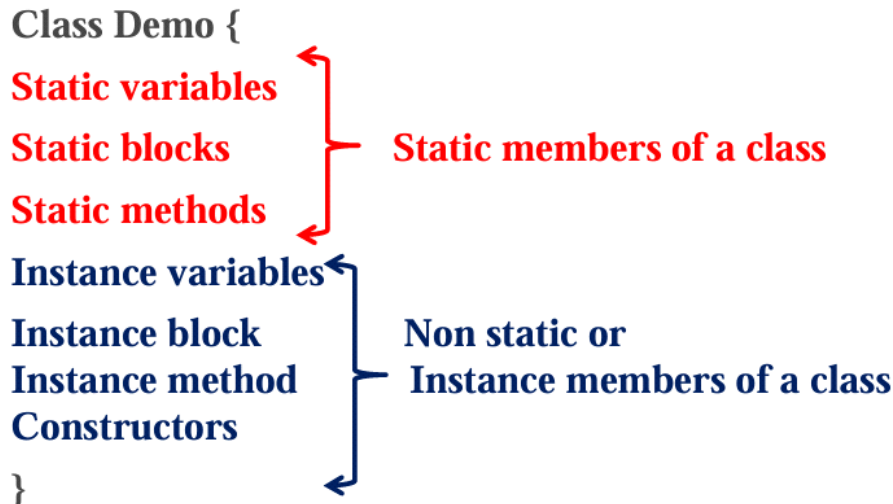


Static Keywords

30 October 2025 18:10

A class in java consist of the following members:



- Static keyword belong to class itself.
- Non static members belong to the individual objects.
- Static member can be accessed by static as well as instances member.
- Instance variable can be accessed by only instances member. Static members cannot access instances variables.
- It is used to create members (variables, method, block or nested classes) that belongs to the class rather than specific object.
- We do not need to create an object to access a static member we can access it using the class name directly.

Class

Static variables
Static blocks
Static methods

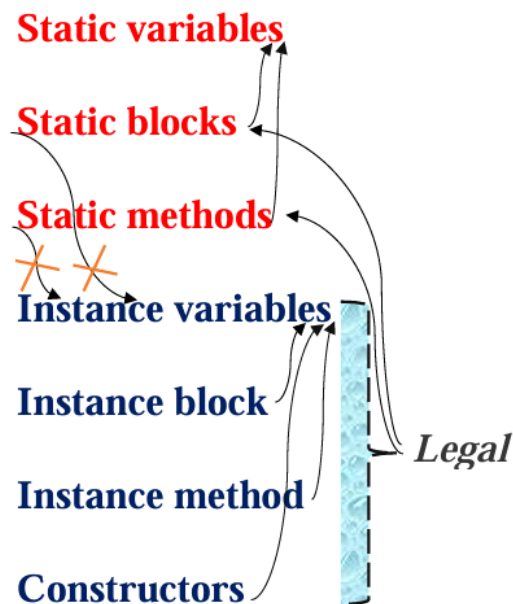
Object

Instance variables
Instance blocks
Instance methods
Constructors.

Used:-

- It is used for memory managements by sharing data among all instances of a class.
- It means a static member is shared among all objects of the class.

RULES OF STATIC:



Static Variables can be accessed by static as well as instance variables.

Instance variables can be accessed by only instance members.

Static members cannot access instance variables.

In the Java programming language, the keyword `static` indicates that the particular member belongs to a type itself, rather than to an instance of that type. This means that only one instance of that static member is created which is shared across all instances of the class. The keyword can be applied to variables, methods, blocks and nested class.

```
class Student {
    static String school = "ABC School"; // static variable
    String name; // non-static variable
}

public class Main {
    public static void main(String[] args) {
        // Accessing static variable directly using class name
        System.out.println(Student.school); // ✔ No object needed

        // Creating an object to access non-static variable
        Student s = new Student();
        s.name = "Ravi";
        System.out.println(s.name); // ✔ Object needed
    }
}
```


Static variables:

- Static variable in Java is a variable which belongs to the class, not to object and initialized only once at the start of the execution. These variables will be initialized first, before the initialization of any instance variables.
- In the case of instances variables for every object a separate copy will be create but in the case of static variables for the entire class only one copy will be created and shared by every object of that class.
- Static variables will be created at the time of class loading and destroyed at the time of class unloading hence the scope of the static variable exactly the same as the scope of the class file.
- If we change the value if the static variable using any object, it will affect all instances of the class because the variables belongs to the class not the individual objects.

