Kurucu Metotlar (Constructors)

- Kurucu metot çeşitleri
- Varsayılan kurucu metot
- Parametreli kurucu metot
- Kurucu metot overloading
- Kurucu metot geriye değer döndürür mü?
- Bir nesnenin değerlerini diğerine kopyalamak
- Yapıcı metoda ilk değer atamanın dışında başka işler yükleme

Kurucu Metotlar (Constructors)

- Nesne ilk olarak yapıldığı anda çalıştırılan özel bir nesne kurucu metottur.
- İki kural tanımlıdır
  - Kurucu ismi sınıf ismi ile aynı olmalıdır.
  - Kurucu metot dışarıya bir değer döndürmemelidir.
- Türleri
  - Varsayılan (parametresiz)
  - Parametreli

Kurucu Metotlar (Constructors)

- Java'da varsayılan kurucu metodun syntax yapısı
- <class_name>(){}
- Örnek

```
class Student{
        Student(){System.out.println(‘’Student is generated");}
        public static void main(String args[]){
                Student b=new Student();
        }
}
```

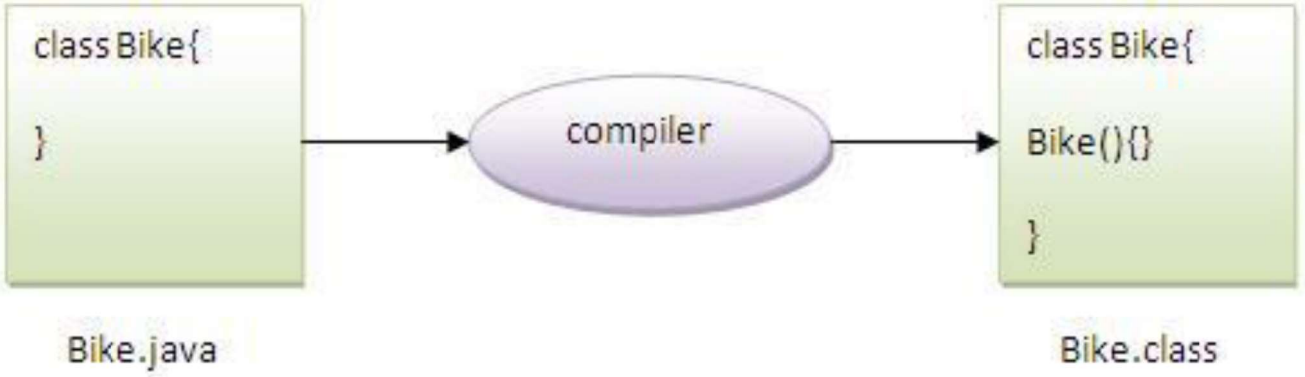
Test it Now

Output:

Student is created

Kurucu Metotlar (Constructors)

- Kural: Eğer bir sınıfta açık olarak yapıcı metot tanımlanmamışsa derleyici otomatik olarak varsayılan kurucu metodu sınıfa ekler.



Kurucu Metotlar (Constructors)

```
class Student {
        int id;
        String name;
        void show(){System.out.println(id+" "+name);}
        public static void main(String args[]){
                Student s1=new Student ();
                Student s2=new Student ();
                s1.show();
                s2.show();
        }
}
```

Test it Now

Output:

0 null

0 null

Yandaki örnekte Student sınıfında default Constructor devreye girerek id değişkenine 0, name değişkenine de null değerini atamıştır.

C dilindeki her değişkene mutlaka ilk değer Verilmesi bir gelenektir. Java'da yapıcı metot bu Sorumluluğu üstlendiğinden gerek kalmamaktadır.

Parametreli Kurucu Metotlar (Parameterized Constructors)

- Default constructor ile nesne yapıldıktan sonra sınıf değişkenlerine ulaşıp değerlerini değiştirmemiz mümkündür.
- Bu özellik ancak public veya protected tanımlanmış değişkenler için mümkündür. Private tanımlı değişkenlere sınıf içinde erişim mümkündür. Sınıf dışında erişelemezler.
- Değişkenlerin değerlerini nesne yapıldıktan sonra değiştirmek eğer devamlı aynı kural çerçevesinde olacaksa yazılımcıya ekstra külfet getirir.
- Bu problemi aşmak için nesne yapıldığı anda değerler kurucu metoda parametre olarak gönderilebilir.

```
class Student {
    int id;
    String name;

    Student (int i,String n){
      id = i;
      name = n;
    }

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

    public static void main(String args[]){
      Student s1 = new Student (123,"Ahmet");
      Student s2 = new Student (456,"Mehmet");
      s1.display();
      s2.display();
    }
}
```

