Nesne Yönelimli Programlama

Static ve This Anahtar Kelimeleri

Hüseyin Ahmetoğlu

statik anahtar kelimesi

- ***Static* anahtar kelimesi** temel olarak bellek yönetimi için kullanılır.
- *Static* ile oluşturulmuş bir nesne, sınıfın bir örneğinden çok sınıfa aittir.
- Neler *static* olabilir?
  - Değişken (sınıf değişkeni)
  - Yöntem (sınıf yöntemi)
  - Blok
  - İç içe sınıf

static değişken

- **Static** değişken, tüm nesnelerin (her bir nesne için benzersiz olmayan) ortak özelliklerine atıfta bulunmak için kullanılabilir.
- **Static** değişken, sınıfın yüklenmesi sırasında sınıf alanında yalnızca bir kez bellek alır.

Statik değişkenin avantajları

- Program **belleğinizi verimli** hale getirir (yani, bellek tasarrufu sağlar).
- **Java bir static özelliği tüm nesnelerle paylaşılır.**
- Üniversitede 500 öğrenci olduğunu varsayalım, şimdi tüm örnek veri üyeleri nesne her oluşturulduğunda bellek alacak.
- Tüm öğrenciler için benzersiz *rollno* ve *name* vardır, bu nedenle örnek veri üyesi bu durumda iyidir.
- Burada *college*, tüm nesnelerin ortak alanı anlamına gelir. *Static* yaparsak, bu alan belleği sadece bir kez alır.

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

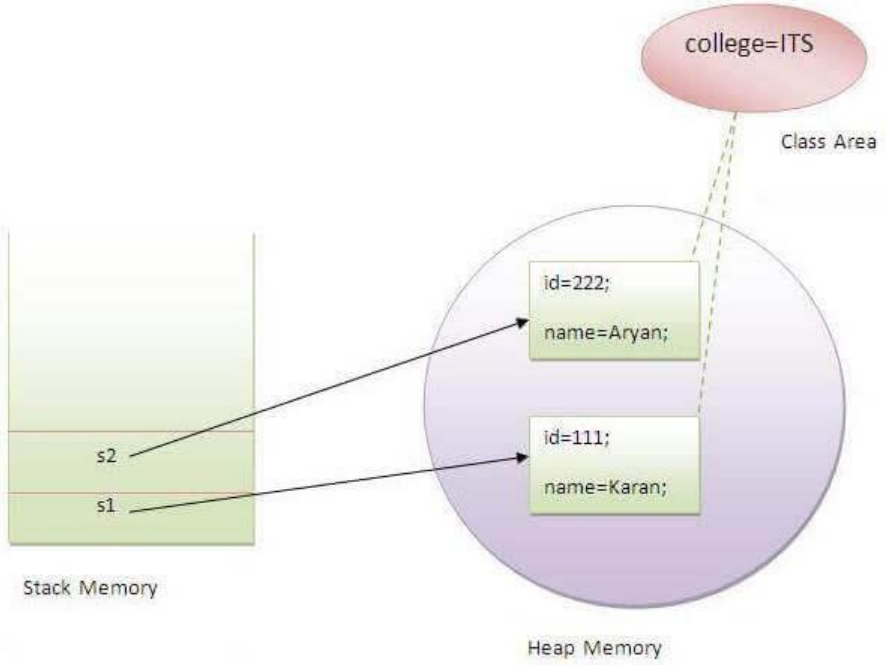
Static değişken örneği

```
//Java Program to demonstrate the use of static variable
class Student{
   int rollno;//instance variable
   String name;
   static String college ="ITS";//static variable
   //constructor
   Student(int r, String n){
   rollno = r;
   name = n;
   }
   //method to display the values
   void display (){System.out.println(rollno+" "+name+" "+college);}
}
//Test class to show the values of objects
public class TestStaticVariable1{
 public static void main(String args[]){
 Student s1 = new Student(111,"Karan");
 Student s2 = new Student(222,"Aryan");
 //we can change the college of all objects by the single line of code
 //Student.college="BBDIT";
 s1.display();
 s2.display();
 }
}
```

