Static and Non-Static Block

Static Block

```java
static {
    // Code here
}
```

- A `static` block is executed automatically **before** the execution of the `main()` method.
- A class can contain **multiple** static blocks.
    - **Execution Order**: Top to bottom.

Execution Steps

1. Identification of static members from top to bottom.
2. Execution of static variable assignments and static blocks in order.
3. Execution of `main()` method.

Example

```java
class StaticBlockExample {
    static int x = 10; // Step 1.1, Step 2.1

    static {
        func(); // Step 2.2
        System.out.println("FSB"); // Step 2.3
    }

    public static void main(String[] args) {
        func(); // Step 3.1
        System.out.println("main"); // Step 3.2
    }

    public static void func() {
        System.out.println(y); // Step 2.2.1 (0), Step 3.1.1 (20)
    }

    static {
        System.out.println("SSB"); // Step 2.4
    }

    static int y = 20; // Step 2.5
}
```

Output:

```
0
FSB
SSB
20
main
```

Multiple Static Blocks

```java
class Demo {
    static {
        System.out.println("First Static Block");
    }
    public static void main(String[] args) {
        System.out.println("Main Method");
    }
    static {
        System.out.println("Second Static Block");
    }
}
```

Output:

```
First Static Block
Second Static Block
Main Method
```

Direct vs Indirect Read

Direct Read

> Accessing a variable directly in a static block.

```java
class Demo {
    static int x = 10;
    static {
        System.out.println(x); // Direct Read
    }
}
```

Indirect Read

> Accessing a variable via method call in a static block.

```java
class Demo {
    static int x = 20;
    static {
        m1(); // Indirect Read
    }
    public static void m1() {
        System.out.println(x); // Indirect Read
    }
}
```

Without Static or Main

```java
class Demo {
    static {
        System.out.println("Hi");
        System.exit(0);
    }
}
```

Another Trick (Static variable initialization via method)

```java
class Demo {
    static int x = m1();
    public static int m1() {
        System.out.println("Hi");
        System.exit(0);
        return 10;
    }
}
```

RIWO (Read Indirect Write Only)

- If a variable is in RIWO state, **direct read is not allowed.**
- Attempting to do so leads to **Illegal Forward Reference Error**.

Example

```java
class Test {
    static {
        System.out.println(x); // ❌ Error: Illegal Forward Reference
    }
    static int x = 10;
}
```

Non-Static Block

```java
{
    // Code here
}
```

- Executes **each time an object is created.**
- Execution order: Top to bottom.
- **Priority:** Non-static block → Constructor

Example

```java
class NonStaticBlockExample {
    int x = 10;
    {
        func();
        System.out.println("FIB");
    }
    NonStaticBlockExample() {
        System.out.println("Constructor");
    }
    public static void main(String[] args) {
        NonStaticBlockExample obj = new NonStaticBlockExample();
        System.out.println("main");
    }
    public void func() {
        System.out.println(y); // Output: 0
    }
    {
        System.out.println("SIB");
    }
    int y = 20;
}
```

Output:

```
0
FIB
SIB
Constructor
main
```

Combined Example

```java
class StaticAndNonStaticBlocks {
    static int x = 10;
    int a = 100;

    static {
        s_func();
        System.out.println("FSB");
    }
    {
        ns_func();
        System.out.println("FIB");
    }
    StaticAndNonStaticBlocks() {
        System.out.println("Constructor");
    }
    public static void main(String[] args) {
        s_func();
        StaticAndNonStaticBlocks obj = new StaticAndNonStaticBlocks();
        obj.ns_func();
        System.out.println("main");
    }
    public static void s_func() {
        System.out.println(y);
    }
    public void ns_func() {
        System.out.println(b);
    }
    static {
        System.out.println("SSB");
    }
    {
        System.out.println("SIB");
    }
    static int y = 20;
    int b = 200;
}
```

Output:

```
0
FSB
SSB
20
0
FIB
SIB
Constructor
200
main
```