L38

Thursday, February 3, 2022 11:52 PM

Non-static Init Blocks, Static Init Blocks

Non-static init blocks

Q1. In a Java class, the fields are normally initialized in the constructors. However, there is another way of initializing them in an unnamed block also called **initializer block** or a **non-static initializer block.**

For example, in the below code you will notice that the field value is initialized to 3 in an initializer block.

Once the code is executed you will notice that the code snippets in the initializer blocks are executed (in the order in which they appear in the code) before the code in the constructor is executed. This is because the Java compiler copies all initializer blocks in the order they appear, into every constructor.

Since we are creating two instances of class A using the new keyword, the constructor is called twice and during the constructor call first, all the initializer blocks are executed.

If there is a block of code which has to be executed before the code inside any of the constructors is executed, such code can be placed in an initializer block.

```
package q11289;
public class NonStaticInitBlockDemo {
    public static void main(String[] args) {
        A a1 = new A();
        A a2 = new A();
    }
}
class A {
    private int value;
    //below is an example of non-static initialization block
    {
        value = 3;
        System.out.println("In non-static init block 1");
    }
    public A() {
        System.out.println("In constructor");
        System.out.println("value = " + value);
    }
    //below is an example of another non-static initialization block
    {
        System.out.println("In non-static init block 2");
    }
}

```

Static init blocks

Q1. Initializer blocks which are prefixed with the static keyword are called **static initializer block.**

Like the **non-static** initializer blocks, a class can have any number of static initializer blocks and they will be executed in the order in which they appear in the code.

The code inside the **non-static** initalizer block is copied inside the constructor and executed during every invocation of the constructor.

However, a **static** initializer block is executed only once when the class is loaded in memory. It has nothing to do with constructor invocation. Even if the constructor is not called to create an object instance, as long as the class is loaded the static initializer blocks get executed.

Since the classes are loaded only once by the system class loader, these static initializer blocks get executed only once.

See and retype the below code. On execution you will notice that code snippets in the static initializer blocks are executed only once even though we create two instances of class A.

```
package q11290;
public class StaticInitBlockDemo {
    public static void main(String[] args) {
        A a1 = new A();
        A a2 = new A();
    }
}
class A {
    //below is an example of static initialization block
    static {
        System.out.println("In static init block 1");
    }
    public A() {
        System.out.println("In constructor");
    }
    //below is an example of another static initialization block
    static {
        System.out.println("In static init block 2");
    }
}

```