Output:

```
111 Karan ITS
222 Aryan ITS
```

Statik değişken örneği



Statik değişkensiz sayaç programı

```
//Java Program to demonstrate the use of an instance variable
//which get memory each time when we create an object of the class.
class Counter{
int count=0;//will get memory each time when the instance is created

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

public static void main(String args[]){
//Creating objects
Counter c1=new Counter();
Counter c2=new Counter();
Counter c3=new Counter();
}
}
```

Output:

```
1
1
1
```

Statik değişken ile sayaç programı

```
//Java Program to illustrate the use of static variable which
//is shared with all objects.
class Counter2{
static int count=0;//will get memory only once and retain its value

Counter2(){
count++;//incrementing the value of static variable
System.out.println(count);
}

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

Output:

```
1
2
3
```

Java static metot

- *Static* bir yöntem, bir sınıfın nesnesi yerine sınıfa aittir.
- Bir sınıf örneği oluşturmaya gerek kalmadan *Static* bir yöntem çağrılabilir.
- *Static* bir yöntem *Static* veri üyesine erişebilir ve değerini değiştirebilir.

Statik yöntem örneği

```
//Java Program to demonstrate the use of a static method.
class Student{
    int rollno;
    String name;
    static String college = "ITS";
    //static method to change the value of static variable
    static void change(){
    college = "BBDIT";
    }
    //constructor to initialize the variable
    Student(int r, String n){
    rollno = r;
    name = n;
    }
    //method to display values
    void display(){System.out.println(rollno+" "+name+" "+college);}
}
```

```
//Test class to create and display the values of object
public class TestStaticMethod{
    public static void main(String args[]){
    Student.change();//calling change method
    //creating objects
    Student s1 = new Student(111,"Karan");
    Student s2 = new Student(222,"Aryan");
    Student s3 = new Student(333,"Sonoo");
    //calling display method
    s1.display();
    s2.display();
    s3.display();
    }
}
```

```
Output:111 Karan BBDIT
       222 Aryan BBDIT
       333 Sonoo BBDIT
```

Normal bir hesaplama yapan statik bir yöntem örneği

```
//Java Program to get the cube of a given number using the static method

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);
  }
}
```

```
Output:125
```

Statik yöntem için kısıtlamalar

- Statik yöntem için iki ana kısıtlama vardır.
  - Statik yöntem statik olmayan veri üyesini kullanamaz veya statik olmayan yöntemi doğrudan çağıramaz.
  - this ve super statik bağlamda kullanılamaz.

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

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

```
Output:Compile Time Error
```

Java ana yöntemi neden statik?

- Nesnelerin statik bir yöntem çağırması gerekmediği için.
- Statik olmayan bir yöntem olsaydı, JVM önce bir nesne oluştururdu ve ardından ekstra bellek ayırma sorununa yol açacak main () yöntemini çağırırdı.

Java statik bloğu

- Statik veri elemanını başlatmak için kullanılır.
- Sınıf yükleme sırasında ana yöntemden önce yürütülür.

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

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

main () yöntemi olmadan bir program yürütebilir miyiz?

- Hayır!
- Daha önceden bunun yollarından biri statik bloktu, ancak bu durum JDK 1.6'ya kadar mümkün oldu.
- JDK 1.7'den bu yana, ana yöntem olmadan bir Java sınıfı yürütmek mümkün değildir.

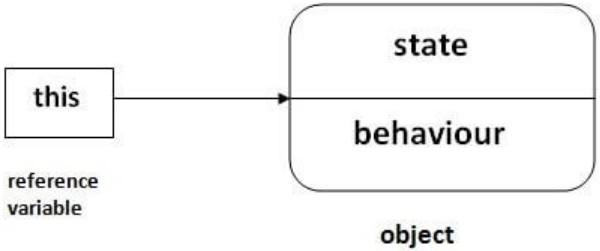
```
class A3{
  static{
  System.out.println("static block is invoked");
  System.exit(0);
  }
}
```

Since JDK 1.7 and above, output would be:

```
Error: Main method not found in class A3, please define the main method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
```

this anahtar kelimesi

- Java'da this, geçerli nesneye **başvuran** bir **referans değişkendir.**



this anahtar kelimesinin kullanımı

1. this, geçerli sınıf ortamı değişkenini belirtmek için kullanılabilir.
2. this, geçerli sınıf yöntemini çağırmak için kullanılabilir (örtük olarak)
3. this(), geçerli sınıf yapıcısını çağırmak için kullanılabilir.
4. this, yöntem çağrısında bağımsız değişken olarak geçirilebilir.
5. this, yapıcı çağrısında argüman olarak iletilebilir.
6. this, geçerli sınıf örneğini yöntemden döndürmek için kullanılabilir.

1) this: geçerli sınıf örneği değişkenini ifade etmek için

- Değişkenler ve parametreler arasında belirsizlik varsa, *this* anahtar kelime belirsizlik sorununu çözer.

