**Static-**

* It is used for memory management.
* It can be applied to variable, method, inner class and static block.
* It means single copy storage.
* It’s belong to class instance.

**Static variable-**

* A variable which is defined with static keyword called as “static variable.”
* It is used to refer the common property of all the objects.
* Static variables get loaded into memory at the time of class loading.
* A single copy of that field is created and shared among all instances of that class.

**How to declare static variable.**

Example- static int a=5;

Note-:- Local variables cannot be static.

**How to access static variable.**

There are two ways to access the static variables.

1. By using class name
2. By using object name.
3. Directly in same class

**Program for static variables**

**package** com.test;

**public** **class** StaticDemo {

**static** **int** *a* = 5;

**public** **static** **void** main(String[] args) {

StaticDemo staticDemo = **new** StaticDemo();

System.***out***.println(StaticDemo.*a*); // by using class name

System.***out***.println(staticDemo.*a*); // by using object name

System.***out***.println(*a*); // directly in same class

}

}

Output-

5

5

5

In this example, we are calling static variable by using statickeyword class name, also calling it by using object obj and directly as its in same class as shown in above program.

**Why it is called as single copy storage?**

Note- Static members get loaded into memory as soon as (StaticDemo) read this line. Non static members gets loaded into memory after reading new StaticDemo()- this line.

**package** com.test;

**public** **class** StaticDemo {

**int** a = 5;

**static** **int** *b* = 5;

**public** **static** **void** main(String args[]) {

StaticDemo staticDemo1 = **new** StaticDemo();

System.***out***.println("non static>>" + staticDemo1.a++);

System.***out***.println("static>>" + staticDemo1.*b*++);

StaticDemo staticDemo2 = **new** StaticDemo();

System.***out***.println("non static>>" + staticDemo2.a++);

System.***out***.println("static>>" + staticDemo2.*b*++);

StaticDemo staticDemo3 = **new** StaticDemo();

System.***out***.println("non static>>" + staticDemo3.a++);

System.***out***.println("static>>" + staticDemo3.*b*++);

StaticDemo staticDemo4 = **new** StaticDemo();

System.***out***.println("non static>>" + staticDemo4.a++);

System.***out***.println("static>>" + staticDemo4.*b*++);

}

}

Output-

non static>>5

static>>5

non static>>5

static>>6

non static>>5

static>>7

non static>>5

static>>8

Note- We cannot call non-static member from static member because static variables stored into memory before object creation and non-static variables stored into memory after object creation.

**How to access static members from non-static members.**

**Following program shows you.**

**package** com.test;

**public** **class** StaticDemo {

**void** x2() {

System.***out***.println("This is non static method");

*x1*(); // calling static method from non static method.

}

**static** **void** x1() {

System.***out***.println("This is static method");

}

**public** **static** **void** main(String args[]) {

StaticDemo staticDemo = **new** StaticDemo();

staticDemo.x2();

}

}

Output-

This is non static method.

This is static method.

**Static method-**

* If you define any method with static keyword then it is called as static method.
* It belongs to class rather than object of class.
* It loads into memory before object creation.
* It can access only static data member only.

**How to call static method:**

We discuss how to call static method during method topic please go through it.

1. By using class name
2. By using object name.
3. Directly in same class

Note: - Main method is static method.

**package** com.test;

**public** **class** StaticDemo {

**static** **void** x1() {

System.***out***.println("This is static method");

}

**public** **static** **void** main(String args[]) {

StaticDemo.*x1*();

}

}

Output-

This is static method.

**Static block-**

* It is group of statements that are executed when class is loading into memory by Classloader.
* It is widely used to create the static resource.
* We cannot access non-static variable into static block.
* It is always executed first.

Example-1

**public** **class** Test {

**static** {

System.***out***.println("this is the 1st static block...");

}

**public** **static** **void** main(String[] args) {

System.***out***.println("this is main method..");

}

}

Here, we will get first output is “This is 1st static block” because it is executed first than main method.

Output-

this is the 1st static block...

this is main method..

Example-2

**public** **class** StaticExample {

**static** {

System.***out***.println("this is the 1st static block...");

}

**static** {

System.***out***.println("this is the 2nd static block...");

}

**static** {

System.***out***.println("this is the 3rd static block...");

}

**public** **static** **void** main(String[] args) {

System.***out***.println("this is main method..");

}

}

output

this is the 1st static block...

this is the 2nd static block...

this is the 3rd static block...

this is main method..

Note- Outer class cannot be static but inner class can be static.

Constructor cannot be static.