

# Nesne Yönelimli Programlama

*Kurucu Metotlar*

Hüseyin Ahmetođlu

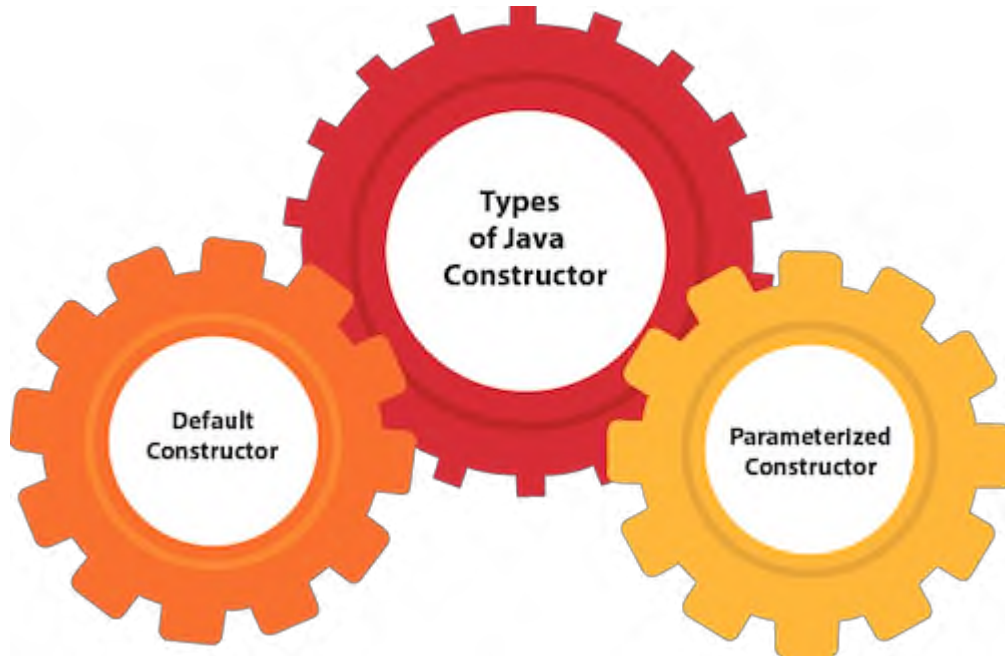
## Java'daki Yapıcılar

- ▶ Kurucu metotların diğer yöntemlere benzer kodlar bloğudur.
- ▶ Sınıfın bir örneği oluşturulduğunda kurucular çağrılır.
- ▶ Kurucu çağrılırken, nesne için hafızada bir yer ayrılır.
- ▶ Kurucular, nesneyi başlatmak için kullanılan özel bir yöntem türüdür.
- ▶ `new ()` anahtar sözcüğü kullanılarak her nesne oluşturulduğunda, en az bir kurucu çağrılır.
- ▶ Sınıfta kullanılabilir kurucu yoksa varsayılan kurucu çağrılır. Bu durumda, Java derleyicisi varsayılan olarak varsayılan bir kurucu metot sağlar.

# Java kurucu oluşturma kuralları

- ▶ Kurucu adı, sınıf adıyla aynı olmalıdır
- ▶ Bir kurucu açık bir dönüş türüne sahip olmamalıdır
- ▶ Java kurucusu abstract, static ve final olamaz.
- ▶ Not: Bir kurucu bildirirken erişim değiştiricileri kullanabiliriz . Nesne oluşturma işlemini kontrol eder. Başka bir deyişle, Java'da private, protected, public veya default kurucuya sahip olabiliriz.

# Java kurucu türleri



## Varsayılan kurucu örneği

```
//Java Program to create and call a default constructor
class Bike1{
//creating a default constructor
Bike1(){System.out.println("Bike is created");}
//main method
public static void main(String args[]){
//calling a default constructor
Bike1 b=new Bike1();
}
}
```

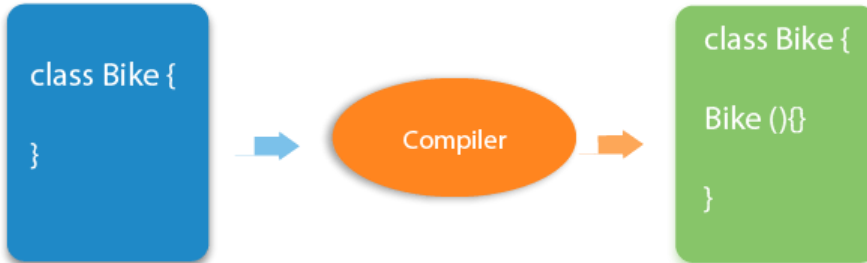
Output:

```
Bike is created
```

Kural: Bir sınıfta kurucu yoksa, derleyici otomatik olarak varsayılan bir kurucu oluşturur.

# Varsayılan kurucunun amacı nedir?

- ▶ Varsayılan kurucu, türe bağlı olarak 0, null vb. nesneye varsayılan değerleri sağlamak için kullanılır.
- ▶



# Varsayılan değerleri görüntüleyen varsayılan kurucu örneği

```
//Let us see another example of default constructor
//which displays the default values
class Student3{
int id;
String name;
//method to display the value of id and name
void display(){System.out.println(id+" "+name);}

public static void main(String args[]){
//creating objects
Student3 s1=new Student3();
Student3 s2=new Student3();
//displaying values of the object
s1.display();
s2.display();
}
}
```

Output:

```
0 null
```

```
0 null
```

## Java Parametrelili kurucu

- ▶ Belirli sayıda parametreye sahip olan bir kurucuya parametrelili kurucu denir.
- ▶ Parametrelili kurucu, farklı nesnelere farklı deęerler saęlamak için kullanılır. Ancak, aynı deęerleri de saęlayabilirsiniz.

