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“. In this article, I will give you a brief insight into how static keyword in Java is applicable for various aspects of programming.

**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. 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**

**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.

**// Java program to demonstrate the use of static blocks**

import java.util.\*;

public class A{

// 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);

}

}

Output:

Static block initialized

Inside main method

Value of j:10

Value of n : 80

**Static Variable**

When you declare a variable as static, then a single copy of the variable is created and divided among all objects at the class level. 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.

import java.util.\*;

public class A

{

// 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:

from n

Inside the static block

Value of j: 20

Inside main method

**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.

**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

public class NestedExample{

private static String str= "king"

//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();

}

}

Output:

king