DAY-21

10 July 2023 15:06

METHODS

WHAT IS METHOD IN JAVA ?

- A METHOD IS BLOCK OF CODE OR COLLECTION OF STATEMENTS OR A SET OF CODE GROUPED TOGETHER TO PERFORM A CERTAIN TASK OR OPERATION
- IT IS USED TO ACHIEVE THE RESUABILITY OF CODE
- WE WRITE A METHOD ONCE AND USE IT MANY TIMES
- WE DO NOT REQUIRE TO WRITE CODE AGAIN AND AGAIN
- IT ALSO PROVIDES THE EASY MODIFICATION AND READABILITY OF CODE, JUST BY ADDING OR REMOVING A CHUNK OF CODE
- THE METHOD IS EXECUTED ONLY WHEN WE CALL OR INVOKE IT

UNSTRUCTURED WAY OF WRITING PROGRAM

- IN THIS, WE WRITE ALL THE TASK INSIDE THE MAIN METHOD DUE TO WHICH SAME PIECE OF CODE IS REPEATED MULTIPLE TIMES AND THE PROGRAM BECOMES INEFFICIENT.

EXAMPLE :

```
PUBLIC STATIC VOID MAIN(STRING[] ARGS)
{
        -----ADD---
        ----SORT---
        ---ADD---
        ----SORT----


}
```

STRUCTURTED WAY OF WRITING PROGRAM

- IN THIS , WE DECIDE THE TASK INTO MULTIPLE METHODS AND WHENEVER WE WANT TO IMPLEMENT ANY TASK, WE WILL CALL THAT METHOD FROM THE MAIN METHOD
- HERE , WE DEFINE TASK ONCE IN THE METHOD AND CAN IMPLEMENT ANY NUMBER OF TIMES.

SYNTAX FOR METHOD DECLARATION

```
ACCESS_MODIFIER MODIFIER RETURNTYPE METHODNAME (ARGUMENT LIST)
{

        //METHOD BODY

}
```

```
public static void main(String[] args)
{

     method body
}
```

```
ACCESS_MODIFIER MODIFIER RETURNTYPE METHODNAME (ARGUMENT LIST)
{

        //METHOD BODY

}
```

Public : ACCESS MODIFIER

Static : MODIFIER

Void : RETURN TYPE

Main : METHOD NAME

(Strings[] args] --> argument /parameter list

- **ACCESS MODIFIER** : IT DEFINES THE ACCESS TYPE OF THE METHOD
- **MODIFIER** : IT IS USED TO SPECIFY THE TYPE OF METHOD STATIC /NON STATIC
- **RETURN TYPE** : METHOD MAY RETURN A VALUE
- **NAME OF METHOD** : THIS IS THE METHOD NAME. THE METHOD SIGNATURE CONSIST OF THE METHOD NAME AND THE PARAEMETER LIST
- **PARAMETER LIST** : THE LIST OF PARAMETERS , IT IS THE TYPE , ORDER AND NUMBER OF PARAMETER OF A METHOD. THESE ARE OPTIONAL , METHOD MAY CONTAIN XERO PARAMETERS
- **METHOD BODY** : THE METHOD BODY DEFINES WHAT THE METHOD DOES WITH THE STATEMENT

RULES TO WRITE METHOD NAME

- METHOD NAME SHOULD START WITH LOWER CASE.
- ONLY SPECIAL CHARACTER UNDERSCORE (_) CAN BE USED
- SHOULD NOT START WITH DIGIT
- SPACE IS NOT ALLOWED
- CAN BE ALPHANUMERIC
- KEYWORDS CAN NOT BE METHOD NAME

NOTE : IF METHOD NAME CONSIST OF 2 OR MORE WORDS , CONSECUTIVE WORDS FIRST LETTER SHOULD BE ALWAYS UPPERCASE

```
import java.util.Scanner;

class Test1
{
	public static void main(String[] args)
	{
		System.out.println("Main starts");

		System.out.println("Main ends");

	}

	public static void printHi()
	{
		System.out.println("Hi");
	}
}
```

METHOD CALLING

- IF WE ARE WRITING A USER DEFINED METHODS IN OUR PROGRAM AND IF WE WANT TO EXECUTE THEM , WE HAVE TO CALL THOSE METHODS EXPLICITLY IN THE MAIN METHOD
- IF WE DON'T CALL THE METHOD INSIDE MAIN METHOD, COMPILER WILL COMPILER SUCCESSULLY BUT USER DEFINED METHODS WILL NOT

EXECUTE.

Example : 1

```
import java.util.Scanner;

class Test1
{
	public static void main(String[] args)
	{
		System.out.println("Main starts");
		System.out.println("Main ends");

	}

	public static void printHi()
	{
		System.out.println("Hi");
	}
}
```

Example : 2

```
import java.util.Scanner;

class Test1
{
	public static void main(String[] args)
	{
		System.out.println("Main starts");

		printHi();
		printBye();
		System.out.println("Main ends");

	}

	public static void printHi()
	{
		System.out.println("Hi");
	}

	public static void printBye()
	{
		System.out.println("Bye");
	}
}
```

Example : 3

```
import java.util.Scanner;

class Test1
{
	public static void main(String[] args)
	{
		System.out.println("Main starts");

		m1();
		m2();
		System.out.println("Main ends");
```

```
        }

        public static void m1()
        {
                System.out.println("m1 starts");
                m2();
                System.out.println("m1 ends");
        }

        public static void m2()
        {
                System.out.println("m2 starts");
                System.out.println("m2 ends");
        }
}
```

Example 4 :

```
import java.util.Scanner;

class Test1
{
        public static void main(String[] args)
        {
                System.out.println("Main starts");

                m1();
                System.out.println("Main ends");

        }

        public static void m1()
        {
                int a = 10, b=20;
                int c = a+b;

                System.out.println(c);
        }

}
```

TYPES OF METHODS

THERE ARE 4 TYPES OF METHOD CALLING

1. METHOD NOT ACCEPTING PARAMETER AND NOT RETURNING VALUE
2. METHOD NOT ACCEPTING PARAMETERS BUT RETURNING VALUE
3. METHOD ACCEPTING PARAMETER LIST BUT NOT RETURNING VALUE
4. METHOD ACCEPTING PARAEMTER LIST AND RETURNING VALUE

RETURN TYPES

- VOID
- INT
- FLOAT
- DOUBLE
- CHAR
- STRING
- BOOLEAN