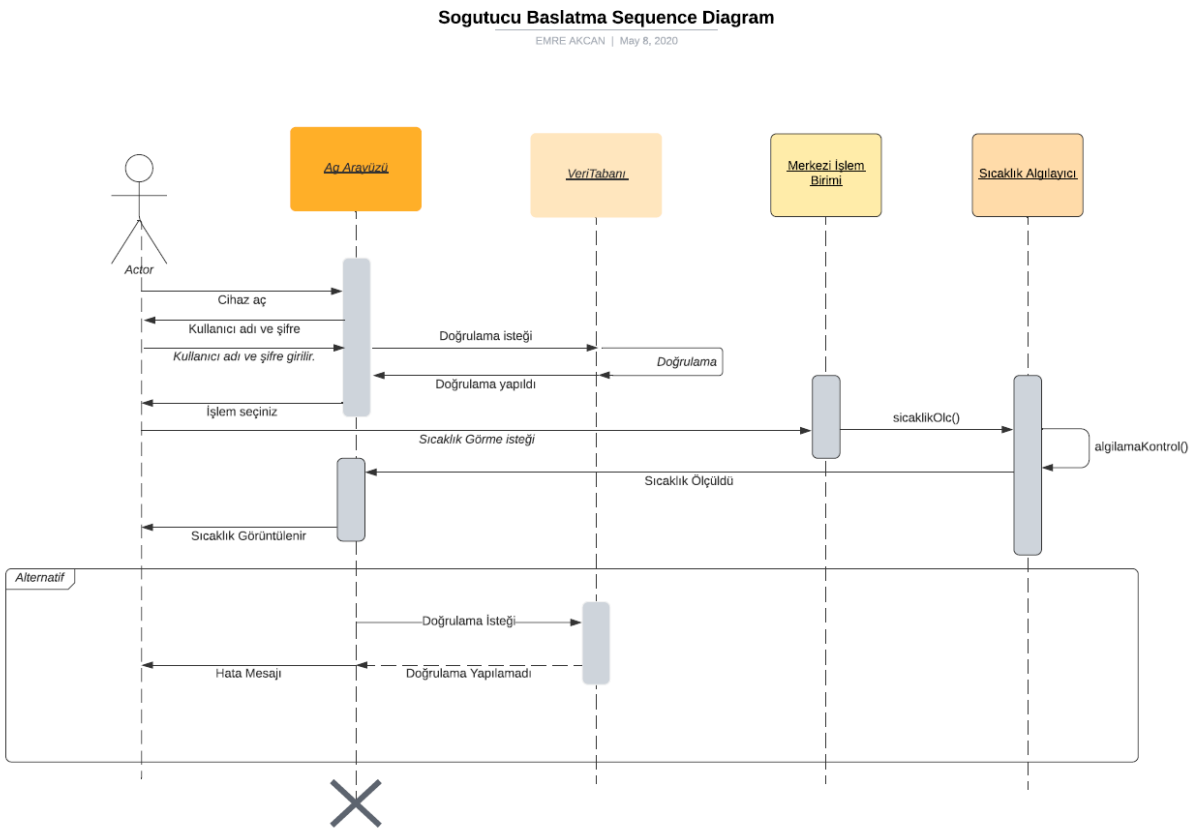
1. Kullanıcı soğutucu cihazı çalıştırır.
2. Ekrana kullanıcı adı ve şifre girilmesini istenen bir mesaj gönderilir.
3. Kullanıcı, kullanıcı adı ve şifresini girer.
4. Cihaz klavyeden girilen değerleri alır.
5. Kullanıcı doğrulama yapmak için cihaza istekte bulunur.
6. Soğutucu doğrulama yapar ve gerekli işlemleri başlatır.
7. Kullanıcı sıcaklık görüntülemek ister.
8. Cihaz, algılama ve kontrol durumuna geçer.
9. Sıcaklık ölçülür.
10. Sıcaklık görüntülenir.

Alternatif Olay Akışı:

A1. Kullanıcı adı ve şifre yanlış
6. 3 kez yanlış girildi.
7. Çıkış.

