|  |  |
| --- | --- |
| ***Static block*** | ***instance block*** |
| Known only as [**static**](http://www.javamadesoeasy.com/2015/05/static-keyword-in-java-variable-method.html) **initialization block** in java. | Also known as **non-static initialization block** in java. |
| **static blocks** executes **before instance blocks** in java. | **instance blocks** executes **after static blocks** in java**.** |
| Only static variables can be **accessed** inside **static block** | Static and non-static variables (instance variables) can be **accessed** inside **instance block.** |
| **static blocks** can be used for **initializing** static **variables**  **or**  calling any static **method** in java. | **instance blocks** can be used for initializing instance **variables**  ***or***  calling any instance **method** in java. |

When instance block is executed?

Instance block will be executed **only once for each object during its creation**. So, the number of times an instance block is executed signifies the number of objects created in the program. The execution of instance block depends only on the object creation and not on the execution of a constructor.

Static block in java is executed **every time when a class loads**. This is also known as Static initialization block. Static block in java initializes when class load into memory , it means when JVM read the byte code.

Example using static and initializer block

public class one extends two {

static {

System.out.println("inside satic block");

}

one() {

System.out.println("inside constructor of child");

}

{

System.out.println("inside initialization block");

}

public static void main(String[] args) {

new one();

new one();

System.out.println("inside main");

}

}

class two{

static {

System.out.println("inside parent Static block");

}

{

System.out.println("inside parent initialisation block");

}

two() {

System.out.println("inside parent constructor");

}

}