What is Block in Java?

- A **block in Java** is a set of code enclosed within curly braces { } within any class, method, or constructor.
- It begins with an opening brace ({) and ends with an closing braces (}).
- Between the opening and closing braces, we can write codes which may be a group of one or more statements.

Types of blocks in Java

There are three types of blocks in Java as

1. Local block
2. Instance initialization block (Non-static initialization block)
3. Static initialization block

1. Local Block in Java

- A block defined inside a method, block, or constructor is called **local block in Java**.
- Local block is also called inner block in Java.
- It will be executed whenever the enclosing method, constructor, or block is executed.
- It is not similar to the instance block. It cannot be static.
- We can declare local block inside a method, constructor or block and can also be nested.

Example:

```
public class LocalBlock {
        public static void main(String[] args) {
                int x=13;
                System.out.println("x = "+x);

                //inner block
                {
                        int y = 7;
                        System.out.println("y = "+y);
                }
        }
}
```

Scope of Variables in Local block

- All the variables declared inside a block are local variables. Therefore, they can be accessed only within that block.
- That is, the scope of these local variables will be accessed only within the block. We cannot call these variables from outside the block.

Example:

```
public class LocalBlock {
	public static void main(String[] args) {
		int x=13;
		System.out.println("x = "+x);

		//inner block
		{
			int y = 7;
			System.out.println("y = "+y);
		}
		System.out.println("y = "+y); // compile time error
	}
}
```

2. Instance Initialization Block

- An instance initialization block is also known as non-static block in Java.
- It is used to write that logic which we want to execute during the object creation. It is mainly used to initialize the instance data member.
- It is also used to initialize variables.
- It will be executed after the execution of the static block if any static block is declared inside the class.
- Static and Non-static variables can be accessed inside the non-static block.
- Instance block executes before the constructor only when an instance of the class is created.

Syntax of Instance block:

```
{
 // logic here.
}
```

Example 1:

```
public class InstanceInitializerBlock {
	//constructor
	public InstanceInitializerBlock() {
		System.out.println("In Instance Intializer Block Constructor...");
	}

	//instance initializer block
	{
		System.out.println("Instance Initializer block executed...");
	}

	public static void main(String[] args) {
		System.out.println("In Main Method");

		InstanceInitializerBlock iib = new InstanceInitializerBlock();

		System.out.println("In Main Method Again");
	}
}
```

Multiple Instance Block

Example 2:

```
class Test {
	Test() {
		System.out.println("0 argument constructor");
	}

	Test(int a) {
		System.out.println("1 argument constructor = "+a);
	}

	{
		System.out.println("Instance block-01");
	}
	{
		System.out.println("Instance block-02");
	}

	public static void main(String[] args) {
		System.err.println("In Main Method");

		new Test();
		new Test(10);

		System.err.println("In Main Method Again");
	}
}
```

Example 3:

```
class InstanceBlockDemo {
	int ivar;

	public InstanceBlockDemo() {
		System.out.println("Inside constructor");
	}

	// initializer block
	{
		ivar = 20;
		System.out.println("First initializer block");
	}

	public static void main(String args[]) {
		InstanceBlockDemo ibd = new InstanceBlockDemo();
		System.out.println("Inside main method");
		System.out.println("ivar = " + ibd.ivar);
	}
}
```

Difference between Instance Block and Constructor block

Instance Block	Constructor Block
The **static blocks** are executed at the time of **class loading**.	A **Constructor** will be executed while **creating an object** in Java.
The **static blocks** are executed before running the **main () method**.	A **Constructor** is called while creating an object of a class.
The **static blocks don't have any name** in its prototype.	The **name of a constructor** must be always the **same name as a class**.
If we want any logic that needs to be executed at the time of class loading that logic needs to placed inside the **static block** so that it will be executed at the time of class loading.	A **Constructor** is called only once for an object and it is **called as many times** as we can create an object. i.e The constructor gets executed automatically when the object is created.

3. Static Initialization Block:

- Static block is used to initialize the static data member.
- A static initialization block is mostly used for **changing the default value of static variables.** A static variable in Java is a variable that belongs to the class and is initialized during the loading of the class into memory.
- Static block is executed before the main method at the time of class loading.
- If you need to do the computation in order to initialize your static variables, you can declare a static block that gets executed exactly once, when the class is first loaded.
- We can’t access non-static variables in the static block.
- A class can have multiple Static blocks, which will execute in the same sequence in which they have been written into the program.

Syntax of static block:

```
class ClassName
{
        static
        {
                //logic here
        }
}
```

Example 1:

```
public class Main {
        static {
                System.out.println("Hi, I'm a Static Block!");
        }

        public static void main(String[] args){
                System.out.println("Hi, I'm a Main Method!");
        }
}
```

Example 2:

```
import java.util.Scanner;
public class Solution{
        static Scanner input = new Scanner(System.in);

        public static boolean flag = false;
        public static int B = input.nextInt();
        public static int H = input.nextInt();

        static{
                if(B > 0 && H > 0)
                flag = true;
                else
                System.out.println("Breadth and height must be positive");
        }


        public static void main(String[] args){
                if(flag){
                        int area = B * H;
                        System.out.print(area);
                }
        }
}
```

Static Block	Instance Block
A static block is also known as a static initialization block.	An instance block is also known as instance initialization block or non-static block.
The Static blocks execute before instance blocks in java.	The instance block executes after the static blocks.
Only static variables can be accessed inside the static block, if we try to access any non-static variable (instance variables) inside the static block it will throw an error stating non-static variable cannot be referenced from a static context.	Both the static and non-static variables can be accessed inside the instance block.
The static block can be used for initializing static variables or calling any static method in java.	The instance blocks can be used for initializing instance variables or calling any instance method in java.
Static blocks execute during the loading of its dot class (.class) file in memory.	The instance block executes only when the instance of the class is created, not called during the loading of its .class file in memory in java.
Inside a static block we cannot use the this keyword	Inside an instance block we can use this keyword.