# this keyword in java

There can be a lot of usage of **java this keyword**. In java, this is a **reference variable** that refers to the current object.

**Usage of java this keyword**

Here is given the 6 usage of java this keyword.

1. this keyword can be used to refer current class instance variable.
2. this() can be used to invoke current class constructor.
3. this keyword can be used to invoke current class method (implicitly)
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this keyword can also be used to return the current class instance.

### 1) The this keyword can be used to refer current class instance variable.

|  |
| --- |
| If there is ambiguity between the instance variable and parameter, this keyword resolves the problem of ambiguity. |

#### Understanding the problem without this keyword

|  |
| --- |
| Let's understand the problem if we don't use this keyword by the example given below: |

1. class Student10{
2. int id;
3. String name;
5. Student10(int id,String name){
6. id = id;
7. name = name;
8. }
9. void display(){System.out.println(id+" "+name);}
11. public static void main(String args[]){
12. Student10 s1 = new Student10(111,"Karan");
13. Student10 s2 = new Student10(321,"Aryan");
14. s1.display();
15. s2.display();
16. }
17. }

Output: 0 null

0 null

|  |
| --- |
| In the above example, parameter (formal arguments) and instance variables are same that is why we are using this keyword to distinguish between local variable and instance variable. |

#### Solution of the above problem by this keyword

1. //example of this keyword
2. class Student11{
3. int id;
4. String name;
6. Student11(int id,String name){
7. this.id = id;
8. this.name = name;
9. }
10. void display(){System.out.println(id+" "+name);}
11. public static void main(String args[]){
12. Student11 s1 = new Student11(111,"Karan");
13. Student11 s2 = new Student11(222,"Aryan");
14. s1.display();
15. s2.display();
16. }
17. }



### 2) this() can be used to invoked current class constructor.

The this() constructor call can be used to invoke the current class constructor (constructor chaining). This approach is better if you have many constructors in the class and want to reuse that constructor.

1. //Program of this() constructor call (constructor chaining)
3. class Student13{
4. int id;
5. String name;
6. Student13(){System.out.println("default constructor is invoked");}
8. Student13(int id,String name){
9. this ();//it is used to invoked current class constructor.
10. this.id = id;
11. this.name = name;
12. }
13. void display(){System.out.println(id+" "+name);}
15. public static void main(String args[]){
16. Student13 e1 = new Student13(111,"karan");
17. Student13 e2 = new Student13(222,"Aryan");
18. e1.display();
19. e2.display();
20. }
21. }

Output:

default constructor is invoked

default constructor is invoked

111 Karan

222 Aryan

### this keyword can be passed as an argument in the method.

|  |
| --- |
| The this keyword can also be passed as an argument in the method. It is mainly used in the event handling. Let's see the example: |

1. class S2{
2. void m(S2 obj){
3. System.out.println("method is invoked");
4. }
5. void p(){
6. m(this);
7. }
9. public static void main(String args[]){
10. S2 s1 = new S2();
11. s1.p();
12. }
13. }

# super keyword in java

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

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

## Usage of java super Keyword

1. super is used to refer immediate parent class instance variable.
2. super() is used to invoke immediate parent class constructor.
3. super is used to invoke immediate parent class method.

## 1) super is used to refer immediate parent class instance variable.

***Problem without super keyword***

1. class Vehicle{
2. int speed=50;
3. }
4. class Bike3 extends Vehicle{
5. int speed=100;
6. void display(){
7. System.out.println(speed);//will print speed of Bike
8. }
9. public static void main(String args[]){
10. Bike3 b=new Bike3();
11. b.display();
12. }
13. }

Output:100

|  |
| --- |
| In the above example Vehicle and Bike both class have a common property speed. Instance variable of current class is refered by instance bydefault, but I have to refer parent class instance variable that is why we use super keyword to distinguish between parent class instance variable and current class instance variable. |

***Solution by super keyword***

1. //example of super keyword
3. class Vehicle{
4. int speed=50;
5. }
7. class Bike4 extends Vehicle{
8. int speed=100;
10. void display(){
11. System.out.println(super.speed);//will print speed of Vehicle now
12. }
13. public static void main(String args[]){
14. Bike4 b=new Bike4();
15. b.display();
17. }
18. }

