

# Statik Değişkenler Ve Metotlar Lab Uygulaması

<https://github.com/irhallac/comp-alg/tree/master/algorithm2/java-static>

## 1. BÖLÜM

**Adım-1:** “Giris” isminde bir sınıf oluşturup konsola “merhaba dünya” yazdıralım.

```
public class Giris {  
    public static void main(String[] args) {  
        System.out.println("merhaba dünya");  
    }  
}
```

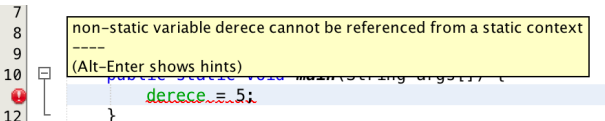
```
public static void main(String[]
```

**Adım-2:** Buradaki **main ()** metodu statik olmasaydı derleyici programı yürütmeye nasıl ve nereden başlayabilirdi?

**Cevap:** Herhangi bir Java programında **main ()** metodu derleyicinin program yürütmeye başladığı başlangıç noktasıdır. Eğer statik metot olmasaydı Java Virtual Machine ana sınıf olan Giris sınıfından bir nesne oluşturacaktı. Bu yöntem kesin ve tutarlı bir çalışma mekanizması değildir. Bir nesneden türetilmeye bağımlı olmayan, program başladığında direk yürütülebilir bir main metodu bu belirsizlikleri ortadan kaldırır.

**Adım-3:** Bu sınıf içerisinde *integer* tipinde derece isminde bir değişken tanımlayalım. Bu değişkenin değerini main () metodu içerisinde değiştirelim. Verilen hatayı inceleyelim.

```
public class Giris {  
    int derece;  
    public static void main(String args[]) {  
        derece = 5;  
    }  
}
```



derece değişkeninin erişip değişiklik yapmamız engellendi. Eğer bu değişken statik olsaydı herhangi bir nesne üretme işlemi gerekmeden doğrudan kullanılabilirdi.

**Adım-4:** main () içerisinde Giriş sınıfından yeni bir nesne türetilip derece değerini 8 yapalım. Bu şekilde *non-static variable* hatası almayız.

```
public class Giris {  
    int derece;  
    public static void main(String args[]) {  
        Giris g1 = new Giris();  
        g1.derece = 8;  
        System.out.println("Bugün hava " + g1.derece + " derece");  
    }  
}
```

Output: *Bugün hava 8 derece*

**Adım-5:** 3.Adım'daki hatayı almamanın en basit yolu bu değişkenin static olarak tanımlanmasıdır.

```
public class Giriş {
    static int derece;
    public static void main(String args[]) {
        derece = 5;
        System.out.println("Bugün hava " + derece + " derece");
    }
}
```

Output: *Bugün hava 5 derece*

**Adım-6:** Değişkenlerde olduğu gibi metotlar da static olduklarında doğrudan erişilebilir. selamGonder adında bir metot ile gelen String için konsola selam yazdıralım. Bu metodu main () içerisinde çağıralım

```
8 public class Giriş {
9
10     public void selamGonder(String isim) {
11         System.out.println("Selam " + isim);
12     }
13
14     public static void main(String args[]) {
15         selamGonder("Ali");
16     }
17 }
```

Adım-3'te değişken için verilen hata aynı nedenden dolayı bu kez metot için alınır.

```
11         System.out.println("Selam " + isim);
12     }
13     non-static method selamGonder(java.lang.String) cannot be referenced from a static context
14     (Alt-Enter shows hints)
15     public static void main(String args[]) {
16         selamGonder("Ali");
17     }
```

**Adım-7:** Alınan hatayı Adım-4'teki gibi engelleyiniz.

```
public class Giriş {
    public void selamGonder(String isim) {
        System.out.println("Selam " + isim);
    }

    public static void main(String args[]) {
        Giriş g2 = new Giriş();
        g2.selamGonder("Ali");
    }
}
```

Çözüm: Output: *Selam Ali*

**Adım-7:** Alınan hatayı Adım-5'teki gibi engelleyiniz.

```
8 public class Giriş {
9
10     public static void selamGonder(String isim) {
11         System.out.println("Selam " + isim);
12     }
13
14     public static void main(String args[]) {
15         selamGonder("Gamze");
16     }
17 }
18
```

