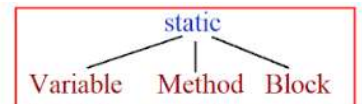


Understanding Static

- The **static keyword** is used in java mainly for memory management.
- It is used to define common functionalities of a java application.
- We may apply static keyword with variables, methods, blocks.
- The static keyword belongs to the class rather than instance of the class.
- The static can be:
 - Variable
 - Method
 - Block
 - Main Method

Understanding static Keyword: static Keyword is mainly used for memory management.

static Variable:



- 1) A variable which is declared inside a class outside any method block or a constructor with the help of static keyword is known as static variable.
- 2) We can access static variable in **three** ways, by using identifier name, by using ClassObject & **directly by using ClassName**
- 3) For static variables JVM will automatically initialize them with their default values.
- 4) For **final** static variables JVM will not initialize them with their default values, it is the responsibility of the programmer.
- 5) **There will be only one copy of static variable available throughout the program.**
- 6) static variables are **not constants** we can change the value of a static variable (Once we have changed the value of a static variable the updated value will be reflected to the entire program).
- 7) **static variables will be initialized at the time of class loading.**
- 8) A static variable can never be a local variable.

static Method

- 1) The method which is declared as static is known as static method.
- 2) We can access static method in **three** ways, by using identifier name, by using ClassObject & **directly by using ClassName**
- 3) We can access a non static variable inside a static method, but with the help of its respective class object

static Block

- 1) In our Java program if we are having a main() and a static block **first priority** will be given to the static block.
- 2) We can write **any number** of static blocks in our program.
- 3) We cannot call static blocks they will be getting executed in the defined order [i.e, from **top to bottom**]
- 4) Before Java 1.5 version we can run a Java program only with the help static block, but after that it is mandatory to write main() in our program to run the java application.
- 5) **final static variables** we are supposed to initialize them at the time of its declaration or only inside a static block anywhere else if you are trying to initialize we will be getting an compile time error.

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.(eg: company name of employees).
- In java applications it is possible to access the static variables either by using the respective class object reference (or) by using respective class name directly.
- Static variables never be 'local variables'.
- JVM executes static members according to their priorities.
- Static block and static variable will have equal priorities, so these execute in defined order.
- If a static variable and a local variable is having same name, Then compiler will first search for local variable and then static variable.

- For the static variables it is not required to perform initialization explicitly jvm will always provide default values.
- If we declare a static variable as final then 100% we should perform initialization explicitly whether we are using or not otherwise we will get compile time error.
- For final static variables JVM won't provide any default values, JVM will provide default values only for static variables.
- Final static variables can be initialized inside a static block. (any where else we will be getting compile time error)

Static Method

- If you apply static keyword with any method, it is known as static method
- 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.
- In java applications it is possible to access the static methods either by using the respective class object reference (or) by using respective class name directly.

Restrictions for static method:

- The static method can not use **non static data** member or call non-static method directly.

Static Method Vs Instance Method

Static Method	Instance Method
A method i.e. declared as static is known as static method.	A method i.e. not declared as static is known as instance method
Object is not required to call static method.	Object is required to call instance methods.
Non-static (instance) members cannot be accessed in static context (static method, static block and static nested class) directly.	static and non-static variables both can be accessed in instance methods.

Static Block:

- Is used to initialize the static data member.
- We can not invoke a static block, rather JVM invokes the static block at the time of class loading.
- It is executed before main method at the time of class loading.
- We can define more than one static block in the java program.

public class Demo

```
{
    static
    {
        System.out.println("hi Static block is Invoked");
        System.exit(0);
    }
    public static void main(String[] args)
    {
        System.out.println("hi from main method");
    }
}
```

```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     int x;
6     static int y; // 3
7
8     ClassA()
9     {
10         x++; //1 1 1
11         y++;
12         System.out.println("Instance Variable x : "+x); //1
13         System.out.println("Static Variable y : "+y); // 3
14         System.out.println("-----");
15     }
16     public static void main(String[] args)
17     {
18         new ClassA(); // 1st Object
19         new ClassA(); // 2nd Object
20         new ClassA(); // 3rd Object
21     }
22 }
```

```
^ Instance Variable x : 1
  Static Variable y : 1
-----
Instance Variable x : 1
Static Variable y : 2
-----
Instance Variable x : 1
Static Variable y : 3
-----
```