```
class Student{
int rollno;
String name;
float fee;
Student(int rollno,String name,float fee){
rollno=rollno;
name=name;
fee=fee;
}
void display(){System.out.println(rollno+" "+name+" "+fee);}
}
class TestThis1{
public static void main(String args[]){
Student s1=new Student(111,"ankit",5000f);
Student s2=new Student(112,"sumit",6000f);
s1.display();
s2.display();
}}
```

```
class Student{
int rollno;
String name;
float fee;
Student(int rollno,String name,float fee){
this.rollno=rollno;
this.name=name;
this.fee=fee;
}
void display(){System.out.println(rollno+" "+name+" "+fee);}
}

class TestThis2{
public static void main(String args[]){
Student s1=new Student(111,"ankit",5000f);
Student s2=new Student(112,"sumit",6000f);
s1.display();
s2.display();
}}
```

this anahtar kelimesinin gerekli olmadığı program

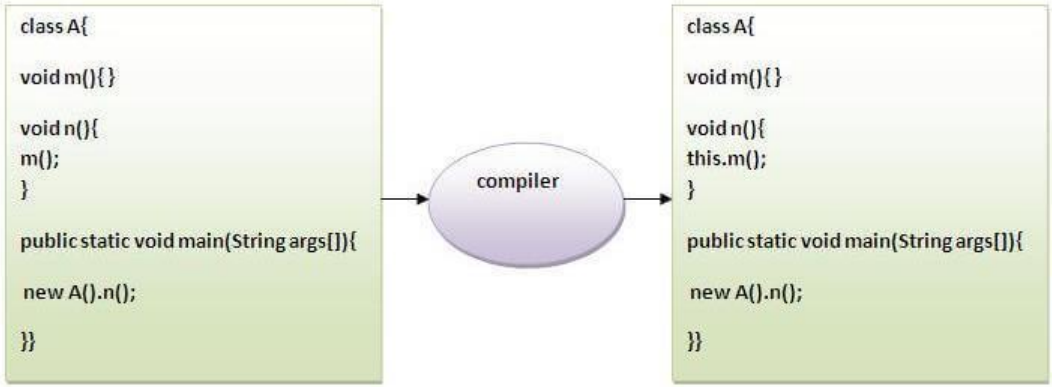
- Değişkenler için anlamlı isimler kullanmak daha iyi bir yaklaşımdır. Bu nedenle, gerçek zamanlı olarak örnek değişkenleri ve parametreleri için aynı adı kullanırız ve her zaman this anahtar kelimesini kullanırız.

```
class Student{
int rollno;
String name;
float fee;
Student(int r,String n,float f){
rollno=r;
name=n;
fee=f;
}
void display(){System.out.println(rollno+" "+name+" "+fee);}
}

class TestThis3{
public static void main(String args[]){
Student s1=new Student(111,"ankit",5000f);
Student s2=new Student(112,"sumit",6000f);
s1.display();
s2.display();
}}
```

2) this: geçerli sınıf yöntemini çağırmak için

- this anahtar kelimesi kullanarak geçerli sınıfın yöntemini çağırabilirsiniz.
- this anahtar kelimesi kullanılmazsa, derleyici yöntemi çağırırken this anahtar kelimesini otomatik olarak ekler.



Örnek

```
class A{
void m(){System.out.println("hello m");}
void n(){
System.out.println("hello n");
//m();//same as this.m()
this.m();
}
}
class TestThis4{
public static void main(String args[]){
A a=new A();
a.n();
}}
```

Output:

```
hello n
hello m
```

3) this (): geçerli sınıf kurucusunu çağırmak için

- this() kurucu çağrısı, geçerli sınıf kurucusunu çağırmak için kullanılabilir. Kurucuyu yeniden kullanmak için kullanılır. Başka bir deyişle, kurucu zincirlemesi için kullanılır.

