**What is a Static Keyword in Java?**

In Java, keywords are the reserved words that cannot be used as identifiers. In total there are 57 keywords in Java. One among them is “**Static**“.

Below topics are covered in this article:

* Introduction to Static keyword in Java
* Applications of Static keyword
  + Static Block
  + Static Variable
  + Static Method
  + Static Classes

**Introduction to Static Keyword in Java**

In Java, **static keyword** is mainly used for memory management. It can be used with variables, methods, blocks and nested classes. It is a keyword which is used to share the same variable or method of a given class. Basically, static is used for a constant variable or a method that is same for every [instance of a class](https://www.edureka.co/blog/instance-variable-in-java/). The main method of a class is generally labeled static.

In order to create a static member (block, variable, method, nested class), you need to precede its declaration with the keyword *static*. When a member of the class is declared as static, it can be accessed before the objects of its class are created, and without any object reference.

In Java programming language, static keyword is a non-access modifier and can be used for the following:

* Static Block
* Static Variable
* Static Method
* Static Classes

Let’s get into the details of each of these methods with the help of an example.

**Applications of Static Keyword**

Let’s first understand how static block is used in the Java programming language.

**Static Block**

If you need to do the computation in order to initialize your **static variables**, you can declare a static block that gets executed exactly once, when the class is first loaded. Take a look at the below Java program to understand the usage of Static Block.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | // Java program to demonstrate the use of static blocks  import java.util.\*;  public class BlockExample{  // static variable  static int j = 10;  static int n;    // static block  static {  System.out.println("Static block initialized.");  n = j \* 8;  }    public static void main(String[] args)  {  System.out.println("Inside main method");  System.out.println("Value of j : "+j);  System.out.println("Value of n : "+n);  }  } |

When you execute the above program, static block gets initialized and displays the values of the initialized variables.

**Output**:

|  |  |
| --- | --- |
| 1  2  3  4 | Static block initialized  Inside main method  Value of j:10  Value of n : 80 |

Now that you know how static block works, let’s move further and see what are static variables and how it is helpful.

**Static Variable**

When you declare a variable as static, then a single copy of the variable is created and divided among all [objects](https://www.edureka.co/blog/java-tutorial/#obj) at the [class level](https://www.edureka.co/blog/java-objects-and-classes/). Static variables are, essentially, global variables. Basically, all the instances of the class share the same static variable. Static variables can be created at class-level only.

Now let’s understand this with the help of an example.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | // Java program demonstrate execution of static blocks and variables    import java.util.\*;    public class VariableExample  {  // static variable  static int j = n();    // static block  static {  System.out.println("Inside the static block");  }    // static method  static int n() {  System.out.println("from n ");  return 20;  }    // static method(main !!)  public static void main(String[] args)  {  System.out.println("Value of j : "+j);  System.out.println("Inside main method");  }  } |

When you execute the above program, it will execute static block and the variable in order as defined in the above program.

**Output:**

|  |  |
| --- | --- |
| 1  2  3  4 | from n  Inside the static block  Value of j: 20  Inside main method |

Having understood this, let’s dive deeper into this article on Static keyword in Java and know what are Static methods and nested classes.

**Programming & Frameworks Training**

**Static Methods**

When a method is declared with the *static* keyword, it is known as a static method. The most common example of a static method is the *main( )* method.  Methods declared as static can have the following restrictions:

* They can directly call other static methods only.
* They can access static data directly.

Now let’s understand static methods with the help of an example

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | // java program to demonstrate restriction on static methods  public class StaticMethodExample  {  // static variable  static int j = 100;    // instance variable  int n = 200;    // static method  static void a()  {  a = 200;  System.out.println("Print from a");    // Cannot make a static reference to the non-static field b  n = 100; // compilation error    // Cannot make a static reference to the  // non-static method a2() from the type Test  a2(); // compilation error    // Cannot use super in a static context  System.out.println(super.j); // compiler error  }    // instance method  void a2()  {  System.out.println("Inside a2");  }    public static void main(String[] args)  {  // main method  }  } |

In the above examples, you can see how the restrictions are imposed on the static methods and also how you are allowed to use super keyword in the static context. That was all about Static Methods. Now let’s see what are nested classes.

**Static Class**

A class can be made **static** only if it is a nested class. 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. Let’s take an example to understand how it works

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | public class NestedExample{  private static String str= "Tca"  //Static class  static class MyNestedClass{  //non-static method  public void disp(){  System.out.println(str);  }  }  public static void main(String args[]){  NestedExample.MyNestedClass obj = new NestedExample.MyNestedClass();  obj.disp();    } |

When you execute the above code, your output looks like:

Tca