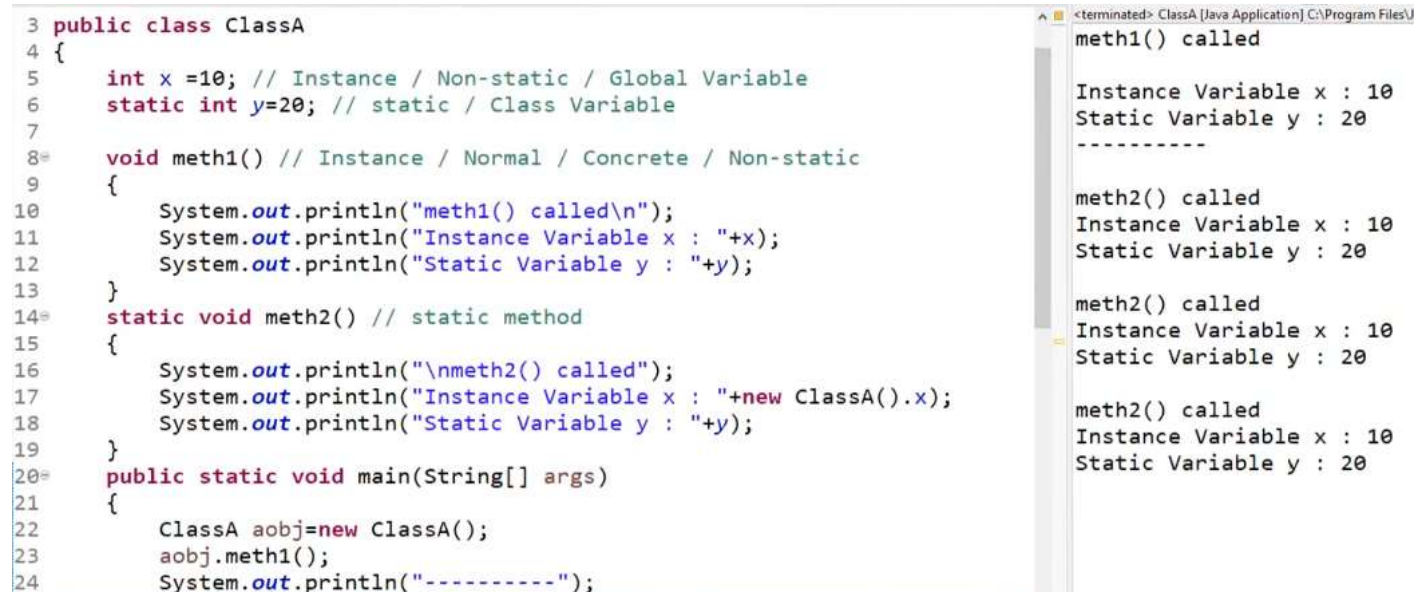
Q) Can we access a non- static variable inside a static method?

Ans) Yes, we can access a non-static variable inside a static method but with the help of its respective class object.

INSTANCE VARIABLE / NON-STATIC VARIABLE / GLOBAL VARIABLE

STATIC VARIABLE / CLASS VARIABLE

INSTANCE METHOD / NORMAL METHOD / CONCRETE METHOD / NON-STATIC METHOD



```
3 public class ClassA
4 {
5     int x =10; // Instance / Non-static / Global Variable
6     static int y=20; // static / Class Variable
7
8     void meth1() // Instance / Normal / Concrete / Non-static
9     {
10         System.out.println("meth1() called\n");
11         System.out.println("Instance Variable x : "+x);
12         System.out.println("Static Variable y : "+y);
13     }
14     static void meth2() // static method
15     {
16         System.out.println("\nmeth2() called");
17         System.out.println("Instance Variable x : "+new ClassA().x);
18         System.out.println("Static Variable y : "+y);
19     }
20     public static void main(String[] args)
21     {
22         ClassA aobj=new ClassA();
23         aobj.meth1();
24         System.out.println("-----");
```

