

Java static keyword

The static keyword in Java is mainly used for memory management.

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



A local variable in Java is a variable that's declared within the body of a method.

Instance variables in Java are non-static variables which are defined in a class while any method, constructor or a block.

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

1. 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, the college name of students, etc.

2. The static variable gets memory only once in the class area at the time of class loading.

```
class Student{
    int rollNo; //instance variable
    String name; //instance variable
    static String college = "Wootia Institute of Technology and Management"; //static variable

    //constructor
    Student(int r, String n) { //static variable r and n
        rollNo = r; //instance variable & local variable r
        name = n; //instance variable & local variable n
    }

    //method to display the values
    void display() {
        System.out.println(rollNo+" "+name+" "+college); //here rollNo, "name" "college" are local variables
    }
}

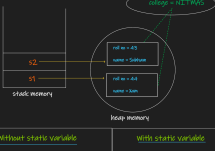
//Test class to show the values of objects
public class CodeKam {
    public static void main(String[] args) {
        Student s1 = new Student(44,"Subham"); //creating an object of Student
        Student s2 = new Student(44,"Xam"); //creating an object of Student

        //we can change the college of all objects by the single line of code
        Student.college="NITMAS";

        s1.display();
        s2.display();
    }
}
```

Output

```
43 Subham NITMAS
44 Xam NITMAS
```



Without static variable

```
class Student {
    //Method get memory each time when the instance is created
    int rollNo;

    //Constructor
    Student(int rollNo, String name) {
        rollNo = rollNo;
        name = name;
    }

    //Method to display values
    void display() {
        System.out.println(rollNo+" "+name+" "+college);
    }
}
```

Output

```
43
44
45
```

With static variable

```
class Student {
    //Method get memory each time when the instance is created
    int rollNo;

    //Constructor
    Student(int rollNo, String name) {
        rollNo = rollNo;
        name = name;
    }

    //Method to display values
    void display() {
        System.out.println(rollNo+" "+name+" "+college);
    }
}
```

Output

```
43
44
45
```

In this example, we have created an instance variable named `rollNo` which is instantiated in the constructor. Each instance variable gets the memory at the time of object creation, each object will have the copy of this instance variable. If it is instantiated, it won't affect other objects. So each object will have this object in the object variable.

static variable will get the memory only once, if any object changes the value of the static variable, it will retain its value.

Java static method

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

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

Note:

1. The static method can not use non-static data members or call the non-static method directly.
2. This and super cannot be used in the static context.

Annotation references in static methods:

```
class A {
    int a=40; //non static

    public static void main(String args[]) {
        System.out.println(a);
    }
}
```

Warning: non static variable a cannot be referenced from a static context

The most common example of a static method is the `main()` method.

As discussed above, any static member can be accessed before any objects of its class are created, and without reference to any object.

Syntax to declare the static method:

```
Access_modifier static void methodName()
{
    // Method body
}
```

Why use Static Methods?

1. To access and change static variables and either non-object-based static methods.
2. Utility and assist classes frequently employ static methods.

Restrictions in Static Methods:

1. Non-static data members or non-static methods cannot be used by static methods, and static methods cannot call non-static methods directly.
2. In a static environment, `this` and `super` aren't allowed to be used.

Why is the main method in Java static?

It's because calling a static method isn't needed of the object. If it were a non-static function, JVM would first build an object before calling the `main()` method, resulting in an extra memory allocation difficulty.

Instance Methods	Static Methods
It requires an object of the class. It can access all attributes of a class. The methods can be accessed only using object reference. Syntax: <code>Object.methodName()</code> It's an example of pass-by-value programming.	It doesn't require an object of the class. It can access only the static attributes of a class. The method is only accessed by class name. Syntax: <code>className.methodName()</code> It's an example of pass-by-reference programming.

```
class Student{
    int rollNo;
    String name;
    static String college = "Wootia Institute of Management and ";
    //static method to change the value of static variable
    static void change(){
        college = "NITMAS";
    }
    //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 CodeKam {
    public static void main(String args[]) {
        Student.change(); //calling change method
        //creating objects
        Student s1 = new Student(44,"Subham");
        Student s2 = new Student(44,"Xam");
        Student s3 = new Student(44,"Subhanish");
        //calling display method
        s1.display();
        s2.display();
        s3.display();
    }
}
```

Output

```
43 Subham NITMAS
44 Xam NITMAS
45 Subhanish NITMAS
```

Java static block

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

```
class A {
    static {
        System.out.println("static block is invoked");
    }
    public static void main(String args[]) {
        System.out.println("Hello main");
    }
}
```

Output

```
static block is invoked
Hello main
```

Static classes

A class can be made static only if it is a nested class. We cannot declare a top-level class with a static modifier but can declare nested classes as static. Such types of classes are called nested static classes. Nested static class doesn't need a reference of Outer class. In this case, a static class cannot access non-static members of the Outer class.

1. A static nested class may be instantiated without instantiating its outer class.
2. Inner classes can access both static and non-static members of the outer class. A static class can access only the static members of the outer class.

```
public class CodeKam {

    private static String str = "CodeKam";

    // static class
    static class MyNestedClass {

        // non-static method
        public void disp(){
            System.out.println(str);
        }
    }

    public static void main(String args[]) {
        CodeKam.MyNestedClass obj = new
        CodeKam.MyNestedClass();
        obj.disp();
    }
}
```

Output

```
CodeKam
```