Static

Before understanding the static keyword in java, one must be aware of the different members a class can have.

A class in java consist of the following members

Class Demo {

Static variables

Static blocks

Static methods

Static members of a class

Instance variables

Instance block

Instance method

Constructors

Non static or Instance members of a class

}

After getting to know different members of a class one must also know Static members belong to class and Non-static members belong to object.

Class

Static variables
Static blocks
Static methods

Object

Instance variables

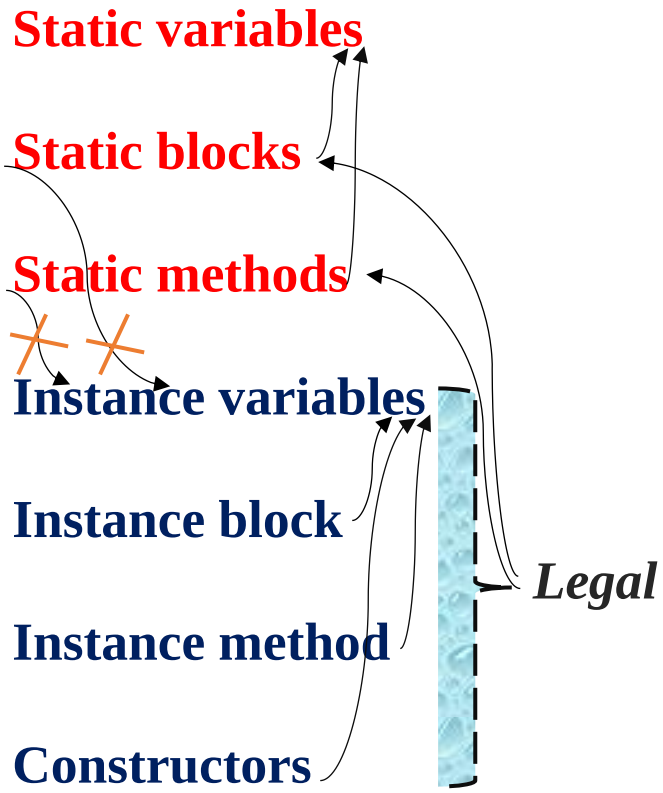
Instance blocks

Instance methods

Constructors.

You are now aware of what belongs to a class and what belongs to an object but this knowledge is not sufficient to understand when to use static and when not to use. To understand that one must learn few rules of static.

RULES OF STATIC:



Static Variables can be accessed by static as well as instance variables.

Instance variables can be accessed by only instance members.

Static members cannot access instance variables.

In the Java programming language, **the keyword *static* indicates that the particular member belongs to a type itself, rather than to an instance of that type**.

This means that only one instance of that static member is created which is shared across all instances of the class.

The keyword can be applied **to variables, methods, blocks and nested class.**

Static variables: Static variable in Java is a variable which belongs to the class, not to object and initialized only once at the start of the execution. These variables will be initialized first, before the initialization of any instance variables.

- A single copy to be shared by all instances of the class
- A static variable can be accessed directly by the class name and doesn't need any object

Syntax: <*class-name*>.<*variable-name*>

Static method: Static method in Java is a method which belongs to the class and not to the object. A static method can access only static data. It cannot acces s non-static data (instance variables).

- A static method can call only other static methods and cannot call a non-static method from it.
- A static method can be accessed directly by the class name and doesn't need any object.
- A static method cannot refer to "this" or "super" keywords in anyway.

