Smit IT Institute
Gives you wings to fly high

Topic: Method

Method:

Method is a bock of code that executes a targeted logic.

It's is a set of instructions and executes when required.

3 pillars are essential: **1. Method Declaration. 2.Method Definition. 3.Method Calling.**

Syntax:

```
return_type method_name()
{
        Code statements;

}
```

Code Below:

```
public class Methods
{
	void talab1() //method declaration
	{
		System.out.println("How are you?");//method definition

	}

	void talab2()//method-->It Executes a targeted logic
	{
		System.out.println("I am fine");//method definition

	}

	void add()
	{
		int a=100;
		int b=400;
		int c=a+b;
```


```
		System.out.println("Addition is:"+c);

	}

	void sub()
	{

		int a=50;
		int b=20;
		int c=a-b;
		System.out.println("Substraction is:"+c);
	}

	void loop()
	{
		int i=1;
		for(;i<=10;)
		{
			System.out.println("i is:"+i);
			i++;

		}

	}



	public static void main(String[] args)
	{
		Methods  obj1=new Methods();

		obj1.talab1();//method calling

		obj1.talab2();//method calling

		obj1.add();
		obj1.sub();
		obj1.loop();




	}

}
```

```
OutPut:

How are you?
I am fine
Addition is:500
Substraction is:30
i is:1
i is:2
i is:3
i is:4
i is:5
i is:6
i is:7
i is:8
i is:9
i is:10
```

^^

*Parametrized method

1.A method where values are passed via object is called as

Parameterized method.

2.The parameters of any method are its own local variable.

3.These parameters cannot be accessed outside the method block.

4.Syntax:

```
return type  method_name(data_type var1,data_type var2)
{
        Code Statements
}
```

5.Each parameter is separated by comma in method declaration.

6.The Number of Parameters declared should match with the

number of parameters or else it gives miss-match errors.

Code Below:

```
public class ParameterizedMethod
{
	void add(int a,int b)// a=5,b=10  method declare
	{
		System.out.println("Addition is: "+(a+b));//method definition

	}

	void multiplication(int a,int b)//a=4, b=17
	{

		System.out.println("Multiplication is: "+(a*b));
	}

	void sub(long p, long q,long r)//p=500,q=434,r=732
	{
		System.out.println("Minus values are: "+(p-q-r));

	}

	public static void main(String[] args)
	{
		ParameterizedMethod obj1=new ParameterizedMethod();

		obj1.add(5,10);//method calling
		obj1.multiplication(4, 17);
		obj1.sub(500, 434, 732);


		}

}
```

OutPut:

```
Addition is: 15
Multiplication is: 68
Minus values are: -666
```

Smit IT Institute
Gives you wings to fly high

Topic:Var Methods

1.When we want to pass multiple number of arguments but the parameter is just one or is not enough to save the multiple values, to solve this issue we use a special method.

2.This special method is called as variance method,

3.Syntax

```
return_type method_name(data_type ...variable)
        {
                Code Statements;

        }
```

Topic:-For Each Loop:

a. It is also called as **Advance For Loop.**

b. It does not have condition.

c. It continues to execute until the set of data is over.

d. Its Execution flow is from start to end.

e. For Each Loop has its own local variable.

Syntax:

```
for(int i: data_containing_variable)
{
        System.out.println(i);

}
```

e.g:

```
void dis(int ...a)
        {
                for(int i:a)
                {
                        System.out.println(i);
                }

        }


*obj1.dis(12,34,56,686);
```

Code Below:

```
public class ParameterizedMethod
{
	void value1(int ...t)//variance method
	{

		//For Each Loop
		for(int i:t)
		{
			System.out.print(i+" ");//9 9 9 9 9 9 9 9 9
		}
	}

	void value2(long ...r)
	{
		for(long j:r)// for(inti;condition;inc/dec)
		{
			System.out.print(j+" ");

		}

		}
	 void loop(int e)//e=15
	{

		for(int i=1;i<=e;i++)
		{
			System.out.print(i+" ");

		}

	}
	public static void main(String[] args)
	{
		ParameterizedMethod obj1=new ParameterizedMethod();


		System.out.println("\n-----For Each Loop------\n");

		obj1.value1(9,9,9,9,9,9,9,9,9,9);
		System.out.println("\n--------------------\n");

		obj1.value2(23,17,19,33,45,92,81,113,73);
		System.out.println("\n--------------------\n");

		obj1.loop(15);

	}
}
```