Output:50

## 2) super is used to invoke parent class constructor.

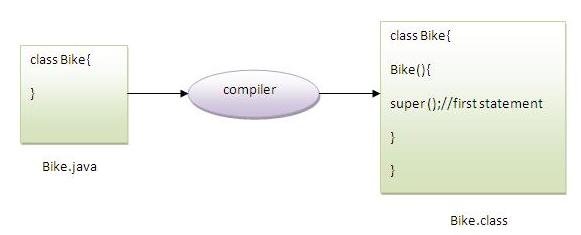
|  |
| --- |
| The super keyword can also be used to invoke the parent class constructor as given below: |

1. class Vehicle{
2. Vehicle(){System.out.println("Vehicle is created");}
3. }
5. class Bike5 extends Vehicle{
6. Bike5(){
7. super();//will invoke parent class constructor
8. System.out.println("Bike is created");
9. }
10. public static void main(String args[]){
11. Bike5 b=new Bike5();
13. }
14. }

Output:Vehicle is created

Bike is created

#### Note: super() is added in each class constructor automatically by compiler.



As we know well that default constructor is provided by compiler automatically but it also adds super() for the first statement.If you are creating your own constructor and you don't have either this() or super() as the first statement, compiler will provide super() as the first statement of the constructor.

# **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:

1. variable
2. method
3. 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.

## 1) 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.

1. **class** Bike9{
2. **final** **int** speedlimit=90;//final variable
3. **void** run(){
4. speedlimit=400;
5. }
6. **public** **static** **void** main(String args[]){
7. Bike9 obj=**new**  Bike9();
8. obj.run();
9. }
10. }//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**

1. **class** Bike{
2. **final** **void** run(){System.out.println("running");}
3. }
5. **class** Honda **extends** Bike{
6. **void** run(){System.out.println("running safely with 100kmph");}
8. **public** **static** **void** main(String args[]){
9. Honda honda= **new** Honda();
10. honda.run();
11. }
12. }

Output:Compile Time Error

## 3) Java final class

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

### **Example of final class**

1. **final** **class** Bike{}
3. **class** Honda1 **extends** Bike{
4. **void** run(){System.out.println("running safely with 100kmph");}
6. **public** **static** **void** main(String args[]){
7. Honda1 honda= **new** Honda();
8. honda.run();
9. }
10. }

Output:Compile Time Error

### **Q) Is final method inherited?**

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

1. **class** Bike{
2. **final** **void** run(){System.out.println("running...");}
3. }
4. **class** Honda2 **extends** Bike{
5. **public** **static** **void** main(String args[]){
6. **new** Honda2().run();
7. }
8. }

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**

1. **class** Student{
2. **int** id;
3. String name;
4. **final** String PAN\_CARD\_NUMBER;
5. ...
6. }

### Que) Can we initialize blank final variable?

Yes, but only in constructor. For example:

1. **class** Bike10{
2. **final** **int** speedlimit;//blank final variable
4. Bike10(){
5. speedlimit=70;
6. System.out.println(speedlimit);
7. }
9. **public** **static** **void** main(String args[]){
10. **new** Bike10();
11. }
12. }

Output:70

### **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(priority more than public static void main).It will be called first.

### **Example of static blank final variable**

1. **class** A{
2. **static** **final** **int** data;//static blank final variable
3. **static**{ data=50;}  //static block
4. **public** **static** **void** main(String args[]){
5. System.out.println(A.data);
6. }
7. }

### Q) What is final parameter?

If you declare any parameter as final, you cannot change the value of it.

1. **class** Bike11{
2. **int** cube(**final** **int** n){
3. n=n+2;//can't be changed as n is final
4. n\*n\*n;
5. }
6. **public** **static** **void** main(String args[]){
7. Bike11 b=**new** Bike11();
8. b.cube(5);
9. }
10. }

Output:Compile Time Error

### **Q) Can we declare a constructor final?**

No, because constructor is never inherited.