

JAVA



What you learn ?

Java

➤ Variable

➤ Method

➤ Call by value

➤ Call by reference

Variables



In Java, a variable is a named container that stores a value of a particular data type. Variables are used to store data that can be accessed and manipulated by the program.

Variable is a name of memory location.

- **Local Variables:** These are variables that are declared within a method or block of code and are only accessible within that method or block.
- **Instance Variables:** These are variables that are declared within a class but outside any method, and are accessible to all methods of the class.

Instance variable are also variable of object commonly known as field or property. They are referred as object variable. Each object has its own copy of each variable and thus, it doesn't effect the instance variable if one object changes the value of the variable.

- **Class/Static Variables:** These are variables that are declared as static within a class, and are shared among all instances of the class.
- In Java, variables must be declared with a data type, such as int, double, boolean, etc. The syntax for declaring a variable is: **dataType variableName = value;**

Variables




```
class Person{  
    //instance variables  
    String name="Java";  
    int age=20;  
    double weight;
```

//The static variables must be declared as a class member in the class.

```
static int num=5; //static variable
```

```
public static void main(String args[]){  
    //local variable  
    String firstName="Apurva";  
}  
  
}
```

Scope of Variables



Static Variables: Static variables are declared using the static keyword, and their value is shared among all instances of a class. The scope of a static variable is the entire class, and it can be accessed using the class name or direct.

Instance Variables: Instance variables are declared inside a class, but outside any method or constructor, and are accessed using an object reference. The scope of an instance variable is the entire class, and it can be accessed using an object reference.

Local Variables: Local variables are declared inside a method or block of code and are accessible only within that method or block. The scope of a local variable is limited to the method or block in which it is declared.


Scope Variables



Scope of instance variable depends on the access-modifiers (public, private, default).

- If variable is declared as private then it is accessible within class only.
- If variable is declared as public then it is accessible for all and throughout the application.
- If variable is declared as default then it is accessible within the same package.

Methods

 In Java, a method is a collection of statements that perform a specific task. Methods are declared inside a class, and they can be called from within the same class or from other classes.

basic syntax of a method declaration:

```
return_type method_name(parameter_list) {  
    // method body  
}
```

return_type: The data type of the value that the method returns. If the method does not return a value, the return type should be void.

method_name: The name of the method, which should be a meaningful and descriptive name that reflects what the method does.

parameter_list: The list of input parameters that the method takes. If the method does not take any input parameters, the parameter list should be empty.

method_body: The collection of statements that perform the task of the method.

Types of Methods

1) **Predefined methods** : Java provides many predefined methods, also known as built-in methods, that are part of the standard Java library and can be used without the need for additional coding. These methods are grouped into different classes, depending on their functionality. Some of the most commonly used classes and their methods are:

String Class: The String class provides a set of methods for working with strings. Some of the commonly used methods are **length()**, **charAt()**, **substring()**, **indexOf()**, **toLowerCase()**, **toUpperCase()**, **trim()**, **startsWith()**, **endsWith()**, and **replace()**.

Math Class: The Math class provides a set of methods for performing mathematical operations. Some of the commonly used methods are **abs()**, **ceil()**, **floor()**, **max()**, **min()**, **pow()**, **sqrt()**, **random()**, and **round()**.

System Class: The System class provides a set of methods for working with the system. Some of the commonly used methods are **out.println()**, **out.print()**, **in.read()**, **currentTimeMillis()**, **gc()**, and **exit()**.

Types of Methods



2) User-defined method is a method that is created by the programmer to perform a specific task. These methods are also known as custom or user-created methods. User-defined methods are declared inside a class and can be called by other methods in the same class or from outside the class.

Method definition and call



```
public class UserDefinedMethodExample {  
  
    public static void main(String[] args) {  
        int num1 = 10;  
        int num2 = 20;  
        int sum = calculateSum(num1, num2); //method call  
        System.out.println("The sum is: " + sum); // built-in method call  
    }  
  
    //method definition  
    public static int calculateSum(int num1, int num2) {  
        int sum = num1 + num2;  
        return sum;  
    }  
}
```

Method Category



Methods can be categorized into two types: static methods and instance methods.

- **Static Methods:** Static methods are methods that belong to a class rather than an instance of a class. Static methods can be called using the class name, without the need to create an instance of the class.
- **Instance Methods:** Instance methods, also known as non-static methods, are methods that are called on an instance of a class. They can access both static and instance members of the class, as they have access to the instance-level data.

Static Method



```
public class StaticMethodExample {  
  
    public static void main(String[] args) {  
        int num = 5;  
        int square = calculateSquare(num);  
        System.out.println("The square of " + num + " is " + square);  
    }  
  
    public static int calculateSquare(int num) {  
        return num * num;  
    }  
}
```

Call by value




- ✓ There is only call by value in java, not call by reference.
- ✓ If we call a method passing a value, it is known as call by value.
- ✓ The changes being done in the called method, is not affected in the calling method.

```
class Operation{  
    int data=50;  
  
    void change(int data){  
        data=data+100;//changes will be in the local variable only  
    }  
  
    public static void main(String args[]){  
        Operation op=new Operation();  
        System.out.println("before change "+op.data);  
        op.change(500);  
        System.out.println("after change "+op.data);  
    }  
}
```

Output: before change 50
after change 50

Call by Reference

 Though Java is strictly call by value when we pass the reference of the object it creates a copy of the reference and then passes it as value to the method. The copy reference also points to the same address so all the changes also reflect in the main method this is the main difference between call by value and call by reference in java.

```
class Ref{
    int number=10;

    // pass object as parameter
    public static void increment(Ref pb){
        pb.number = pb.number+1; // increment variable by 1
        System.out.println("value in method: "+pb.number);
    }
    public static void main(String[] args) {
        Ref pb=new Ref(); // pb is an object of class Ref
        System.out.println("value before method call: "+pb.number);
        increment(pb); // pass object of the class Ref
        System.out.println("value after method call: "+pb.number);
    }
}
```

Output:

```
value before method call: 10
value in method: 11
value after method call: 11
```



Thanks

Anirudha Gaikwad