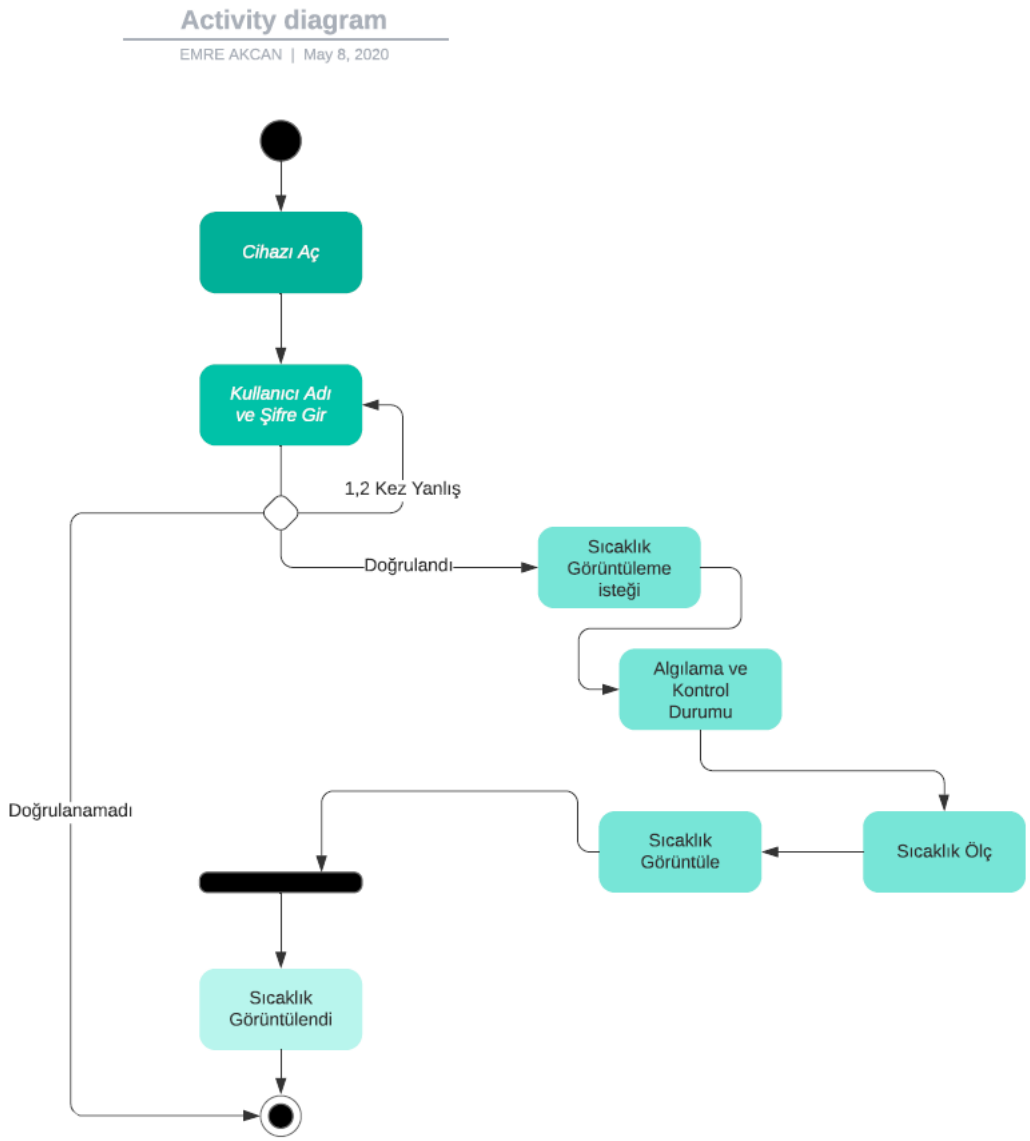
Özel Gereksinimler: Cihaz 24 saat çalışır. İstenilen sıcaklığa ulaştığında bekleme durumuna geçer. Soğutucu açılmadan herhangi bir işlem yapılamaz.

