Methods, Variable and Block

A method is something like a subprogram, which when invoked returns a value. In other words, a method is a collection of words or commands that joined to perform a function. Once this function or task is completed, it sends a result back to the caller.
It can even complete an operation without returning a value, and allows us to use the same code without actually having to type the entire code again and again. Methods are also known as instance members.

- **Methods :**
  - **Syntax :-**

    [Modifiers] return_type Method_name(Arguments1, array2---)
    {

    }

    Class can contain two types of methods:
    - **Instance method**
    - **Static method**

- **Example :-**

```
public class InstanceMethod {
        int a = 10;
        static int b = 20;
        void qc() {
                System.out.println("Hello Queue Codes");
                System.out.println(a);
                System.out.println(b);
        }

        static void m1() {
                System.out.println("Hello Queue Codes");
                System.out.println(b);
        }

        public static void main(String[] args) {
                InstanceMethod i = new InstanceMethod();
                System.out.println(i.a);
                System.out.println(i.b);
                i.qc();
                i.m1();
        }
}
```

Instance Method	Static Method
When you defined a method inside a class without a static keyword, it is called the instance method or the non-static method	When you defined a method with static keyword then it is called the static method
Instance method must be called by you explicitly. You can call using reference variables which contain the object. For Example: (Hello h = new Hello())	Static methods must be called by you explicitly. You can call using the following methods: a) With classname b) With reference variable without initialization. For Example : A a = null; c) With reference variable which contains object

- **Variables:**
  There are two types of variables, Local and Global. Read the following to get more details about them
- **Global Variables :**
  - **Global** variables are initialized automatically
  - It is defined outside of method
  - A static keyword can be applied to global variable
  - It's scope is across class and can be used anywhere
- **Example :**

```
	public class A {
		int a;
		String str;
		public static void main(String[] args) {
			A aa = new A();
			System.out.println(aa.a); //prints 0
			System.out.println(aa.str); // prints null
			System.out.println(aa); //prints address of a2

			}
	}
```

- **Local Variables :**
  - Variable declared inside method or constructor or any block are called local variables
  - The scope of the local variable is within the method or constructor or block were it is defined

- Memory will be allocated for the local variable whenever the enclosing block gets executed
- Local variables is not equivalent to instance variable
- Local variables cannot be static
- Local variables cannot be referenced with class name or reference variable
- Local variables will not be initialized by the JVM automatically , like instance and static variable
- If you use local variables without initializing, you get compile time errors like ‘variable’ x might not have been initialized.
- Local variables can be final, primitive or reference.

```
public class AA {

	static int ctr = 0;
	int i = 100;
	{
		//This is block
		System.out.println("Before change in local block");
		System.out.println("ctr = "+ctr);
		System.out.println("i = "+i);
		int ctr = 2, i = 100;
		System.out.println("");
		System.out.println("After change in local block");
		System.out.println("ctr = "+ctr);
		System.out.println("i= "+i);
	}

	void display2() {
		System.out.println("In another method");
		System.out.println("ctr = "+ctr);
		System.out.println("i = "+i);
	}

	public static void main(String[] args) {
		AA a = new AA();
		System.out.println(" ");
		a.display2();
	}
}
```

- **Block** defined inside a method or block or constructor is called a local block
- **A** local block will be executed whenever the enclosing methods or constructor or block is executed

- **Local** block are not equivalent to instance block
- **Local** block cannot be static
- **You** can write local blocks inside a method or constructor or block, and it can be nested.

```
public class Hello {

        static int a = 31;
        static int b;
        static void math(int x) {
                System.out.println("X = "+x);
                System.out.println("A = "+a);
                System.out.println("B = "+b);
        }
        static {
                System.out.println("Static block initialised.");
                b = a* 4;
        }
        public static void main(String[] args) {
                System.out.println("Before static method b= "+b);
                math(42);
        }
}
```

- **Output --**

```
Static block initialised.
Before static method b= 124
X = 42
A = 31
B = 124
```