Test it Now

Output:

```
111 Karan
222 Aryan
```

Constructor overloading

- Birden fazla farklı parametreler alan kurucu metotlar tanımlamanız mümkündür.

```
class Student {
    int id;
    String name;
    int age;

    Student(int i,String n){
       id = i;
       name = n;
    }
    Student (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[]){
    Student s1 = new Student (111,"Ahmet");
    Student s2 = new Student (222,"Mehmet",25);
    s1.display();
    s2.display();
   }
}
```

Constructor vs Method in JAVA

Java Constructor	Java Method
Constructor is used to initialize the state of an object.	Method is used to expose behaviour of an object.
Constructor must not have return type.	Method must have return type.
Constructor is invoked implicitly.	Method is invoked explicitly.
The java compiler provides a default constructor if you don't have any constructor.	Method is not provided by compiler in any case.
Constructor name must be same as the class name.	Method name may or may not be same as class name.

Copy Constructor

- Bir nesneyi diğer bir nesneye kopyalamanın çeşitli yolları vardır. Bunlar
  - Constructor üzerinden
  - Bir nesnenin değerlerini diğer nesneye atamak şeklinde
  - Clone metodu ile

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

    Student (Student s){
      id = s.id; name=s.name;
    }
```

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


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

Constructor olmadan kopyalama yapmak

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

    Student (){}


    void display(){System.out.println(id+" "+name
    );}
   public static void main(String args[]){
       Student s1 = new Student (111,"Ahmet");
       Student s2 = new Student7();
       s2.id=s1.id;
       s2.name=s1.name;
       s1.display();
       s2.display();
     }
   }
```

Sorular

- Kurucu metodlar geriye değer döndürür mü?
- Nesne başlatmanın dışında constructorlar ne iş yaparlar?

Cevaplar

- Kurucu metodlar geriye değer döndürür mü?
  - Evet, kendi sınıf örneğini varsayılan olarak geri döndürür. Bunun için return demenize gerek yoktur.
- Nesne başlatmanın dışında constructorlar ne iş yaparlar?
  - Nesneyi heapte açma
  - Thread başlatma
  - Metot çağırma

Statik

- Statik değişken
- Statik değişkensiz sayaç programı
- Statik değişkenli sayaç programı
- Statik metot
- Statik metot kısıtlamaları
- Main metot neden statik?
- Statik blok
- Main metot olmadan bir program yürütülebilir mi?

Statik

- Java'daki statik deyimi ağırlıklı olarak bellek yönetimi için kullanılır. Java static anahtar kelimesini değişkenler, yöntemler, bloklar ve iç içe geçmiş sınıflarla uygulayabiliriz.
- Statik anahtar kelime, sınıfın örneğinden ziyade sınıfa aittir.
- Statik şunlardan biri olabilir:

1) değişken(ayrıca sınıf değişkeni olarak da bilinir)
2) Metot(ayrıca sınıf metodu olarak da bilinir)
3) Blok
4) İç içe geçmiş sınıflar

Java statik değişken

- Herhangi bir değişkeni statik değişken olarak tanımlarsan bu statik değişken olarak bilinir.
- Statik değişken, tüm nesnelerin ortak özelliğini (diğer bir deyişle her nesne için farklı olmayan) yönlendirmek için kullanılabilir.
- Statik değişken, sınıf yükleme esnasında sınıf alanında yalnızca bir kez bellek rezerv eder.

Statik değişkenin avantajları

- Programınızın belleğini verimli hale getirir (yani bellekten tasarruf sağlar)
- Statik değişkensiz problemi anlama

```
class Student{
    int rollno;
    String name;
    String college="ITS";
}
```

