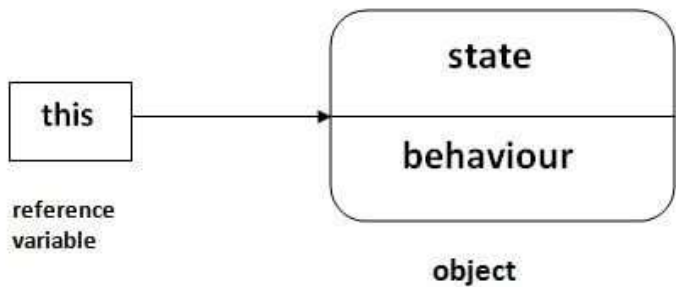




## this keyword in Java

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

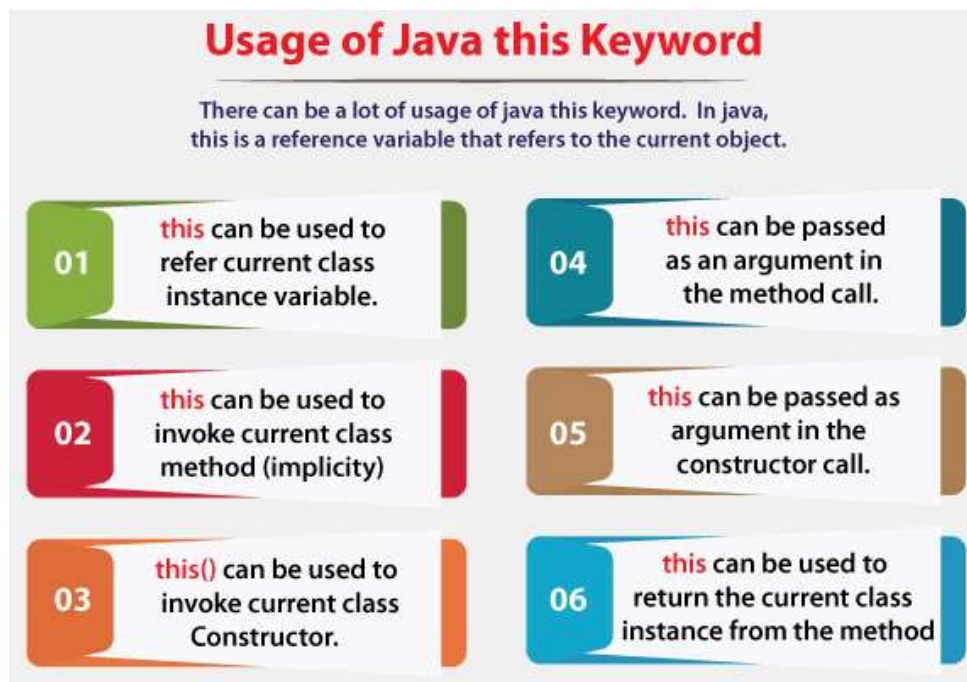


## Usage of Java this keyword

Here is given the 6 usage of java this keyword.

1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly)
3. this() can be used to invoke current class constructor.
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 can be used to return the current class instance from the method.

**Suggestion:** If you are beginner to java, lookup only three usages of this keyword.



## 1) this: to refer current class instance variable

The this keyword can be used to refer current class instance variable. If there is ambiguity between the instance variables and parameters, 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:

```
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();
    }
}
```

Test it Now

### Output:

```
0 null 0.0
0 null 0.0
```

In the above example, parameters (formal arguments) and instance variables are same. So, we are using this keyword to distinguish local variable and instance variable.

### Solution of the above problem by this keyword

```
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();
    }
}
```

#### Test it Now

#### Output:

```
111 ankit 5000.0
112 sumit 6000.0
```

If local variables(formal arguments) and instance variables are different, there is no need to use this keyword like in the following program:

### Program where this keyword is not required

```
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();
    }
}
```

Test it Now

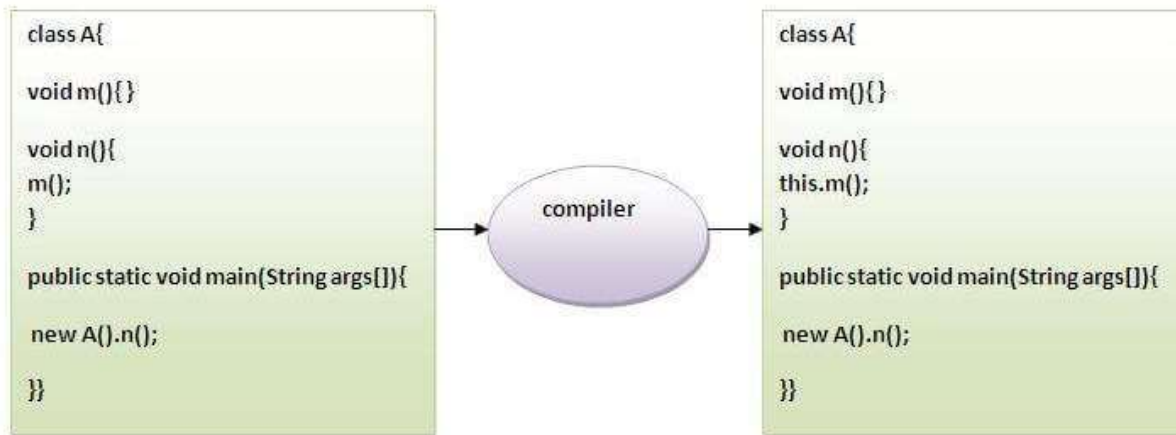
### Output:

```
111 ankit 5000.0
112 sumit 6000.0
```

It is better approach to use meaningful names for variables. So we use same name for instance variables and parameters in real time, and always use this keyword.

