# **Java static keyword**

# Heading 1

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 an instance of the class.

The static can be:

1. Variable (also known as a class variable)
2. Method (also known as a class method)
3. Block
4. Nested class

## **Java static variable**

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

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

### **Advantages of static variable**

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

1. **class** Student{
2. **int** rollno;//instance variable
3. String name;
4. **static** String college ="ITS";//static variable
5. //constructor
6. Student(**int** r, String n){
7. rollno = r;
8. name = n;
9. }
10. //method to display the values
11. **void** display (){System.out.println(rollno+" "+name+" "+college);}
12. }
13. //Test class to show the values of objects
14. **public** **class** TestStaticVariable1{
15. **public** **static** **void** main(String args[]){
16. Student s1 = **new** Student(111,"Karan");
17. Student s2 = **new** Student(222,"Aryan");
18. //we can change the college of all objects by the single line of code
19. //Student.college="BBDIT";
20. s1.display();
21. s2.display();
22. }
23. }

## **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 the object of a class.
* A static method can be invoked without the need for creating an instance of a class.
* A static method can access static data member, cannot access non-static data member and can change the value of it.
* Static method only call (access) the other static method, not call any non-static method.
* **class** Student{
* **int** rollno;
* String name;
* **static** String college = "ITS";
* //static method to change the value of static variable
* **static** **void** change(){
* college = "BBDIT";
* }
* //constructor to initialize the variable
* Student(**int** r, String n){
* rollno = r;
* name = n;
* }
* //method to display values
* **void** display(){System.out.println(rollno+" "+name+" "+college);}
* }
* //Test class to create and display the values of object
* **public** **class** TestStaticMethod{
* **public** **static** **void** main(String args[]){
* Student.change();//calling change method
* //creating objects
* Student s1 = **new** Student(111,"Karan");
* Student s2 = **new** Student(222,"Aryan");
* Student s3 = **new** Student(333,"Sonoo");
* //calling display method
* s1.display();
* s2.display();
* s3.display();
* }
* }

### **Restrictions for the static method**

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

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

### **Q) Why is the Java main method static?**

Ans: It is because the object is not required to call a static method. If it were a non-static method, JVM creates an object first then call main() method that will lead the problem of extra memory allocation.

## **3) Java static block**

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

### **Example of static block**

1. **class** A2{
2. **static**{System.out.println("static block is invoked");}
3. **public** **static** **void** main(String args[]){
4. System.out.println("Hello main");
5. }
6. }

### **Q) Can we execute a program without main() method?**

Ans) No, one of the ways was the static block, but it was possible till JDK 1.6. Since JDK 1.7, it is not possible to execute a java class without the main method.

1. **class** A3{
2. **static**{
3. System.out.println("static block is invoked");
4. System.exit(0);
5. }
6. }

[**Test it Now**](http://www.javatpoint.com/opr/test.jsp?filename=A3)

Output:

static block is invoked

Since JDK 1.7 and above, output would be:

Error: Main method not found in class A3, please define the main method as:

public static void main(String[] args)

or a JavaFX application class must extend javafx.application.Application

**why this and super is not used in static context?**

**This**keyword: It has the following uses:

* It can be used to refer to the current class **instance variable.**
* It is used to refer to the current class object, (the object whose constructor or method is getting called). You can refer to any member of the current object from within an instance method or constructor using **this.**

Now since, static methods are related to the class and not to the objects, **this**cannot be used inside static methods.

Now, **this**keyword in java is a **reference variable** that refers to the **current object**. Also the **super** keyword in java is a **reference variable** which is used to refer immediate **parent class object**.

So, we can say that **this**and **super** both keyword is **reference variable** that refers to some object. In other words, these both keywords belong to instance of the class.

Whereas, **static method** belongs to the class than instance of the class. And so **static method** can’t access **this** and **super** keyword in java.

**public** **class** StaticDemo {

**private** **static** **int** *x*=10;

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

//static int a=12;

//we can not declare the static variable inside block

System.***out***.println("m->"+StaticDemo.*x*);

*fun*();

// static int arr1[] = { 11, 22, 33 };

// static int arr2[] = { 11, 22, 33, 44, 55 };

// static int ptr[]; //error

**int** arr1[] = { 11, 22, 33 };

**int** arr2[] = { 11, 22, 33, 44, 55 };

**int** ptr[];

ptr = arr1;

arr1 = arr2;

arr2 = ptr;

System.***out***.print(arr1.length + " "); //5

System.***out***.println(arr2.length); //3

}

**private** **static** **void** fun() {

StaticDemo d =**new** StaticDemo();

//static int a=12;

System.***out***.println("m-"+*x*);

System.***out***.println("m-"+d.a); //private is not direct access

}

**static** {

//static int a=12;

StaticDemo d =**new** StaticDemo();

System.***out***.println("b="+d.*x*);

}

}

**Output**

b=10

m->10

m-10

5 3

**Question 8: Can a class have more than one anonymous blocks?**

A class can have any no. of anonymous blocks.

**Question 9: Can a class have more than one static blocks?**

class can have only one static block.