Static method

- Static method in java is a method which belong to class and not to the object. A static method can access only static data, it cannot access non static data.
 - A static method can call only static methods and cannot call a non-static method from it.
 - A static method can be accessed directly by the class name and doesn't need any object.
 - A static method cannot refer to this or super keyword in anyways.
-
- Use static if we want to access a function in class using just class name, without creating an object.
 - **Why static method cannot access instances variables?**
 - Because static method are not associated with any object they cannot access instances variables which are object specific .

```
class Demo {  
    static int a = 100;          // static variable  
    int b = 200;                // instance variable  
  
    static void staticMethod() {  
        System.out.println("Static variable a = " + a); // ✓ OK  
        // System.out.println("Instance variable b = " + b); ✗ Error  
    }  
  
    void instanceMethod() {  
        System.out.println("Static variable a = " + a); // ✓ OK  
        System.out.println("Instance variable b = " + b); // ✓ OK  
    }  
}
```



Static block:

- The static block is a block of statement inside a Java class that will be executed after object creation and before call to the constructor. A static block helps to initialize the static data members, just like constructors help to initialize instance members.
- If we want to perform any activity at the time of loading a class file we have to define that activity inside the static block.
- A static block in java is a block of code that is executed only once when the class is loaded into memory by the JVM.
- If we want to perform an action while the class loads (like initializing static variable, loading configuration, printing logs) use a static block.
- In java if we want to execute some code at the time class is loaded we defines the code inside a static block.

```

class Test
{
    static int a,b;
    static
    {
        System.out.println("Inside static block");
        a = 10;
        b = 20;
    }
    static void fun1()
    {
        System.out.println("Inside static method");
    }
    int x,y;
    {
        System.out.println("Inside instance block");
    }
    void fun2()
    {
        System.out.println("Inside instance method");
    }
    Test()
    {
        System.out.println("Inside constructor");
        x = 30;
        y = 40;
    }
}

class Demo
{
    public static void main(String[] args)
    {
        Test.fun1();
        Test t = new Test();
        t.fun2();
    }
}

```

Static block

Static block is used for **initializing the static variables**. Although *static* variables **can be initialized directly during declaration**, there are situations when we're required to do the multiline processing.

This block gets executed when the **class is loaded in the memory**. A class can **have multiple Static blocks**, which will execute in the **same sequence** in which they have been written into the program.

Java static variables

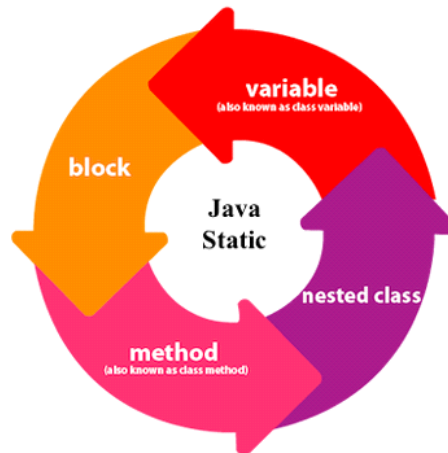
A static variable is common to all the instances (or objects) of the class because it is a class level variable. In other words you can say that only a single copy of static variable is created and shared among all the instances of the class. Memory allocation for such variables only happens once when the class is loaded in the memory.