## 2) this: to invoke current class method

You may invoke the method of the current class by using the this keyword. If you don't use the this keyword, compiler automatically adds this keyword while invoking the method. Let's see the example



```
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();
    }
}
```

Test it Now

Output:

```
hello n
hello m
```

### 3) this() : to invoke current class constructor

The this() constructor call can be used to invoke the current class constructor. It is used to reuse the constructor. In other words, it is used for constructor chaining.

#### Calling default constructor from parameterized constructor:

```
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);
    }
}
```

Test it Now

#### Output:

```
hello a
10
```

#### Calling parameterized constructor from default constructor:

```
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();
    }
}
```

**Test it Now****Output:**

```
5  
hello a
```

## Real usage of this() constructor call

The this() constructor call should be used to reuse the constructor from the constructor. It maintains the chain between the constructors i.e. it is used for constructor chaining. Let's see the example given below that displays the actual use of this keyword.

```
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();
}}
```

**Test it Now**

### Output:

```
111 ankit java 0.0
112 sumit java 6000.0
```

Rule: Call to this() must be the first statement in constructor.

```
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();
    }
}
```

**Test it Now**

**Output:**

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

#### 4) this: to pass 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:

```
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();  
    }  
}
```

Test it Now

**Output:**

```
method is invoked
```

#### Application of this that can be passed as an argument:

In event handling (or) in a situation where we have to provide reference of a class to another one. It is used to reuse one object in many methods.

#### 5) this: to pass as argument in the constructor call

We can pass the this keyword in the constructor also. It is useful if we have to use one object in multiple classes. Let's see the example:

```
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();
    }
}
```

#### Test it Now

Output:10

## 6) this keyword can be used to return current class instance

We can return this keyword as an statement from the method. In such case, return type of the method must be the class type (non-primitive). Let's see the example:

### Syntax of this that can be returned as a statement

```
return_type method_name(){
    return this;
}
```

## Example of this keyword that you return as a statement from the method

```
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();  
}  
}
```

Test it Now

### Output:

```
Hello java
```

## Proving this keyword

Let's prove that this keyword refers to the current class instance variable. In this program, we are printing the reference variable and this, output of both variables are same.

```
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();  
}  
}
```

Test it Now

### Output:

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

← Prev

Next →

 Youtube For Videos Join Our Youtube Channel: [Join Now](#)


## Feedback

- Send your Feedback to [feedback@javatpoint.com](mailto:feedback@javatpoint.com)

## Help Others, Please Share





## Learn Latest Tutorials











 Splunk tutorial  
Splunk

 SPSS tutorial  
SPSS






 Swagger  
tutorial  
Swagger

 T-SQL tutorial  
Transact-SQL













 Tumblr tutorial  
Tumblr

 <b>React tutorial</b> ReactJS	 <b>Regex tutorial</b> Regex	 <b>Reinforcement learning tutorial</b> Reinforcement Learning	 <b>R Programming tutorial</b> R Programming	 <b>RxJS tutorial</b> RxJS
 <b>React Native tutorial</b> React Native	 <b>Python Design Patterns</b> Python Design Patterns	 <b>Python Pillow tutorial</b> Python Pillow	 <b>Python Turtle tutorial</b> Python Turtle	 <b>Keras tutorial</b> Keras

## Preparation

 <b>Aptitude</b> Aptitude	 <b>Logical Reasoning</b> Reasoning	 <b>Verbal Ability</b> Verbal Ability	 <b>Interview Questions</b> Interview Questions	 <b>Company Interview Questions</b> Company Questions
---	---	---	---	---

## Trending Technologies

 <b>Artificial Intelligence Tutorial</b> Artificial Intelligence	 <b>AWS Tutorial</b> AWS	 <b>Selenium tutorial</b> Selenium	 <b>Cloud Computing tutorial</b> Cloud Computing	 <b>Hadoop tutorial</b> Hadoop
 <b>ReactJS Tutorial</b> ReactJS	 <b>Data Science Tutorial</b> Data Science	 <b>Angular 7 Tutorial</b> Angular 7	 <b>Blockchain Tutorial</b> Blockchain	 <b>Git Tutorial</b> Git
 <b>Machine Learning Tutorial</b> Machine Learning	 <b>DevOps Tutorial</b> DevOps			

## B.Tech / MCA



DBMS tutorial  
DBMS



Data Structures  
tutorial  
Data Structures



DAA tutorial  
DAA



Operating  
System tutorial  
Operating System



Computer  
Network tutorial  
Computer Network



Compiler  
Design tutorial  
Compiler Design



Computer  
Organization and  
Architecture  
Computer  
Organization



Discrete  
Mathematics  
Tutorial  
Discrete  
Mathematics



Ethical Hacking  
Tutorial  
Ethical Hacking



Computer  
Graphics Tutorial  
Computer Graphics



Software  
Engineering  
Tutorial  
Software  
Engineering



html tutorial  
Web Technology



Cyber Security  
tutorial  
Cyber Security



Automata  
Tutorial  
Automata



C Language  
tutorial  
C Programming



C++ tutorial  
C++



Java tutorial  
Java



.Net  
Framework  
tutorial  
.Net



Python tutorial  
Python



List of  
Programs  
Programs



Control  
Systems tutorial  
Control System



Data Mining  
Tutorial  
Data Mining



Data  
Warehouse  
Tutorial  
Data Warehouse

