Smit IT Institute
Gives you wings to fly high

What is a method?

A **method** in object-oriented programming (OOP) is a procedure associated with a message and an object. An object consists of *state data* and *behaviour*; these compose an *interface*, which specifies how the object may be utilized by any of its various consumers. A method is a behaviour of an object.

Method Overloading:

1.Method overloading, on the other hand, refers to differentiating the code used to handle a message based on the parameters of the method. If one views the receiving object as the first parameter in any method then overriding is just a special case of overloading where the selection is based only on the first argument.

2.It is a Concept used in a single class.

3.Multiple methods declaration within a single class each differentiated with respect to its number of parameters or data types of parameters is called as Method Overloading.

```
public class Main5
{
                void data()
                {
                        System.out.println("Method 1");
                }
                void data(int a)//23
                {
                        System.out.println("Method 2: "+a);
                }
                void data(int a,int b)//Number of parameters
                {
                        System.out.println("Method 3:"+a+"<-->"+b);
                }
                void data(char a,char b)//Data Types of Parameters are  changed.
                {
                        System.out.println("Method 4:"+a+"<-->"+b);
                }

        public static void main(String[] args)
        {
                Main5 obj=new Main5();

                obj.data();
                obj.data(23);
                obj.data(43, 71);
                obj.data('r', 'p');
        }
}
```

Output:

```
Method 1
Method 2: 23
Method 3:43<-->71
Method 4:r<-->p
```

Method Overriding:

1.Method overriding and overloading are two of the most significant ways that a method differs from a conventional procedure or function call. Overriding refers to a subclass redefining the implementation of a method of its superclass.

2.Inheritance is required to Perform Method Overriding.

3.It is a concept where a method with same name and number of parameters with match data-type are present in Parent class as well as Child Class.

4.Object of Any particular class can access the methods of the its respective method().

5. Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.

6.Method overriding is used for runtime polymorphism.

```
class A3
{
    void data(int a,int b)
    {

        System.out.println("Class A3");
    }
}
class B3 extends A3
{
    void data(int a,int b)
    {

        System.out.println("Class B3");
    }

}

class C3 extends B3
{
    void data(int a,int b)
    {
        System.out.println("Class C3");
    }


}
public class Main6
{
    public static void main(String[] args)
    {
        C3 obj=new C3();
        obj.data(5,10);
        B3 obj2=new B3();
        obj2.data(10,20);
        A3 obj3=new A3();
        obj3.data(1,2);
        }
}
```

```
Class C3
Class B3
Class A3
```

Smit IT Institute
Gives you wings to fly high

this Keyword

Q:What is ‘this’ keyword and explain its use?

There can be a lot of usage of **Java this keyword**. In Java, this is a **reference variable** that refers to the current object.

Uses are as follows:

1.this can be used to refer current class instance variable and to differentiate between local and instance variable.

Scope of Local and Instance Variable

Local Variable:

1.A variable declared inside a block is called as the local variable of that particular block (method, constructor, etc.)

2.Local Variable is accessible only with the block in which it has been declared. Outside access is not possible.

Instance Variable:

1.It is also called as class variable.

2.Its declared in class area but not inside any other method block or any other block.

3.Its Scope is throughout the class.

4.It can be accessed inside any other method block.

E.g:

```java
public class Variables
{
	int a= 100;//instance variable
	int b= 200;

	void values(int a,int b)// a=17,b=33 Local Variables of method
	{
		System.out.println(" values:\n");
		System.out.println("local a is:"+a);//17
		System.out.println("local b is:"+b);//33
	}

	void values2()
	{
		System.out.println("\nvalues 2\n");
		System.out.println("instance a is:"+a);//100
		System.out.println("instance b is:"+b);//200

	}

	public static void main(String[] args)
	{
		Variables obj=new Variables();

		obj.values(17,33);
		obj.values2();
	}

}
```

Output:

```
values:

local a is:17
local b is:33

values 2

instance a is:100
instance b is:200
```