Örnek

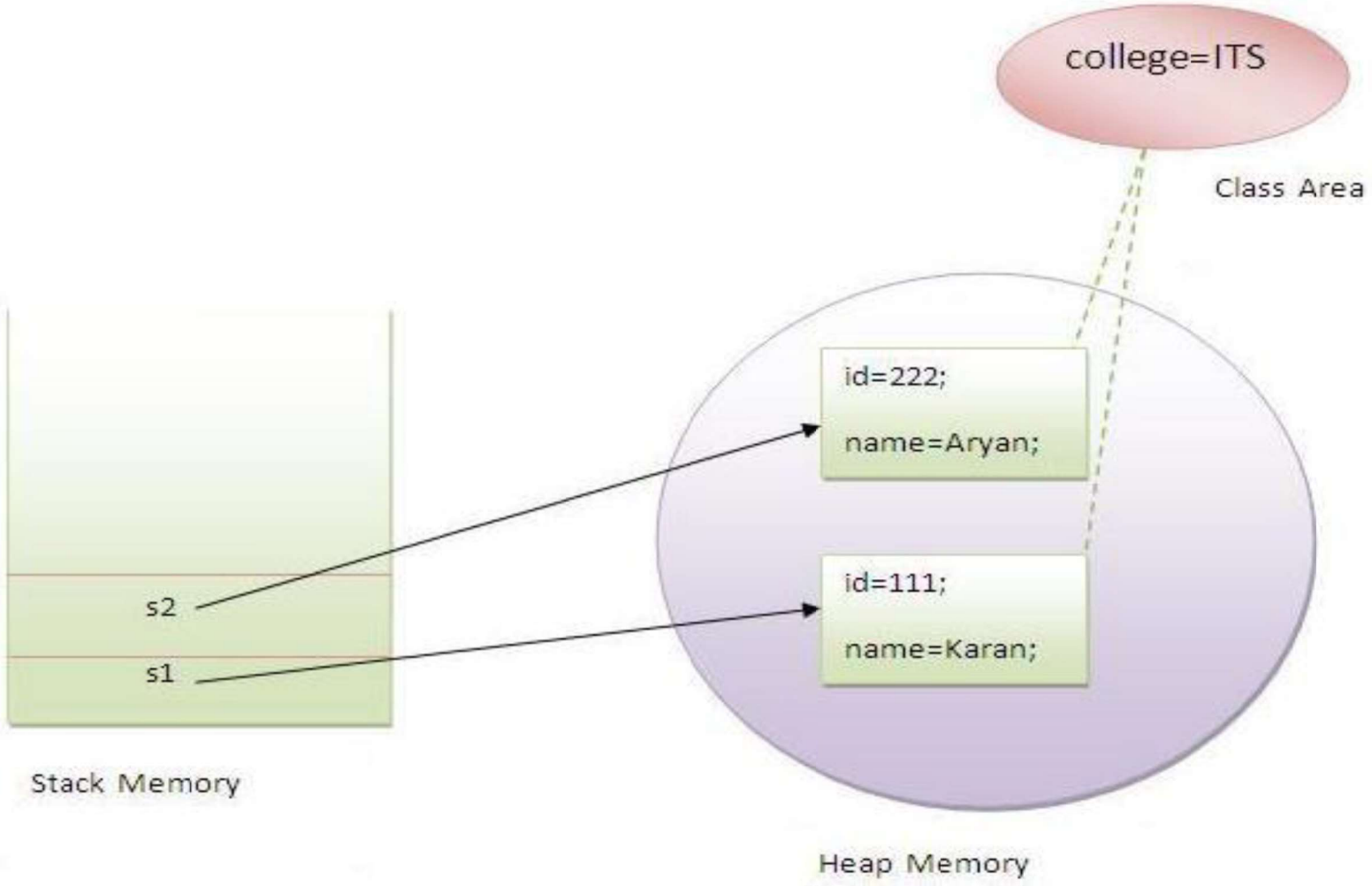
- Üniversitede 500 öğrenci var olduğunu varsayalım , tüm örnek veri üyeleri nesne oluşturulduğunda aynı anda bellekte yer kaplayacaktır. Tüm öğrencilerin farklı rolü ve adı var bunlar statik olmaması iyi olmuş.
- Ancak, burada, kolej, tüm nesnelerin ortak özelliğini ifade eder. Bu alanı statik hale getirirsek, bu alan yalnızca bir kez hafızaya alınacaktır.
- Java static değişkenleri o sınıftan üretilmiş tüm nesneler için ortak kullanılır.

```
class Student8{
  int rollno;
  String name;
  static String college ="ITS";

  Student8(int r,String n){
    rollno = r;
    name = n;
  }
   void display (){System.out.println(rollno+" "+name+" "+college);}

  public static void main(String args[]){
    Student8 s1 = new Student8(111,"Karan");
    Student8 s2 = new Student8(222,"Aryan");
    s1.display();
    s2.display();
  }
}
```