▶




# Parametrelili kurucu örneği

//Java Program to demonstrate the use of the parameterized constructor.

```
class Student4{
    int id;
    String name;
    //creating a parameterized constructor
    Student4(int i,String n){
        id = i;
        name = n;
    }
    //method to display the values
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
        //creating objects and passing values
        Student4 s1 = new Student4(111,"Karan");
        Student4 s2 = new Student4(222,"Aryan");
        //calling method to display the values of object
        s1.display();
        s2.display();
    }
}
```

Output:



```
111 Karan
222 Aryan
```

# Java'da kurucu Aşırı Yükleme

- ▶ Java'da, bir kurucu bir yöntem gibidir ancak dönüş türü yoktur. Ayrıca Java yöntemleri gibi aşırı yüklenebilir.
- ▶ Java'da kurucu aşırı yüklenmesi, farklı parametre listelerine sahip birden fazla kurucuya sahip bir tekniktir. Her kurucunun farklı bir görevi yerine getireceği şekilde düzenlenmiştir. Derleyici tarafından listedeki parametre sayısı ve türleri ile ayırt edilirler.

```
//Java program to overload constructors
class Student5{
    int id;
    String name;
    int age;
    //creating two arg constructor
    Student5(int i,String n){
        id = i;
        name = n;
    }
    //creating three arg constructor
    Student5(int i,String n,int a){
        id = i;
        name = n;
        age=a;
    }
    void display(){System.out.println(id+" "+name+" "+age);}

    public static void main(String args[]){
        Student5 s1 = new Student5(111,"Karan");
        Student5 s2 = new Student5(222,"Aryan",25);
        s1.display();
        s2.display();
    }
}
```

Output:

```
111 Karan 0
222 Aryan 25
```

# Kurucu ile yöntem arasındaki fark

## ► KURUCU

- Bir kurucu, bir nesnenin durumunu başlatmak için kullanılır.
- Bir kurucu bir dönüş türüne sahip olmamalıdır.
- Kurucu dolaylı olarak çağrılır.
- Sınıfta herhangi bir kurucunuz yoksa Java derleyicisi varsayılan bir kurucu sağlar.
- Kurucu adı sınıf adıyla aynı olmalıdır.

## ► METOT

- Bir nesnenin davranışını ortaya çıkarmak için bir metot kullanılır.
- Bir metodun bir dönüş türü olmalıdır.
- Metot doğrudan çağrılır.
- Metot hiçbir durumda derleyici tarafından sağlanmaz.
- Metot adı, sınıf adıyla aynı olabilir veya olmayabilir.

# Kopya Kurucu

- ▶ Java'da bir nesne için kopya kurucu yoktur. Ancak, değerleri bir nesneden diğerine C ++ 'daki kopya kurucuya kopyalayabiliriz.
- ▶ Java'da bir nesnenin değerlerini diğerine kopyalamanın birçok yolu vardır.
  - Kurucu tarafından
  - Bir nesnenin değerlerini başka bir nesneye atayarak
  - Object sınıfının clone () yöntemine göre

# Kopya Oluşturucu

Output:

```
111 Karan
```

```
111 Karan
```

//Java program to initialize the values from one object to another object.

```
class Student6{
    int id;
    String name;
    //constructor to initialize integer and string
    Student6(int i,String n){
        id = i;
        name = n;
    }
    //constructor to initialize another object
    Student6(Student6 s){
        id = s.id;
        name =s.name;
    }
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
        Student6 s1 = new Student6(111,"Karan");
        Student6 s2 = new Student6(s1);
        s1.display();
        s2.display();
    }
}
```

# Kurucu olmadan değerleri kopyalama

```
class Student7{
    int id;
    String name;
    Student7(int i,String n){
        id = i;
        name = n;
    }
    Student7(){}
```

```
void display(){System.out.println(id+" "+name);}

public static void main(String args[]){
    Student7 s1 = new Student7(111,"Karan");
    Student7 s2 = new Student7();
    s2.id=s1.id;
    s2.name=s1.name;
    s1.display();
    s2.display();
}
}
```

Output:

```
111 Karan
111 Karan
```

# Notlar

- ▶ Kurucu herhangi bir değer döndürüyor mu?
  - Evet, geçerli sınıf örneğidir (dönüş türünü kullanamazsınız, ancak bir değer döndürür).
- ▶ Kurucu, başlatma yerine başka görevleri gerçekleştirebilir mi?
  - Evet, nesne oluşturma, bir iş parçacığı başlatma, bir yöntem çağırma vb. Gibi. Yöntemde yaptığınız gibi yapıcıda herhangi bir işlemi gerçekleştirebilirsiniz.
- ▶ Java'da Constructor sınıfı var mı?
  - Evet
- ▶ Constructor sınıfının amacı nedir?
  - Java, sınıftaki bir kurucunun iç bilgilerini almak için kullanılacak bir Kurucu sınıfı sağlar. `Java.lang.reflect` paketinde bulunur.



## KAYNAKLAR

- ▶ Java Tutorial | Learn Java - javatpoint. (2021, March 21). Retrieved from <https://www.javatpoint.com/java-tutorial>