Execution Output:

```
<terminated> ClassA [Java Application] C:\Program Files\U
meth1() called
Instance Variable x : 10
Static Variable y : 20
-----
meth2() called
Instance Variable x : 10
Static Variable y : 20
meth2() called
Instance Variable x : 10
Static Variable y : 20
meth2() called
Instance Variable x : 10
Static Variable y : 20
```

```

25     meth2(); // by using identifier name
26     aobj.meth2(); // by using class Object
27     ClassA.meth2(); // ***** byusing ClassName *****
28 }
29
30
31 }

```

```

1 package com.pack1 ;
2
3 public class ClassA
4 {
5     static
6     {
7         System.out.println("static block called");
8     }
9     public static void main(String[] args)
10    {
11        System.out.println("main() called");
12    }
13 }

```

```

<terminated> ClassA [Java Application] C:\Pro
static block called
main() called

```

```

1 package com.pack1 ;
2
3 public class ClassA
4 {
5
6     public static void main(String[] args)
7     {
8         System.out.println("main() called");
9     }
10    static
11    {
12        System.out.println("static block called");
13    }
14 }

```

```

<terminated> ClassA [Java Application] C:\Pro
static block called
main() called

```



```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     static
6     {
7         System.out.println("1st static block called");
8     }
9     public static void main(String[] args)
10    {
11        System.out.println("main() called");
12    }
13    static
14    {
15        System.out.println("2nd static block called");
16    }
17 }
```

<terminated> ClassA [Java Application] C:\Program F
1st static block called
2nd static block called
main() called

```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     final static int x=50; // final variables are compile-time constants
6     static
7     {
8         System.out.println("1st static block called : "+x);
9     }
10    public static void main(String[] args)
11    {
12        System.out.println("main() called : "+x);
13    }
14    static
15    {
16        System.out.println("2nd static block called : "+x);
17    }
18 }
```

1st static block called : 50
2nd static block called : 50
main() called : 50

Whenever we are declaring a static variable as final, we need to initialize its value at the time of its declaration or inside a static block

Once you declare a final variable you cannot modify it.

```

1 package com.pack1 ;
2
3 public class ClassA
4 {
5     final static int x; // final variables are compile-time constants
6     static
7     {
8         x=50;
9         System.out.println("1st static block called : "+x);
10    }
11    public static void main(String[] args)
12    {
13        //x=50; // C.E
14        System.out.println("main() called : "+x);
15    }
16    static
17    {
18        //x=100; // C.E because of final keyword
19        System.out.println("2nd static block called : "+x);
20    }
21 }

```

```

1 package com.pack1 ;
2
3 public class ClassA
4 {
5     static int x=ClassA.meth1();
6
7     static int meth1()
8     {
9         System.out.println("static meth1() called");
10        return 100;
11    }
12    static
13    {
14        System.out.println("static block called");
15    }
16    public static void main(String[] args)
17    {
18        System.out.println("main() called");
19    }
20 }

```

static meth1() called
static block called
main() called

Static method is executed only whenever we are calling it.

Static variable and static block executed directly before main method.

Static variable and static block will be having same priorities.

They will be getting executed before main() in top to bottom order.

```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     static int x=ClassA.meth1();
6
7     static int meth1()
8     {
9         System.out.println("static meth1() called");
10        return 100;
11    }
12    static
13    {
14        System.out.println("static block called");
15    }
16    public static void main(String[] args)
17    {
18        System.out.println("main() called");
19    }
20 }
```

```
static meth1() called
static block called
main() called
```

```
1 package com.pack1 ;
2
3 public class ClassA
4 {
5     static int meth1()
6     {
7         System.out.println("static meth1() called");
8         return 100;
9     }
10    static
11    {
12        System.out.println("static block called");
13    }
14    static int x=ClassA.meth1();
15    public static void main(String[] args)
16    {
17        System.out.println("main() called");
18    }
19 }
```

static block called
static meth1() called
main() called