24. Static Keyword in Java

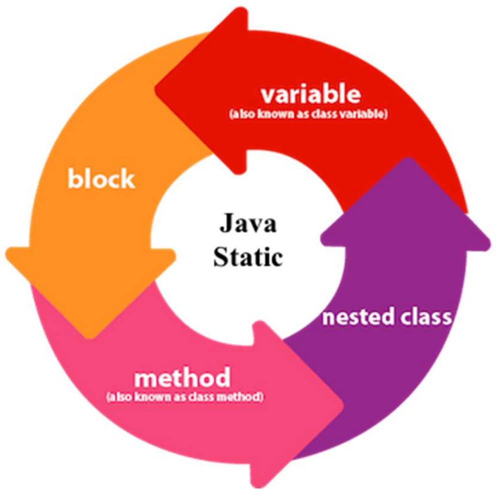
The **static keyword** in Java is used for memory management mainly. We can apply static keyword with variables , methods, blocks and nested classes . The static keyword belongs to the class than an instance of the class.

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

24.1 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).

Understanding the problem without static variable

```
class Student
{
     int rollno;
     String name;
     String college="ITS";
}
```

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.

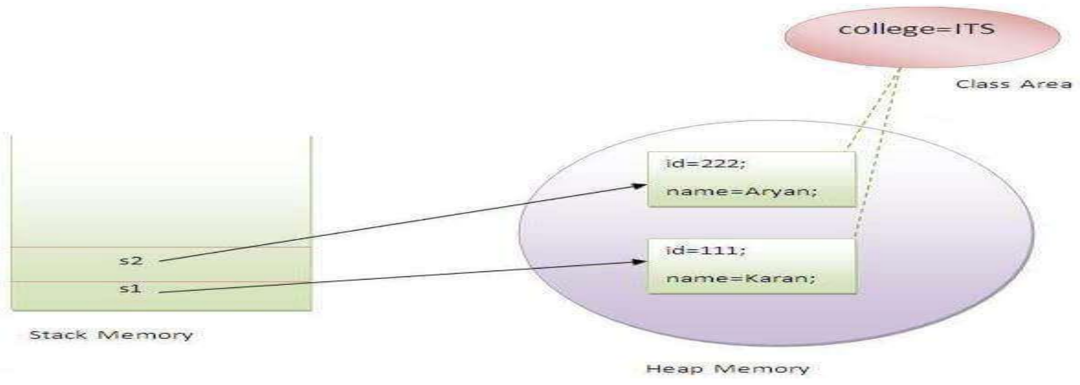
Java static property is shared to all objects.

Example of static variable:

```
//Java Program to demonstrate the use of static variable
class Student
{
	int rollno;//instance variable
	String name;
	static String college ="ITS";//static variable
	//constructor
	Student(int r, String n)
	{
		rollno = r;
		name = n;
	}
	//method to display the values
	void display ()
	{
		System.out.println(rollno+" "+name+" "+college);
	}
}
//Test class to show the values of objects
public class TestStaticVariable1
{
	public static void main(String args[])
	{
		Student s1 = new Student(111,"Karan");
		Student s2 = new Student(222,"Aryan");
		//we can change the college of all objects by the single
		//line of code
		//Student.college="BBDIT";
		s1.display();
		s2.display();
	}
}
```

Output:

```
111 Karan ITS
222 Aryan ITS
```



Example Program of the counter without static variable:

In this example, we have created an instance variable named count which is incremented in the constructor. Since instance variable gets the memory at the time of object creation, each object will have the copy of the instance variable. If it is incremented, it won't reflect other objects. So each object will have the value 1 in the count variable.

```
//Java Program to demonstrate the use of an instance variable
//which get memory each time when we create an object of the class.
public class Counter
{
	//will get memory each time when the instance is created
	int count=0;

	Counter()
	{
		count++;//incrementing value
		System.out.println(count);
	}
	public static void main(String args[])
	{
		//Creating objects
		Counter c1=new Counter();
		Counter c2=new Counter();
		Counter c3=new Counter();
	}
}
```

Output:

```
1
1
1
```

Example Program of counter by static variable:

As we have mentioned above, static variable will get the memory only once, if any object changes the value of the static variable, it will retain its value.

```
//Java Program to illustrate the use of static variable which
//is shared with all objects.
public class Counter2
{
	//will get memory only once and retain its value
	static int count=0;

	Counter2()
	{
		count++;//incrementing the value of static variable
		System.out.println(count);
	}
```

```
     public static void main(String args[])
     {
          //creating objects
          Counter2 c1=new Counter2();
          Counter2 c2=new Counter2();
          Counter2 c3=new Counter2();
     }
}
```

Output:

```
1
2
3
```

24. 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 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.