Çözüm: Output: *Selam Gamze*

## 2. BÖLÜM

**Adım-1:** Bir mağazanın müşterilerine ait **isim**, **yaş**, **müşteri numarası** ve **yüzde** kaç indirimde sahip olduğu bilgilerini tutan bir sınıf yapısı tanımlayalım.

```
public class Musteri {
    String isim;
    int yas;
    int m_no;
    double yuzde_indirim;

    public void bilgileriYazdir(){
        //....
    }
}
```

**Adım-2:** bilgileriYazdir() metodunu bu bilgileri konsola yazdıracak şekilde tamamlayalım. 2 tane müşteri oluşturup bilgilerini girelim.

```
public class Musteri {
    String isim;
    int yas;
    int m_no;
    double yuzde_indirim;

    public void bilgileriYazdir() {
        System.out.println("Müşteri adı " + isim + " yaşı " + yas + " ve "
            + "müşteri no = " + m_no + "indirim yüzdesi = " + yuzde_indirim);
    }

    public static void main(String[] args) {
        Musteri m1 = new Musteri();
        m1.isim = "ali";
        m1.yas = 24;
        m1.m_no = 1;
        m1.yuzde_indirim = 12.5;
        m1.bilgileriYazdir();

        Musteri m2 = new Musteri();
        m2.isim = "gamze";
        m2.yas = 22;
        m2.m_no = 2;
        m2.yuzde_indirim = 12.5;
        m2.bilgileriYazdir();
    }
}
```

### Output:

Müşteri adı ali yaşı 24 ve müşteri no = 1  
indirim yüzdesi = 12.5

Müşteri adı gamze yaşı 22 ve müşteri no  
= 2 indirim yüzdesi = 12.5

**Adım-3:** Bu mağazada indirim oranı müşteriden müşteriye değişmiyorsa, yani tüm müşteriler için aynı değer girilmesi gerekiyorsa bu değişken static olarak tanımlanır. Bu şekilde her zaman son girilen değere göre işlem yapılır.

```
public class Musteri {
    String isim;
    int yas;
    int m_no;
    static double yuzde_indirim;

    public void bilgileriYazdir() {
        System.out.println("Müşteri adı " + isim + " yaşı " + yas + " ve "
            + "müşteri no = " + m_no + "indirim yüzdesi = " + yuzde_indirim);
    }

    public static void main(String[] args) {
        Musteri m1 = new Musteri();
        m1.isim = "ali";
        m1.yas = 24;
        m1.m_no = 1;
        m1.yuzde_indirim = 7.5;
        m1.bilgileriYazdir();

        Musteri m2 = new Musteri();
        m2.isim = "gamze";
        m2.yas = 22;
        m2.m_no = 2;
        m2.bilgileriYazdir();
    }
}
```

### Output:

Müşteri adı ali yaşı 24 ve müşteri no = 1  
indirim yüzdesi = 7.5

Müşteri adı gamze yaşı 22 ve müşteri no  
= 2 indirim yüzdesi = 7.5

Sadece m1 nesnesi için değer 7.5 yapıldı. Bu değer m2 nesnesinde aynı değışkene aynı değeri vermiş oldu. Static değışken nesneler arasında ortak bir değışkendir.

**Adım-4:** Yeni eklenen her müşteriye elle müşteri no değerinin girilmesi yerine bunun otomatik olarak artışı sağlayabiliriz. Bunun için müşteri no static olarak tanımlanır ve başlangıç değeri sıfır olarak verilir. Musteri sınıfının yapıcı metodunda bu değer bir arttırılırsa her yeni nesne oluşturulduğunda bu değer 1 artar.

```
public class Musteri {  
    String isim;  
    int yas;  
    static int m_no = 0;  
    static double yuzde_indirim;  
  
    public void bilgileriYazdir() {  
        System.out.println("Müşteri adı " + isim + " yaşı " + yas + " ve "  
            + "müşteri no = " + m_no + " indirim yüzdesi = " + yuzde_indirim);  
    }  
  
    public Musteri(){  
        m_no += 1;  
    }  
  
    public static void main(String[] args) {  
        Musteri.yuzde_indirim = 8.2;  
  
        Musteri m1 = new Musteri();  
        m1.isim = "ali";  
        m1.yas = 24;  
        m1.bilgileriYazdir();  
  
        Musteri m2 = new Musteri();  
        m2.isim = "gamze";  
        m2.yas = 22;  
        m2.bilgileriYazdir();  
    }  
}
```

### Output:

Müşteri adı ali yaşı 24 ve müşteri no = 1  
indirim yüzdesi = 8.2

Müşteri adı gamze yaşı 22 ve müşteri no  
= 2 indirim yüzdesi = 8.2