SICAKLIK GÖRÜNTÜLEME

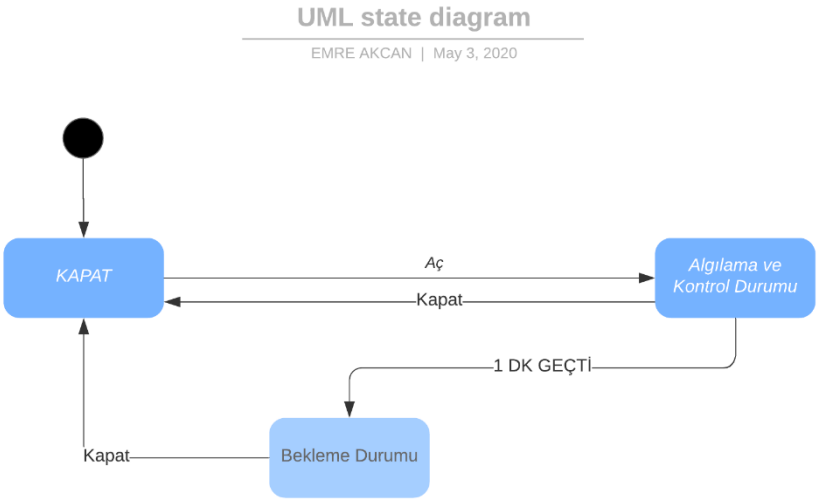


SAKARYA
Nisan 2020

SICAKLIK GÖRÜNTÜLEME AKTİVİTE DİYAGRAMI

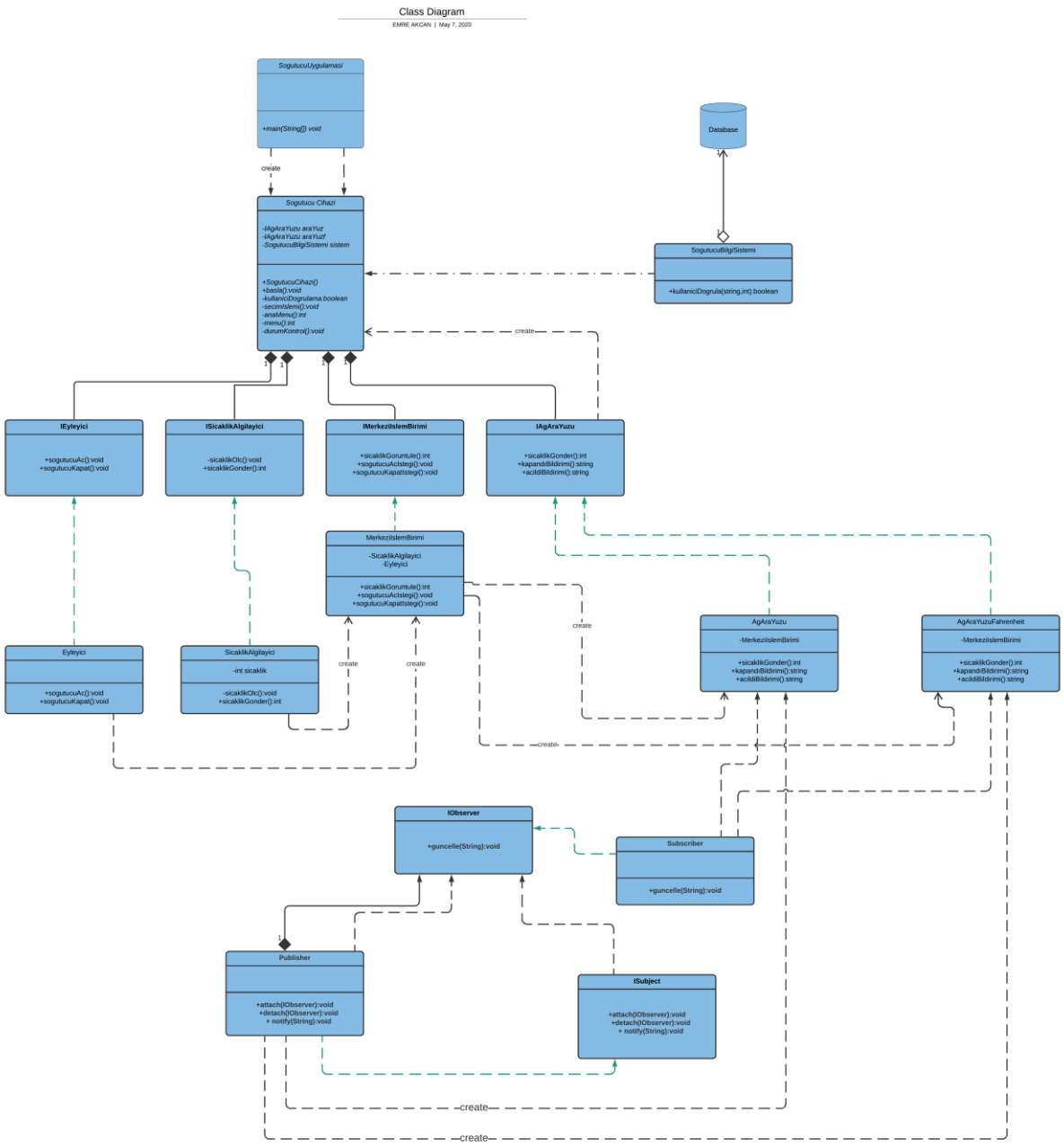


DURUM DİYAGRAMI



SAKARYA
Nisan 2020

SINIF DİYAGRAMI



KULLANICI DOĞRULAMA EKRANI

Programımız çalıştığında, ilk olarak kullanıcı cihazı açması gerekiyor. Daha sonrasında kullanıcı adı ve şifresini girmesi gerekiyor. Girilen kullanıcı adı ve şifre değerleri veri tabanında kontrol edilip sisteme giriş yapılmasını sağlıyor. Eğer minimum 3 kere yanlış girilirse sistem kendi otomatikman kapanıyor.

Başarılı Giriş:

```
*******************************************
Ana Menü
1-Sogutucu Ac
2-Cikis
Seciminiz:
*******************************************
1
Kullanici adinizi giriniz...
admin
Sifrenizi Giriniz...
1234
Veritabanına bağlandı!
Başarılı bir şekilde giriş yapıldı...
*******************************************
Menü
1-Sogutucu Baslat
2-Sogutucu Durdur
3-Sıcaklık Görüntüle
Seciminiz:
*******************************************
```

Başarısız Giriş:

```
Kullanici adinizi giriniz...
admin
Sifrenizi Giriniz...
12
Veritabanına bağlandı!
Yanlış kullanıcı adı veya şifre. Lütfen kontrol ediniz...
Kullanici adinizi giriniz...
ad
Sifrenizi Giriniz...
12
Veritabanına bağlandı!
Yanlış kullanıcı adı veya şifre. Lütfen kontrol ediniz...
Kullanici adinizi giriniz...
ad
Sifrenizi Giriniz...
32
Veritabanına bağlandı!
Yanlış kullanıcı adı veya şifre. Lütfen kontrol ediniz...
Çıkış yapılıyor...
```

SOĞUTUCUNUN BAŞLATILMASI VE SICAKLIĞIN GÖRÜNTÜLENMESİ