*this keyword for Variable

```
public class Main7
{
     int a;
     int b;

     void value1(int a,int b)//a=17,b=33
     {
     Note:- this keyword refers to the instance variable
of class.
          this.a=a;// Differentiating between instance and
local variables  when their names are same.
          this.b=b;

     }
     void value2()
     {
          System.out.println("instance a is:"+a);//17
          System.out.println("instance  b is:"+b);//33
     }
     public static void main(String[] args)
     {
          Main7  obj=new Main7();
          obj.value1(17, 33);
          obj.value2();
     }

}
```

Output:

```
instance a is:17
instance b is:33
```

2.this can be used to invoke current class method (implicitly) from with a method of that particular class only.

*this keyword for method().

1."this" keyword is used to call another method of the same class from within a method of that particular class only.
2.this.method_name() cannot call the method of another class.
3.Its scope is within the class.
4.syntax:

```
void method()
{
        this.method_name(parameter1,parameter2,);

        statements;
}
```

```
public class Main8
{

        void method1()
        {

                System.out.println("method 1");
        }

        void method2(int a,double b)// a=19. b='73.5'
        {
                this.method1();
                System.out.println("\nMethod 2");
                System.out.println("a is:"+a);
                System.out.println("b is:"+b);

        }
        void method3(int a,double b,char ch)//a=11.b=17.9,ch='p'
        {
                this.method2(19, 73.5);
                System.out.println("\nMethod 3");
                System.out.println("a is:"+a);
                System.out.println("b is:"+b);
                System.out.println("c is:"+ch);
```

```
	}
	void method4(int a,char b)
	{
		this.method3(11, 17.9,'p');
		System.out.println("\nMethod 4");
		System.out.println("a is:"+a);
		System.out.println("b is:"+b);

	}

	public static void main(String[] args)
	{
		Main8 obj=new Main8();


		obj.method4(11,'p');


	}

}
```

Output:

```
method 1 →Method1 called from method2 using this.method1();



Method 2 →Method2 called from method3 using this.method2(19,73.5);

a is:19
b is:73.5

Method 3  →Method3 called from method4 using this.method3(11,17.9.'p');
a is:11
b is:17.9
c is: p

Method 4  →Method4 called from object.
a is:11
b is: p
```

3.this() can be used to invoke current class constructor.

*this keyword for constructor().

Constructor
1."this" keyword is used to call another constructor of the same class from with a constructor of that particular class.
2.Syntax:

```
class_constructor()
{
        this(parameter1,parameter2, etc.);
        statements;
}
```

3. this() can be parameterized as well as non-parameterized.
4.this() should be the first line of code within any other constructor.
5.It reduces the number of objects.

```
public class Main9
{
            Main9()
            {

                  System.out.println("\nNormal Constructor\n");
            }
            Main9(int a)
            {
                  this();//// calling normal constructor
                  System.out.println("\n Constructor 1:");
                  System.out.println("a is:"+a);
            }
            Main9(int a,char c)//a=100,c='t'
            {
                  this(233);// calling constructor 1
                  System.out.println("\n Constructor 2:");
                  System.out.println("a is:"+a);
```

```
			System.out.println("c is:"+c);
		}
		Main9(int a,double db,char c)//a =17,db=73.5, c='s'
		{
			this(100,'t');// calling constructor 2
			System.out.println("\n Constructor 3:");
			System.out.println("a is:"+a);
			System.out.println("db is:"+db);
			System.out.println("c is:"+c);
		}
		Main9(char a, char b,char c)
		{
			this(17,73.5,'s');//this constructor, calling 
constructor 3
			System.out.println("\n Constructor 4:");
			System.out.println("a is:"+a);
			System.out.println("db is:"+b);
			System.out.println("c is:"+c);

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

		Main9 obj4=new Main9('s','r','p');//Calling Constructor 4


	}

}
```

Output:

```
Normal Constructor


Constructor 1:
a is:233

Constructor 2:
a is:100
c is:t

Constructor 3:
a is:17
db is:73.5
c is:s

Constructor 4:
a is:s
db is:r
c is:p
```

Smit IT Institute

Gives you wings to fly high