OutPut:

```
-----For Each Loop------

9 9 9 9 9 9 9 9 9 9
--------------------

23 17 19 33 45 92 81 113 73
---------------------

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

****

Topic:Returning value method:

1. Syntax:

```
return_type method_name(parameters)
                {
                        return value;// return is a keyword use to return a value;
                }
```

2.A returning method does not display its returning value within the block.

3.It returns the value from the control from which it was called.

4.Returned value can be stored in another variable via object calling method statement.

5.Returned value can be also displayed directly by object calling method statement by mentioning them after concatenation in the print function.

6. Returning method can be parameterized as well as non-parameterized.

Code Below:

```
public class ReturningMethod
{
```

```
int display()
{
	int a=34;
	return a;

}

long display2()
{
	long b=12345;
	return b;

}

double display3()
{
	double db=1234.56;
	return db;

}

char display4()
{
	char ch='h';
	return ch;
}

boolean display5()
{
	return true;

}


int add()
{
	int a=33;
	int b=8;
	int c=a+b;

	return c;

}

long multi(long p,long q)//p=12,q=4
{
	long r=p*q;//12*4
	return r;//48

}

double sub(double s,double d,double h)//s=100,d=33,h=17
{
```


```
                double sub=s-d-h;//100-33-17
                return sub;

        }



    public static void main(String[] args)
    {
            ReturningMethod obj1=new ReturningMethod();


            System.out.println("int value returned is: "+obj1.display());
            int a2;

            a2=obj1.display();
            System.out.println("a2 is:"+a2);


System.out.println("\nlong value returned is: "+obj1.display2());

long b2;
            b2=obj1.display2();
            System.out.println("b2 is:"+b2);


            System.out.println("\ndouble value returned is: "+obj1.display3());
            double db2;
            db2=obj1.display3();

            System.out.println("db2 is:"+db2);

            System.out.println("\nchar value returned is: "+obj1.display4());
            char ch2;
            ch2=obj1.display4();


            System.out.println("ch2 is:"+ch2);

            System.out.println("\nboolean value returned is: "+obj1.display5());
            boolean bl;
            bl=obj1.display5();
            System.out.println("bl is:"+bl);

            //*******************************************************

            System.out.println("\nAddition is: "+obj1.add());
            System.out.println("\nMultiplication is: "+obj1.multi(12, 4));
            System.out.println("\nSubstraction is: "+obj1.sub(100, 33, 17));
```


```
            }

}
```

```
OutPut:
int value returned is: 34
a2 is:34

long value returned is: 12345
b2 is:12345

double value returned is: 1234.56
db2 is:1234.56

char value returned is: h
ch2 is:h

boolean value returned is: true
bl is:true

Addition is: 41

Multiplication is: 48

Subtraction is: 50.0
```

****

Topic:Method-Overloading.

1.Method-Overloading is a concept where method name is same but the parameters

are different such case is called as Method-Overloading.

2.Different Parameters:

a. Number of Parameters can be different.

b. If Number of Parameters are same then their data-types can be different.

eg:

```
void display(int a)
                {



                }
                void display(double a)
                {



                }

                void display(int a,int b)
                {



                }
```

Coding Below:

```
public class MethodOverloading
{
            void add(int a,int b)
            {
```


```
            System.out.println("Addition is:"+(a+b));
        }

        void add(int a,int b,int c)
        {

            System.out.println("Addition is:"+(a+b+c));
        }
        void add(double a,double b)
        {
            System.out.println("Double Addition is: "+(a+b));

        }


    public static void main(String[] args)
    {
        MethodOverloading obj1=new MethodOverloading();

        obj1.add(12, 20);
        obj1.add(10,20,30);
        obj1.add(51.78, 83.5);

    }

}
```

OutPut:

```
Addition is:32
Addition is:60
Double Addition is: 135.28
```