Example of static method:

```
//Java Program to demonstrate the use of a 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();
    }
}
```

Output:

```
111 Karan BBDIT
222 Aryan BBDIT
333 Sonoo BBDIT
```

Another example of a static method that performs a normal calculation

```
//Java Program to get the cube of a given number using the static
//method
public 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);
     }
}
```

Output:

```
125
```

Restrictions for the 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.

```
public class A
{
	int a=40;//non static
	public static void main(String args[])
	{
		System.out.println(a);
	}
}
```

Output:

```
Exception in thread "main" java.lang.Error: Unresolved compilation
problem:
	Cannot make a static reference to the non-static field a

	at A.main(A.java:6)
```

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.

24.3 Java static block

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

Example of static block

```
public class A2
{
	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
```

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.

```
public class A
{
	static
	{
		System.out.println("static block is invoked");
		System.exit(0);
	}
}
```

Output:

```
static block is invoked
```

Since JDK 1.7 and above, Output will be:

Output:

```
Error: Main method not found in class A, please define the main
method as:
   public static void main(String[] args)
or a JavaFX application class must extend
javafx.application.Application
```

Note: When you hit run program, JVM will searches for class which consists main() Method. JVM will loads the class first. JVM Class loading procedure is given below as steps wise.

1. JVM loads the static variables in the class.
2. JVM loads the static block in the class.
3. JVM goes to the main() Method.

Example1:

```
public class A
{
	static int a,b,c;
	static
	{
		System.out.println("static block");
	}
	public static void main(String[] args)
	{
		//static variable is loading first that's why we are
		//getting default values for below statement
		System.out.println(a);
		System.out.println("main method");
	}
}
```

Output:

```
static block
0
```

```
main method
```

Note: When you write another class in the above as mentioned in the below example, Here also JVM follows the same procedure.

Example2:

```
class B
{
	static int a,b,c;
	static
	{
		System.out.println("class B static");
	}
}
public class A
{
	static int a,b,c;
	static
	{
		System.out.println("static block");
	}
	public static void main(String[] args)
	{
		//static variable is loading first that's why we are
		//getting default values for below statement
		System.out.println(a);
		System.out.println("main method");
		B obj=new B();
		System.out.println(obj.a);
	}
}
```

Output:

```
static block
0
main method
class B static
0
```

Note: We can directly call static methods from the class without creating objects of the class as mentioned in the example.

Example3:

```
class B
{
	static int display()
	{
		System.out.println("display method");
		return 2;
```

```
      }
}
public class A
{
      static void show()
      {
            System.out.println("show method");
      }
      public static void main(String[] args)
      {
            System.out.println("main method");
            A.show();
            System.out.println(B.display());
      }
}
```

Output:

```
main method
show method
display method
2
```

Note: static block will loads only once. static means it is common for the class so it will be loaded only once. static block cannot be loaded every time when you create an object the class. Example for this is mentioned below.

Example4:

```
class Emp
{
      int id;
      int Salary;
      static String ceo;
      static
      {
            //to check how many times it is loading
            System.out.println("static block");
            ceo="Sundar";
      }
}
public class A
{
      public static void main(String[] args)
      {
            Emp lokesh=new Emp();
            lokesh.id=229;
            lokesh.Salary=10000;

            Emp bhavana=new Emp();
            bhavana.id=235;
            bhavana.Salary=200000;

System.out.println(lokesh.id+" "+lokesh.Salary+" "+lokesh.ceo);
System.out.println(bhavana.id+" "+bhavana.Salary+" "+bhavana.ceo);
      }
}
```

Output:

```
static block
229 10000 Sundar
235 200000 Sundar
```

Note: We can directly access static variables using class name as mentioned in the below example.

