Super Keyword in Java

The **super** keyword in Java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

Usage of Java super Keyword

super can be used to refer immediate parent class instance variable.

super can be used to invoke immediate parent class method.

super() can be used to invoke immediate parent class constructor.

## super is used to refer immediate parent class instance variable

**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) super can be used to invoke parent class method

The super keyword can also be used to invoke parent class method. It should be used if subclass contains the same method as parent class. In other words, it is used if method is overridden.

**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...

In the above example Animal and Dog both classes have eat() method if we call eat() method from Dog class, it will call the eat() method of Dog class by default because priority is given to local.

To call the parent class method, we need to use super keyword.

3) super is used to invoke parent class constructor.

The super keyword can also be used to invoke the parent class constructor. Let's see a simple example:

**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

As we know well that default constructor is provided by compiler automatically if there is no constructor. But, it also adds super() as the first statement.

**Another example of super keyword where super() is provided by the compiler implicitly.**

**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

Final Keyword In Java

The **final keyword** in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

variable

method

class

The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable. It can be initialized in the constructor only. The blank final variable can be static also which will be initialized in the static block only. We will have detailed learning of these. Let's first learn the basics of final keyword.

## Java final variable

If you make any variable as final, you cannot change the value of final variable(It will be constant).

### Example of final variable

There is a final variable speedlimit, we are going to change the value of this variable, but It can't be changed because final variable once assigned a value can never be changed.

**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 method

If you make any method as final, you cannot override it.

### Example of final method

**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

If you make any class as final, you cannot extend it.

**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

### Is final method inherited?

Ans) Yes, final method is inherited but you cannot override it. For Example:

**class** Bike{

**final** **void** run(){System.out.println("running...");}

}

**class** Honda2 **extends** Bike{

**public** **static** **void** main(String args[]){

**new** Honda2().run();

   }

}

Output:running...

### Q) What is blank or uninitialized final variable?

A final variable that is not initialized at the time of declaration is known as blank final variable.

If you want to create a variable that is initialized at the time of creating object and once initialized may not be changed, it is useful. For example PAN CARD number of an employee.

It can be initialized only in constructor.

### Example of blank final variable

**class** Student{

**int** id;

String name;

**final** String PAN\_CARD\_NUMBER;

...

}

### Can we initialize blank final variable?

Yes, but only in constructor. For example:

**class** Bike10{

**final** **int** speedlimit;//blank final variable

  Bike10(){

  speedlimit=70;

  System.out.println(speedlimit);

  }

**public** **static** **void** main(String args[]){

**new** Bike10();

 }

}

### static blank final variable

A static final variable that is not initialized at the time of declaration is known as static blank final variable. It can be initialized only in static block.

### Example of static blank final variable

**class** A{

**static** **final** **int** data;//static blank final variable

**static**{ data=50;}

**public** **static** **void** main(String args[]){

    System.out.println(A.data);

 }

}

### Can we declare a constructor final?

No, because constructor is never inherited.