Few Important Points:

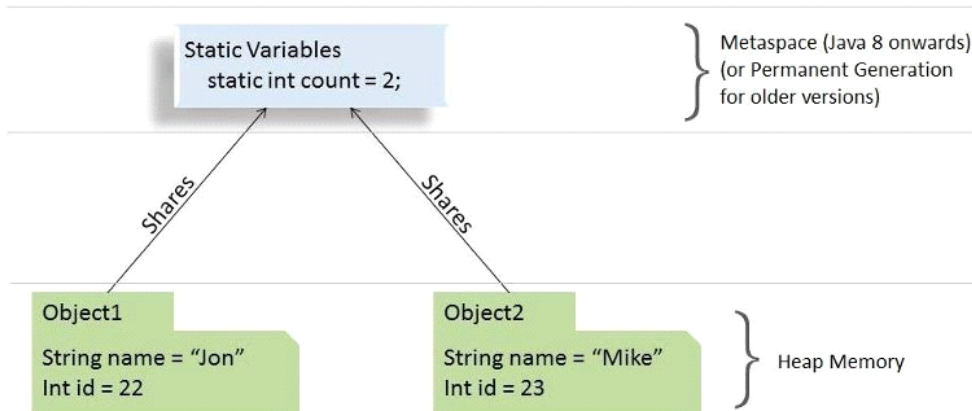
- Static variables are also known as Class Variables/static fields.
- Unlike **non-static variables**, such variables can be accessed directly in static and non-static methods.

Let's see what static variables can do...

1. Static variables can be **accessed directly in Static method**.
2. Static variables are **shared among all the instances of class**. Whereas **non-static variables cannot be accessed by static but can only be accessed by non-static**. That is because **non-static/instance variable would have not got allocated in the memory**



In Java, if a field is declared **static**, then exactly a single copy of that field is created and shared among all instances of that class. It doesn't matter how many times we initialize a class; there will always be only one copy of *static* field belonging to it. The value of this *static* field will be shared across all object of either same or any different class.



From the memory perspective, **static variables go in a particular pool in JVM memory called Metaspace** (before Java 8, this pool was called Permanent Generation or PermGen, which was completely removed and replaced with Metaspace).

Where to use static variables????

- When the value of variable is independent of objects.
- 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.
- If **static** variables require additional, multi-statement logic while initialization, then a **static** block can be used.



If you are thinking is there any **advantage** of using **static variables** then yes there is: It makes your program **memory efficient** (i.e., it saves memory).

How?? Well this is how...

Suppose there are 500 students in my college, now all instance data members will get memory each time when the object is created. All students have its unique rollno and name, so instance data member is good in such case. Here, "college" refers to the common property of all objects. If we make it static, this field will get the memory only once.

Key points to remember..

- Since *static* variables belong to a class, they can be **accessed directly using class name** and **don't need any object reference**.
- *static* variables can only be **declared at the class level**.
- *static* fields can be **accessed without object initialization**.
- Although we can access *static* fields or the class variables using an object reference, we should refrain from using it as in this case it becomes difficult to figure whether it's an instance variable or a class variable; instead, we should always refer to *static* variables using class.
- If initialization of *static* variables requires some additional logic except the assignment
- If the initialization of static variables is error-prone and requires exception handling

Static Methods

Similar to *static* variables, ***static* methods also belong to a class** instead of the object, and so they can be **called without creating the object of the class** in which they reside. They're meant to be used without creating objects of the class.

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 and can change the value of it.

***static* methods** are generally used to perform an operation that is **not dependent upon instance creation**.

If there is a code that is supposed to be **shared across all instances of that class**, then write that code in a *static* method.

A static method can **access only static variables of class** and **invoke only static methods of the class**



The **main() method** that is the entry point of a java program itself is a **static method**. Wonder why? 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**.

static methods are also widely used to **create utility or helper classes** so that they can be obtained without creating a new object of these classes.

Why to use static methods???

- To access/manipulate static variables and other static methods that don't depend upon objects.
- *static* methods are widely used in utility and helper classes.

Points to be remembered...

- *static* methods in Java are resolved at compile time. Since method overriding is part of Runtime Polymorphism, **so static methods can't be overridden**.
- abstract methods can't be static. //will be covered in future classes
- *static* methods **cannot use *this* or *super* keywords**. //will be covered in future classes
- The static method **cannot use non static data member** or call non-static method directly.
- The following combinations of the instance, class methods and variables are valid:
 1. Instance methods can directly access both instance methods and instance variables
 2. Instance methods can also access *static* variables and *static* methods directly
 3. *static* methods can access all *static* variables and other *static* methods
 4. ***static* methods cannot access instance variables and instance methods directly**; they need some object reference to do so

Next Chapter is Memory Management:-