```
class A{
A(){System.out.println("hello a");}
A(int x){
this();
System.out.println(x);
}
}
class TestThis5{
public static void main(String args[]){
A a=new A(10);
}}
```

Output:

```
hello a
10
```

```
class A{
A(){
this(5);
System.out.println("hello a");
}
A(int x){
System.out.println(x);
}
}
class TestThis6{
public static void main(String args[]){
A a=new A();
}}
```

Output:

```
5
hello a
```

this() kurucu çağrısının gerçek kullanımı

- this() yapıcı çağrısı, yapıcıyı yapıcı içerisinde yeniden kullanmak için kullanılmalıdır. Yapıcılar arasındaki zinciri korur, yani yapıcı zincirleme için kullanılır.

```
class Student{
int rollno;
String name,course;
float fee;
Student(int rollno,String name,String course){
this.rollno=rollno;
this.name=name;
this.course=course;
}
Student(int rollno,String name,String course,float fee){
this(rollno,name,course);//reusing constructor
this.fee=fee;
}
void display(){System.out.println(rollno+" "+name+" "+course+" "+fee);}
}
class TestThis7{
public static void main(String args[]){
Student s1=new Student(111,"ankit","java");
Student s2=new Student(112,"sumit","java",6000f);
s1.display();
s2.display();
}}
```

Output:

```
111 ankit java null
112 sumit java 6000
```

this() öğesine yapılan çağrı kurucucudaki ilk ifade olmalıdır.

```
class Student{
int rollno;
String name,course;
float fee;
Student(int rollno,String name,String course){
this.rollno=rollno;
this.name=name;
this.course=course;
}
Student(int rollno,String name,String course,float fee){
this.fee=fee;
this(rollno,name,course);//C.T.Error
}
void display(){System.out.println(rollno+" "+name+" "+course+" "+fee);}
}
class TestThis8{
public static void main(String args[]){
Student s1=new Student(111,"ankit","java");
Student s2=new Student(112,"sumit","java",6000f);
s1.display();
s2.display();
}}
```

```
Compile Time Error: Call to this must be first statement in constructor
```

4) this: yöntemde bir argüman olarak verilebilir

- this, yöntemde bağımsız değişken olarak da geçirilebilir. Esas olarak olay işlemede kullanılır. Bir nesneyi birçok yöntemde yeniden kullanmak için kullanılır.

```
class S2{
  void m(S2 obj){
  System.out.println("method is invoked");
  }
  void p(){
  m(this);
  }
  public static void main(String args[]){
  S2 s1 = new S2();
  s1.p();
  }
}
```

Output:

```
method is invoked
```

5) this: kurucu çağrısında argüman olarak verilebilir

- this anahtar kelimesini kurucuya aktarabiliriz. Bir nesneyi birden fazla sınıfta kullanmamız gerektiğinde faydalıdır.

```
Output:10
```

```
class B{
  A4 obj;
  B(A4 obj){
    this.obj=obj;
  }
  void display(){
    System.out.println(obj.data);//using data member of A4 class
  }
}

class A4{
  int data=10;
  A4(){
   B b=new B(this);
   b.display();
  }
  public static void main(String args[]){
   A4 a=new A4();
  }
}
```

6) mevcut sınıf örneğini döndürmek için

- this anahtar kelimesini yöntemden bir ifade olarak döndürebiliriz.
- Bu durumda, yöntemin dönüş tipi sınıf tipi olmalıdır (ilkel olmayan).

```
class A{
A getA(){
return this;
}
void msg(){System.out.println("Hello java");}
}
class Test1{
public static void main(String args[]){
new A().getA().msg();
}
}
```

Output:

```
Hello java
```

```
class A5{
void m(){
System.out.println(this);//prints same reference ID
}
public static void main(String args[]){
A5 obj=new A5();
System.out.println(obj);//prints the reference ID
obj.m();
}
}
```

Output:

```
A5@22b3ea59
A5@22b3ea59
```

KAYNAKLAR

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