4. Functions / Methods

Date
Page No.

⇒ A method in Java is a named block of code designed to perform a specific task. You define a method once and can reuse it multiple times in your program. Methods help in organizing code and avoiding repetition.

Syntax :-

optional

```
access_modifier non-access_modifier return-type method-name(parameter_list){
    // method body
    return statement; // only if return_type is not void
}
```

- Component of a Method - Explained.

Part	Description
access_modifier	Controls where the method is accessible (e.g. public, private, protected)
non-access_modifier	Optional modifiers like (static, final, or abstract)
return_type	Type of data the method returns (e.g. int, void, String)
method_name	A descriptive name (camelCase is the convention)
parameter_list	Input values passed to the method, written as type-name pairs (eg, int a, String name)
method body	The code block that runs when the method is called.
return statement	Sends back a value (used only if return_type is not void)

- Static VS Non-static Methods

Static Method	Non-static Method
(i) Declared with the static keyword	(i) Declared without static keyword
(ii) Called using class name or directly.	(ii) Called using an object of the class.
(iii) Belongs to the class	(iii) Belongs to an instance of the class.

- Return Statement Rules.
  (i) Used only in methods with non-void return type.
  (ii) Syntax : return value;
  (iii) The returned value must match the return type.

- # Java Method Types

(1) Method with No parameters & No return value

Syntax :

```
void methodName( ){
    // code to perform a task.
}
```

Explanation :

(1) Takes no input
(2) Returns nothing (void).
(3) Used when you want to perform action like printing, displaying a msg

(datatype ... v) — is used to many parameters.
(any character with 3 dots)

Date
Page No.

2. Method with parameters & NO Return value.

Syntax :

```
void methodName(type1 parm1, type2 parm2,...){
    //use parameters to perform a task.
}
```

Explanation :

(i) Takes input using parameters.
(ii) Returns nothing (void).
(iii) useful when you want to perform a task based on inputs, but don't need to return anything.

(3) Method with NOparameters & Return value

Syntax :

```
returnType methodName(){
    //logic
    return value; // value must match returnType
}
```

Explanation :

(i) Takes no input.
(ii) Returns a single value (like int, String, etc).
(iii) Used when you want to calculate or fetch something without requiring user input.

4. Method with parameters & Return Value

Syntax:

```
returnType methodName(type1 param1, type2 param2, ...){
    // use parameters
    return value;
}
```

Explanation:

(i) Takes input using parameters.
(ii) Returns a single value based on the inputs.
(iii) This is the most powerful and commonly used method type.

Method overloading

→ It means define multiple methods with the same name in the same class, but with different parameters (type, number, or order)

Syntax:

```
void show ( ){ }                          ← main method
void show (int a) { }                     ← overloading
Void show (String name) { }               ← overloading
Void show (int a, String name) { }        ← overloading
Void show (String name, int a){ } //different order   ← overloading
```

Use Cases:

When you want to perform a similar operation but handle different types or numbers of inputs.