Nesne Yönelimli Programlama

Super, Final ve Instance initializer

Hüseyin Ahmetoğlu

Java'da *super* Anahtar Kelimesi

- Üst sınıf nesnesine başvurmak için kullanılan bir başvuru değişkenidir.
- Alt sınıf örneğini her oluşturduğunuzda, aslında ***super*** başvuru değişkeni tarafından başvurulan bir üst sınıf örneği de oluşturulur.

Java süper Anahtar Kelimesi kullanımı

- super, üst sınıf örneği değişkenine başvurmak için kullanılabilir.
- super, üst ebeveyn sınıf metodunu çağırmak için kullanılabilir.
- super () üst sınıf yapıcısını çağırmak için kullanılabilir.

1) super, üst sınıf örnek değişkenini belirtmek için kullanılır.

- Üst sınıfın veri üyesine veya alanına erişmek için süper anahtar kelimesini kullanabiliriz.
- Üst sınıf ve alt sınıf aynı alanlara sahipse kullanılır.

```
class Animal{
String color="white";
}
class Dog extends Animal{
String color="black";
void printColor(){
System.out.println(color);//prints color of Dog class
System.out.println(super.color);//prints color of Animal class
}
}
class TestSuper1{
public static void main(String args[]){
Dog d=new Dog();
d.printColor();
}}
```

Output:

```
black
white
```

2) süper, ebeveyn sınıfı metodunu çağırmak için kullanılabilir

- Üst sınıf metodunu çağırmak için de kullanılabilir.
- Alt sınıf, üst sınıfla aynı metodu içeriyorsa kullanılmalıdır.
- Başka bir deyişle, metot geçersiz kılınmışsa kullanılır.
- Çünkü öncelik yerel olana verilir

2) super ebeveyn sınıfı metodunu çağırmak için kullanılabilir

```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void eat(){System.out.println("eating bread...");}
void bark(){System.out.println("barking...");}
void work(){
super.eat();
bark();
}
}
class TestSuper2{
public static void main(String args[]){
Dog d=new Dog();
d.work();
}}
```

Output:

```
eating...
barking...
```

3) super, üst sınıf yapıcısını çağırmak için kullanılır.

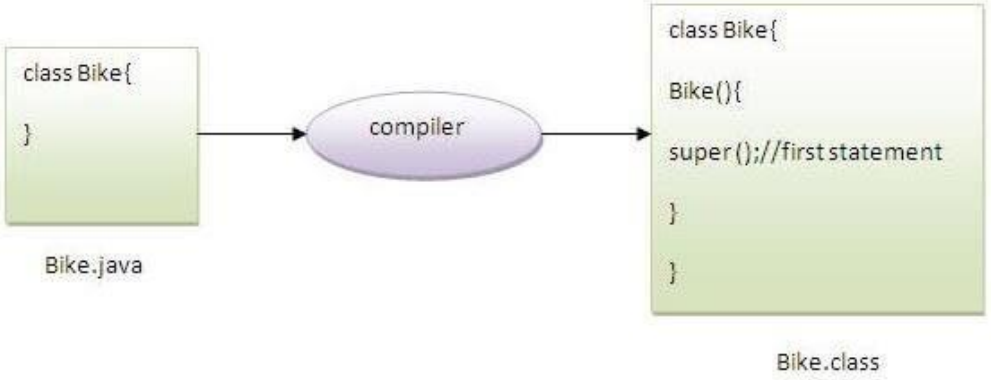
```
class Animal{
Animal(){System.out.println("animal is created");}
}
class Dog extends Animal{
Dog(){
super();
System.out.println("dog is created");
}
}
class TestSuper3{
public static void main(String args[]){
Dog d=new Dog();
}}
```

Output:

```
animal is created
dog is created
```

Not:

- **super () veya this () yoksa, super () her sınıf yapıcısına derleyici tarafından otomatik olarak eklenir.**
- Bildiğimiz gibi varsayılan kurucu, kurucu yoksa derleyici tarafından otomatik olarak sağlanır. Ancak, ilk ifade olarak super () öğesini de ekler.