Example5:

```
class Emp
{
      int id;
      int Salary;
      static String ceo;
      static
      {
            ceo="Sundar";
      }
}
public class A
{
      public static void main(String[] args)
      {
            System.out.println(Emp.ceo);
            Emp lokesh=new Emp();
            lokesh.id=229;
            lokesh.Salary=10000;

            Emp bhavana=new Emp();
            bhavana.id=235;
            bhavana.Salary=200000;

System.out.println(lokesh.id+" "+lokesh.Salary+" "+Emp.ceo);
System.out.println(bhavana.id+" "+bhavana.Salary+" "+Emp.ceo);

      }
}
```

Output:

```
Sundar
```

```
229 10000 Sundar
235 200000 Sundar
```

Note: if we change the static variable at anywhere it will be changed for the entire class. It is explained with the example below.

Example6:

```
class Emp
{
      int id;
      int Salary;
      static String ceo;
      static
      {
            ceo="Sundar";
      }
}
public class A
{
      int i;
      public static void main(String[] args)
      {
            System.out.println(Emp.ceo);
            Emp lokesh=new Emp();
            lokesh.id=229;
            lokesh.Salary=10000;

            Emp bhavana=new Emp();
            bhavana.id=235;
            bhavana.Salary=200000;

            bhavana.ceo="navin";

System.out.println(lokesh.id+" "+lokesh.Salary+" "+Emp.ceo);
System.out.println(bhavana.id+" "+bhavana.Salary+" "+Emp.ceo);

      }
}
```

Output:

```
Sundar
229 10000 navin
235 200000 navin
```

Note: We can be able to change the static variable using class name. The changes will be applied for the entire class. Example for this is given below.

Example7:

```
class Emp
{
	int id;
	int Salary;
	static String ceo;
	static
	{
		ceo="Sundar";
	}
}
public class A
{
	int i;
	public static void main(String[] args)
	{
		System.out.println(Emp.ceo);
		Emp lokesh=new Emp();
		lokesh.id=229;
		lokesh.Salary=10000;

		Emp bhavana=new Emp();
		bhavana.id=235;
		bhavana.Salary=200000;

		Emp.ceo="navin";

System.out.println(lokesh.id+" "+lokesh.Salary+" "+Emp.ceo);
System.out.println(bhavana.id+" "+bhavana.Salary+" "+Emp.ceo);

	}
}
```

Output:

```
Sundar
229 10000 navin
235 200000 navin
```

Note: In static methods or in static blocks only static variables are to be used or else we can also declare variables in the static methods or in the static blocks. We can't able to use instant variables in static methos or in static blocks. Also we can't able to assign instant variables to static variable. If we try to do the above than it will raise error as mentioned in the example below.

Example8:

```
public class A
{
	int i; //instant variable
	public static void main(String[] args)
	{
		//using instant variable in the static method
		//will raise an error
		System.out.println(i);
	}
}
```

Output:

```
Exception in thread "main" java.lang.Error: Unresolved compilation
problem: 
	Cannot make a static reference to the non-static field i

	at A.main(A.java:8)
```

Example9:

```
public class A
{
	int i; //instant variable
	static
	{
		//using instant variable in the static block
		//will raise an error
		System.out.println(i);
	}
	public static void main(String[] args)
	{
		System.out.println("main method");
	}
}
```

Output:

```
Exception in thread "main" java.lang.Error: Unresolved compilation
problem: 

	at A.main(A.java:10)
```

Example10:

```
public class A
{
	int i; //instant variable
	// we can't able to assign instant variable
	//to the static variable. It will raise an error
	static int a=i;
	public static void main(String[] args)
	{
		System.out.println("main method");
	}
}
```

Output:

```
Exception in thread "main" java.lang.Error: Unresolved compilation
problem: 

	at A.main(A.java:7)
```

Note: Java block will be executed when you create an object of the class. When you create an object it will calls the constructor either it is default constructor or it is defined constructor. In the first line of the constructor java developers had written code at back end that it will first goes to the java block. After completion of java block it will comes back to the constructor and it will execute the constructor.