Soğutucu cihaza erişim gerçekleştirdikten sonra kendimizi menüde buluyoruz. Menü ekranında kullanıcıyı soğutucu başlat, soğutucu durdur ve sıcaklık görüntüle seçenekleri karşılıyor. Soğutucu başlat seçeneği ortamdaki havanın soğutulmaya başlanmasını sağlıyor. Soğutucu durdur seçeneği soğutma işleminin durdurulmasını sağlıyor. Eğer kullanıcı sıcaklık görüntülemek isterse 3. seçenekten sıcaklık fahrenheit ve celcius cinsinden görüntüleniyor.

Soğutucu Başlatılması:

```
Menü
1-Sogutucu Baslat
2-Sogutucu Durdur
3-Sıcaklık Görüntüle
Seciminiz:
**************************************
1
Sogutucu Başlatıldı...
Algılama ve Kontrol Durumuna Geçildi...
```

Soğutucu Durdurulması:

```
Menu
1-Sogutucu Baslat
2-Sogutucu Durdur
3-Sıcaklık Görüntüle
Seciminiz:
**********************************************
2
Sogutucu Durduruldu...
Cihaz kapalı durumdadır.
```

Sıcaklığın Görüntülenmesi:

```
Menü
1-Sogutucu Baslat
2-Sogutucu Durdur
3-Sıcaklık Görüntüle
Seciminiz:
******************************************
3
Admine gelen mesaj --->Ortam Sıcaklığı: 84.2 fahrenheit.
Cihaz üzerinde gözüken mesaj --> Ortam Sıcaklığı : 84 fahrenheit
Admine gelen mesaj --->Ortam Sıcaklığı: 29 derece
Cihaz üzerinde gözüken mesaj --> Ortam Sıcaklığı : 29 derece
******************************************
```

OPEN/CLOSED İLKESİ

Open/Closed ilkesinde bir sınıfın farklı davranışlara açık olması gerekir ve temel özelliklerinin sabit kalması zorunludur. Bu ilkede sınıfa yeni davranışların eklenebilmesi kolaylıkla sağlanır. Bu ilkeyi kodumdan örnek verecek olursam:

Aynı ara yüzden kalıtım alan farklı iki sınıfımız var. Biri sıcaklığın fahrenheit cinsinden gözükmesini diğeri ise celcius cinsinden gözükmesini sağlayan metodlara sahip. Fakat bu metodların ikisi de aynı metod sadece işlevleri farklı.

```
@Override
public int sicaklikGonder() {
    int sicaklik = MerkeziIslemBirimi.getObj().sicaklikGoruntule();
    publisher.notify("Ortam Sıcaklığı: " + sicaklik + " derece");
    return sicaklik;
}
```

```
@Override
public int sicaklikGonder() {
    int sicaklik = MerkeziIslemBirimi.getObj().sicaklikGoruntule();
    double fahrenheit = (sicaklik * 1.8) + 32;
    publisher.notify("Ortam Sıcaklığı: " + fahrenheit + " fahrenheit.");
    return (int)fahrenheit;
}
```

Ara yüzden nesneler tanımlayıp bu nesneler farklı sınıflardan oluşturuyorum.

```
private final IAgAraYuzu araYuz;          araYuz = new AgAraYuzu(p);
private final IAgAraYuzu araYuzf;         araYuzf = new AgAraYuzuFahrenheit(p);
```

```
                    : " + araYuzf.sicaklikGonder() +
                    : " + araYuz.sicaklikGonder() +
```

Burada nesnelerden aynı metodları çağırdığımda biri fahrenheit cinsinde değer döndürürken diğeri ise celcius cinsinde değer döndürüyor. Böylelikle open/closed ilkesini kodda uygulamış oluyoruz.

SINGLETON ve OBSERVER DESENLERİ

Singleton deseni tanımı şudur ki; yalnızca bir örneği olan ve buna global erişim noktası sağlayan bir sınıf tanımlamasıdır. Başka bir deyişle, bir sınıf yalnızca tek bir örneğin oluşturulmasını ve tek bir nesnenin diğer tüm sınıflar tarafından kullanılabilmesini sağlamalıdır.

```
public class SicaklikAlgilayici implements ISicaklikAlgilayici {

    private static final SicaklikAlgilayici obj = new SicaklikAlgilayici();

    private SicaklikAlgilayici() {}

    public static SicaklikAlgilayici getObj()
    {
        return obj;
    }
```

Sıcaklık algılayıcı sınıfı içerisinde ona ait bir örnek oluşturup, merkezi işlem birimi sınıfı içerisinde ondan nesne üretmeden içerisindeki metoda ulaşıyoruz.

```
@Override
public String sogutucuAcIstegi() //soğutucu aç isteğine cevap
{
    return Eyleyici.getObj().sogutucuAc(); //singleton pattern ile oluşturduğumuz nesne üzeriden fonksiyona erişiyoruz.
}
```

Observer deseninden, bire bir bağımlılığı durumunda bir nesnenin durumu değiştiğinde buna bağlı olan nesnelere bu değişim bildirilir ve nesneler güncellenir.

Observer *deseninin koddaki uygulanması:*

Observer, ara yüz sınıfı bir değişiklik olduğunda nesneyi uyaracaktır.

```
 */
public interface IObserver {
    public void guncelle(String m);
}
```

Subject ara yüz sınıfı ise durumu değişecek nesneyi temsil eder.

```
public interface ISubject {
    public void attach(IObserver o);
    public void detach(IObserver o);
    public void notify(String m);
}
```

Publisher(haber verici) sınıfına subject ara yüzünü implemente ederek metod gövderlini tanımladık.

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

    @Override
    public void attach(IObserver o) {
        subscribers.add(o);
    }

    @Override
    public void detach(IObserver o) {
        subscribers.remove(o);
    }

    @Override
    public void notify(String m) {
        for(IObserver subscriber : subscribers)
        {
            subscriber.guncelle(m);
        }
    }
}
```

Subscriber(abone) sınıfına observer ara yüzünü implemente ederek metod gövderlini tanımladık.

```
public class Subscriber implements IObserver{


    @Override
    public void guncelle(String m) {
        System.out.println("Admine gelen mesaj --->" + m);
    }


}
```

SıcaklıkAlgılayıcı sınıfı içerisinde sicaklikGonder() fonksiyonundaki sıcaklık değişimini buna bağlı olan nesnelere duyuruyoruz.

```
@Override
public int sicaklikGonder() {
    int sicaklik = MerkeziIslemBirimi.getObj().sicaklikGoruntule();
    publisher.notify("Ortam Sıcaklığı: " + sicaklik + " derece");
    return sicaklik;
}
```