Örnek:

```
class Animal{
Animal(){System.out.println("animal is created");}
}
class Dog extends Animal{
Dog(){
System.out.println("dog is created");
}
}
class TestSuper4{
public static void main(String args[]){
Dog d=new Dog();
}}
```

Output:

```
animal is created
dog is created
```

super örnek: gerçek kullanım

```
class Person{
int id;
String name;
Person(int id,String name){
this.id=id;
this.name=name;
}
}
class Emp extends Person{
float salary;
Emp(int id,String name,float salary){
super(id,name);//reusing parent constructor
this.salary=salary;
}
void display(){System.out.println(id+" "+name+" "+salary);}
}
class TestSuper5{
public static void main(String[] args){
Emp e1=new Emp(1,"ankit",45000f);
e1.display();
}}
```

Output:

```
1 ankit 45000
```

Örnek başlatıcı bloğu (Instance initializer block)

- **Örnek** Başlatıcı **bloğu** , örnek veri üyesini başlatmak için kullanılır. Sınıfın nesnesi her oluşturulduğunda çalışır.
- Örnek değişkeninin başlatılması doğrudan yapılabilir, ancak örnek başlatıcı bloğunda örnek değişkeni başlatılırken fazladan işlemler gerçekleştirilebilir.

Neden örnek başlatıcı bloğu kullanılmalı?

- Örnek veri üyesine değer atarken bazı işlemler yapmak zorunda olduğumuzu varsayalım, örneğin karmaşık bir diziyi doldurmak için bir for döngüsü veya hata işleme vb.

```
class Bike7{
   int speed;

   Bike7(){System.out.println("speed is "+speed);}

   {speed=100;}

   public static void main(String args[]){
   Bike7 b1=new Bike7();
   Bike7 b2=new Bike7();
   }
}
```

```
Output:speed is 100
       speed is 100
```

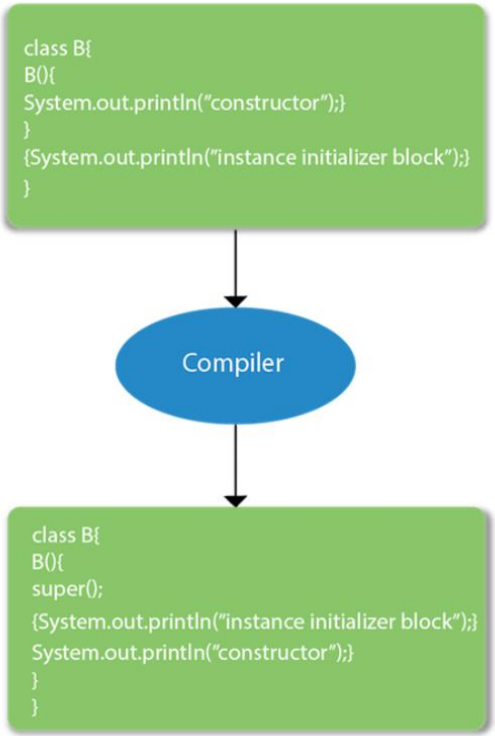
İlk önce hangisi başlatılır, örnek başlatıcı bloğu mu kurucu mu?

```
class Bike8{
   int speed;

   Bike8(){System.out.println("constructor is invoked");}

   {System.out.println("instance initializer block invoked");}

   public static void main(String args[]){
   Bike8 b1=new Bike8();
   Bike8 b2=new Bike8();
   }
}
```

```
Output:instance initializer block invoked
       constructor is invoked
       instance initializer block invoked
       constructor is invoked
```

Java derleyicisi, her yapıcıdaki örnek başlatıcı bloğunun kodunu kopyalar.

- Örnek başlatıcı bloğu, sınıfın örneği oluşturulduğunda oluşturulur.
- Örnek başlatıcı bloğu, üst sınıf yapıcısı çağrıldıktan sonra (yani, super () yapıcı çağrısından sonra) çağrılır.
- Örnek başlatıcı bloğu, göründükleri sırayla çalışır.