Example11:

```
public class A
{
	//Java Block
	{
		System.out.println("java Block");
	}
	public static void main(String[] args)
	{
		System.out.println("main method");
		A obj = new A();
	}
}
```

Output:

```
main method
java Block
```

Example12:

```
public class A
{
	A()
	{
		System.out.println("constructor");
	}
	//Java Block
	{
		System.out.println("java Block");
	}
	public static void main(String[] args)
	{
		System.out.println("main method");
		A obj = new A();
	}
}
```

Output:

```
main method
java Block
constructor
```

Example13:

```
class B
{
	{
		System.out.println("class B Java Block");
	}
}

public class A
{
	A()
	{
		System.out.println("constructor");
	}
	//Java Block
	{
		System.out.println("java Block");
	}
	public static void main(String[] args)
	{
		System.out.println("main method");
		A obj1 = new A();
		B obj2 = new B();
	}
}
```

Output:

```
main method
java Block
constructor
class B Java Block
```

Example14:

```
class B
{
	{
		System.out.println("class B Java Block");
	}
	B()
	{
```

```
            System.out.println("class B constructor");
      }
}

public class A
{
      A()
      {
            System.out.println("constructor");
      }
      //Java Block
      {
            System.out.println("java Block");
      }
      public static void main(String[] args)
      {
            System.out.println("main method");
            A obj1 = new A();
            B obj2 = new B();
      }
}
```

Output:

```
main method
java Block
constructor
class B Java Block
class B constructor
```

Example15:

```
class B
{
      static String ceo;
      static
      {
            System.out.println("class B static block");
            ceo="navin";
      }
      {
            System.out.println("class B Java Block");
      }
      void show()
      {
            System.out.println("show method");
      }
      static void display()
      {
            System.out.println("class B static display method");
      }
      B()
      {
            System.out.println("Class B constructor");
      }
}

public class A
{
      int i;
      static int a;
      static
      {
            System.out.println("class A static block");
      }
      static void display()
      {
            System.out.println("class A static display method");
      }
      void show()
      {
            System.out.println(a);
            System.out.println("show method");
      }
      {
            System.out.println("class A java Block");
      }
      public static void main(String[] args)
      {
            System.out.println(B.ceo);
            System.out.println("main method");
            A.display();
            B.display();
            A obj1=new A();
            obj1.show();
            B obj2=new B();
            obj2.show();
      }
}
```

Output:

```
class A static block
class B static block
navin
main method
class A static display method
class B static display method
class A java Block
0
show method
```

```
class B Java Block
Class B constructor
show method
```

Note: If multiple stack blocks and multiple java blocks are there than one by one will be executed. It follows the order.

Example16:

```
public class A
{
	static
	{
		System.out.println("static block 1");
	}
	{
		System.out.println("Java block 1");
	}
	static
	{
		System.out.println("static block 2");
	}
	{
		System.out.println("Java block 2");
	}
	static
	{
		System.out.println("static block 3");
	}
	{
		System.out.println("Java block 3");
	}
	public static void main(String[] args)
	{
		System.out.println("main method");
		A obj=new A();
	}
}
```

Output:

```
static block 1
static block 2
static block 3
main method
Java block 1
Java block 2
Java block 3
```

Example17:

```
class B
{
	static
	{
		System.out.println("class B static block");
	}
	{
		System.out.println("class B Java block");
	}
}
public class A extends B
{
	{
		System.out.println("class A java block");
	}
	public static void main(String[] args)
	{
		System.out.println("main method");
		A obj1=new A();
		B obj2=new B();
	}
}
```

Output:

```
class B static block
main method
class B Java block
class A java block
class B Java block
```

Example18:

```
class B
{
	static
	{
		System.out.println("class B static block");
	}
	{
		System.out.println("class B Java block");
	}
	B()
	{
		System.out.println("class B constructor");
	}
}
public class A extends B
{
	{
		System.out.println("class A java block");
	}
	A()
	{
		System.out.println("class A constructor");
	}
	public static void main(String[] args)
	{
		System.out.println("main method");
		A obj1=new A();
		B obj2=new B();
	}
}
```

Output:

```
class B static block
main method
class B Java block
class B constructor
class A java block
class A constructor
class B Java block
class B constructor
```