The **static keyword** in java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than instance of the class.

The static can be:

1. variable (also known as class variable)
2. method (also known as class method)
3. block

1) Java static variable

If you declare any variable as static, it is known static variable.

* The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees, college name of students etc.
* The static variable gets memory only once in class area at the time of class loading.

Advantage of static variable

It makes your program **memory efficient** (i.e it saves memory).

**class** Student

{

**int** rollno;

   String name;

**static** String college ="ITS";

   Student(**int** r,String n)

{

   rollno = r;

   name = n;

   }

**void** display (){System.out.println(rollno+" "+name+" "+college);}

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

{

 Student s1 = **new** Student(111,"Karan");

 Student s2 = **new** Student(222,"Aryan");

 s1.display();

 s2.display();

 }

}

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=Student8)

Output:111 Karan ITS

222 Aryan ITS

2) Java static method

If you apply static keyword with any method, it is known as static method.

* A static method belongs to the class rather than object of a class.
* A static method can be invoked without the need for creating an instance of a class.
* static method can access static data member and can change the value of it.

Example of 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);

  }

}

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=Calculate)

Output:125

Restrictions for static method

|  |
| --- |
| There are two main restrictions for the static method. They are: |

|  |
| --- |
| 1. The static method can not use non static data member or call non-static method directly. 2. this and super cannot be used in static context. |

**class** A{

**int** a=40;//non static

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

  System.out.println(a);

 }

}

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=A)

Output:Compile Time Error

3) Java static block

* Is used to initialize the static data member.
* It is executed before main method at the time of classloading.

Example of static block

**class** A2{

**static**{System.out.println("static block is invoked");}

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

   System.out.println("Hello main");

  }

}

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=A2)

Output:static block is invoked

Hello main

# 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.

### 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.

1. **class** Student{
2. **int** rollno;
3. String name;
4. **float** fee;
5. Student(**int** rollno,String name,**float** fee){
6. **this**.rollno=rollno;
7. **this**.name=name;
8. **this**.fee=fee;
9. }
10. **void** display(){System.out.println(rollno+" "+name+" "+fee);}
11. }

|  |  |
| --- | --- |
| 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.  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: |  1. **class** Student{ 2. **int** rollno; 3. String name; 4. **float** fee; 5. Student(**int** rollno,String name,**float** fee){ 6. rollno=rollno; 7. name=name; 8. fee=fee; 9. } 10. **void** display(){System.out.println(rollno+" "+name+" "+fee);} 11. } 12. **class** TestThis1{ 13. **public** **static** **void** main(String args[]){ 14. Student s1=**new** Student(111,"ankit",5000f); 15. Student s2=**new** Student(112,"sumit",6000f); 16. s1.display(); 17. s2.display(); 18. }}   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  1. **class** Student{ 2. **int** rollno; 3. String name; 4. **float** fee; 5. Student(**int** rollno,String name,**float** fee){ 6. **this**.rollno=rollno; 7. **this**.name=name; 8. **this**.fee=fee; 9. } 10. **void** display(){System.out.println(rollno+" "+name+" "+fee);} 11. } 13. **class** TestThis2{ 14. **public** **static** **void** main(String args[]){ 15. Student s1=**new** Student(111,"ankit",5000f); 16. Student s2=**new** Student(112,"sumit",6000f); 17. s1.display(); 18. s2.display(); 19. }}   [**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=TestThis2)  Output:  111 ankit 5000  112 sumit 6000 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 |

1. **class** A{
2. **void** m()
3. {
4. System.out.println("hello m");
5. }
6. **void** n()
7. {
8. System.out.println("hello n");
9. //m();//same as this.m()
10. **this**.m();
11. }
12. }
13. **class** TestThis4{
14. **public** **static** **void** main(String args[]){
15. A a=**new** A();
16. a.n();
17. }}

### 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.

1. **class** A{
2. A()
3. {
4. System.out.println("hello a");
5. }
6. A(**int** x)
7. {
8. **this**();
9. System.out.println(x);
10. }
11. }
12. **class** TestThis5{
13. **public** **static** **void** main(String args[]){
14. A a=**new** A(10);
15. }}

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:

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