Örnek:

```
class A{
A(){
System.out.println("parent class constructor invoked");
}
}
class B2 extends A{
B2(){
super();
System.out.println("child class constructor invoked");
}

{System.out.println("instance initializer block is invoked");}

public static void main(String args[]){
B2 b=new B2();
}
}
```

```
Output:parent class constructor invoked
       instance initializer block is invoked
       child class constructor invoked
```

```
class A{
A(){
System.out.println("parent class constructor invoked");
}
}

class B3 extends A{
B3(){
super();
System.out.println("child class constructor invoked");
}

B3(int a){
super();
System.out.println("child class constructor invoked "+a);
}

{System.out.println("instance initializer block is invoked");}

public static void main(String args[]){
B3 b1=new B3();
B3 b2=new B3(10);
}
}
```

```
Output:parent class constructor invoked
       instance initializer block is invoked
       child class constructor invoked
       parent class constructor invoked
       instance initializer block is invoked
       child class constructor invoked 10
```

Final Anahtar Kelimesi

- Kullanıcının değer atamalarında kısıtlanma yapmak için kullanılır.
- Java'da üç şey final olabilir.
  - değişken
  - metot
  - sınıf

1) Java Final değişkeni

- Final tanımlanmış bir değişkenin değeri değiştirilemez.

```
class Bike9{
 final int speedlimit=90;//final variable
 void run(){
  speedlimit=400;
 }
 public static void main(String args[]){
 Bike9 obj=new  Bike9();
 obj.run();
 }
}//end of class
```

```
Output:Compile Time Error
```

2) Java final metot

- Final tanımlanmış bir metot override edilemez.

```
class Bike{
  final void run(){System.out.println("running");}
}

class Honda extends Bike{
   void run(){System.out.println("running safely with 100kmph");}

   public static void main(String args[]){
   Honda honda= new Honda();
   honda.run();
   }
}
```

```
Output:Compile Time Error
```

3) Java final class

- Herhangi bir sınıfı final olarak yaparsanız, onu genişletemezsiniz.

```
final class Bike{}

class Honda1 extends Bike{
void run(){System.out.println("running safely with 100kmph");}

  public static void main(String args[]){
  Honda1 honda= new Honda1();
  honda.run();
  }
}
```

```
Output:Compile Time Error
```

Final metot miras alınır mı?

- Final metot miras olarak aktarılabilir fakat override edilemez.

```
class Bike{
  final void run(){System.out.println("running...");}
}
class Honda2 extends Bike{
   public static void main(String args[]){
    new Honda2().run();
   }
}
```

```
Output:running...
```

Boş veya başlatılmamış final değişken

- Deklare sırasında başlatılmayan bir final değişken, boş final değişken olarak bilinir.
- Nesne oluşturma sırasında başlatılan ve başlatıldıktan sonra değiştirilemeyen bir değişken oluşturmak istiyorsanız, yararlıdır.

```
class Student{
int id;
String name;
final String PAN_CARD_NUMBER;
...
}
```

Boş final değişkeni başlatabilir miyiz?

```
class Bike10{
  final int speedlimit;//blank final variable

  Bike10(){
  speedlimit=70;
  System.out.println(speedlimit);
  }

  public static void main(String args[]){
    new Bike10();
  }
}
```

```
Output: 70
```

static boş final değişken

- Deklare sırasında başlatılmayan bir static final değişken statik final değişken olarak bilinir. Yalnızca statik blokta başlatılabilir.

```
class A{
  static final int data;//static blank final variable
  static{ data=50;}
  public static void main(String args[]){
    System.out.println(A.data);
 }
}
```

Final parametre nedir?

- Herhangi bir parametreyi final olarak bildirirseniz, değerini değiştiremezsiniz.

```
class Bike11{
  int cube(final int n){
   n=n+2;//can't be changed as n is final
   n*n*n;
  }
  public static void main(String args[]){
    Bike11 b=new Bike11();
    b.cube(5);
 }
}
```

```
Output: Compile Time Error
```

KAYNAKLAR

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