Syntax: <*class-name*>.<*method-name*>

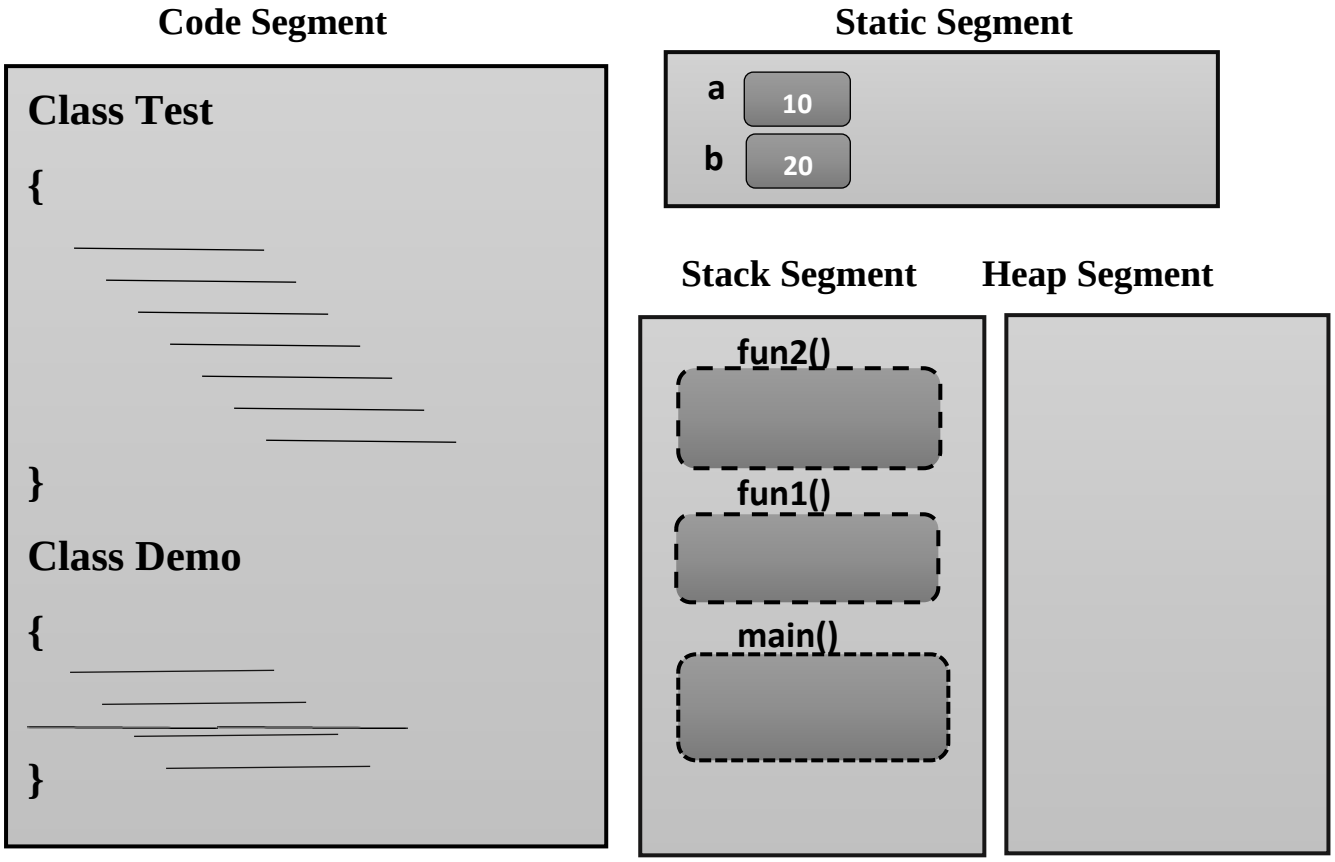
Static block: The static block is a block of statement inside a Java class that will be executed after object creation and before call to the constructor. A **static block helps to initialize the static data members**, just like constructors help to initialize instance members.

Let us explore static in detail with the help of codes.

Have a look at the code given below which consists Of all the members.

```
class Test
{
    static int a,b;
    static
    {
        System.out.println("Inside static block");
        a = 10;
        b = 20;
    }
    static void fun1()
    {
        System.out.println("Inside static method");
    }
    int x,y;
    {
        System.out.println("Inside instance block");
    }
    void fun2()
    {
        System.out.println("Inside instance method");
    }
    Test()
    {
        System.out.println("Inside constructor");
        x = 30;
        y = 40;
    }
}
class Demo
{
    public static void main(String[] args)
    {
        Test.fun1();
        Test t = new Test();
        t.fun2();
    }
}
```

Let us now trace the code and understand it from the memory perspective.



Explanation:

When the first class which consists of main() is loaded onto the code segment, the first check that is made JVM is static variables. Since there are no static variables inside class Demo, It will now check for static blocks and again there is no static block inside class Demo. Now the JVM will search for main(). Now main() gets called and its stack frame gets created on stack segment. Now main() starts executing and inside main() the very first line is call to fun1(). But fun1() is present inside class Test which is not yet loaded inside code segment. JVM will now ask class loader to load Test class. Once the class Test is loaded, JVM will now trace the class and look for static variables. Class Test consists of 2 static variables a,b which are allocated memory on static segment and JVM gives default values. Now JVM checks for static block. Since the Test class consists of static block, it gets executed. Now control goes back to the main(). The first line is call to fun1(). Control now goes to Test class and stack frame of fun1() is created on stack segment. Once this fun1() gets executed, control leaves the fun1() and its stack frame gets deleted. Control comes back to main() and the next line is object creation. Assignment means start from RHS and the moment JVM encounters new keyword, it allocates a block of memory. It now goes to class Test and allocates memory for instance variables x,y. It also gives default values to x,y.

After object creation and before call to the constructor, instance block gets executed. After this, constructor gets exceuted. Now the reference "t" is assigned to the object. The last line inside main() is call to fun2(). Once a method gets called, its stack frame gets created and then it gets executed.

Once there are no more lines left inside fun2() control leaves the fun2() and its stack frame gets deleted. Control now goes back to where it came from that is to main(). Inside main() there are no more lines left so its stack frame also gets deleted.

In this way, a complete java program executes.