Test it Now

Output:111 Karan ITS

222 Aryan ITS


college=ITS
Class Area
id=222;
name=Aryan;
id=111;
name=Karan;
s2
s1
Stack Memory
Heap Memory

Statik değişkensiz sayaç programı

- Biz bu örnekte, sayacın yapısını artıran bir örnek değişkeni oluşturduk. Örnek değişkeni, nesne oluşturma zamanında belleği aldığından, her nesne örnek değişkeninin kopyasına sahip olacak, eğer bu artarsa, diğer nesnelere yansıtılmayacaktır. Böylece her nesnenin sayaç değişkeninde 1 değeri olacaktır.

```
class Counter{
   int count=0;//will get memory when instance is created

   Counter(){
      count++;
      System.out.println(count);
   }

   Public static void main(String args[]){

   Counter c1=new Counter();
   Counter c2=new Counter();
   Counter c3=new Counter();
 }
}
```

Test it Now

Output:1

1

1

Sayıcının statik değişkene göre programlanması

- Yukarıda belirttiğimiz gibi, statik değişken hafızada bir yerde tutulur, herhangi bir nesne statik değişkenin değerini değiştirirse, ilgili değişkenin değerini korunur.

```
class Counter2{
static int count=0;//will get memory only once and retain its value

Counter2(){
  count++;
  System.out.println(count);
}

public static void main(String args[]){
  Counter2 c1=new Counter2();
  Counter2 c2=new Counter2();
  Counter2 c3=new Counter2();
 }
}
```

2) Java statik metot

- Herhangi bir metot ile statik anahtar kelime uygularsanız bu statik metot olarak bilinir.
- Statik metot nesneden ziyade sınıfa aittir.
- Statik metot nesne inşa edilmeden de doğrudan Sınıf üzerinden çağrılabilir. Mesela: Math.abs() metodu gibi.
- Statik metod sınıfta tanımlı statik değişkenlere ulaşabilir ve değerlerini değiştirebilir.

```
class Student9{
    int rollno;
    String name;
    static String college = "ITS";

    static void change(){
    college = "BBDIT";
    }

    Student9(int r, String n){
    rollno = r;
    name = n;
    }

    void  display  (){System.out.println(rollno+"  "+na
    me+" "+college);}

    public static void main(String args[]){
    Student9.change();

    Student9 s1 = new Student9 (111,"Karan");
    Student9 s2 = new Student9 (222,"Aryan");
    Student9 s3 = new Student9 (333,"Sonoo");

    s1.display();
    s2.display();
    s3.display();
    }
}
```

```
class Calculate{
  static int cube(int x){
     return x*x*x;
  }

  public static void main(String args[]){
     int result=Calculate.cube(5);
     System.out.println(result);
  }
}
```

Test it Now

```
Output:125
```

Statik metot üzerindeki kısıtlamalar

- Statik metot, statik olmayan bir değişkeni ve metodu doğrudan çağıramaz.
- this ve süper anahtar kelimeleri statik metotların içerisinde kullanılmazlar.

```
class A{
 int a=40;//non static

 public static void main(String args[]){
  System.out.println(a);
 }
}
```

Test it Now

Output:Compile Time Error

Soru

- Statik metot neden kullanılır?
- Neden java’da main metodu statiktir?
- Main metodu olmadan bir sınıf çalıştırılabilir mi?

Cevap

- Belleği daha verimli ve iktisatlı kullanmak için. Çünkü JVM nesne yapmaya gerek duymadan sınıf üzerinden main metodunu çağırarak daha az sayıda nesne yapmış olabilir.
- JDK 1.7 den önceki sürümlerde statik bloklar üzerinden gerçekleştirilebilir.

```
class A3{
  static{
    System.out.println("static block is invoked");
    System.exit(0);
  }
}
```

Statik Blok

- Sınıf içerisinde tanımladığınız bir bölgeyi statik olarak belirleyebilirsiniz. Bu durumda nesne yapılmadan önce bu kısımlar bir kez yapılır.
- Bu alanda statik değişkenler tanımlayabilirsiniz.
- Sınıf yüklendiğinde main metodundan önce ele alınırlar.

```
class A2{
  static{System.out.println("static block is invoked
    ");}
  public static void main(String args[]){
   System.out.println("Hello main");
  }
}
```

Test it Now

```
Output:static block is invoked
       Hello main
```