6-5-2020

18 Wednesday

8th week
049-316

February

Use of 'this' Keyword in Java

→ this is a reference variable that refers to the current object in java language.

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

→ this keyword invoke current class method.

→ this keyword can be passed as argument in method calling.

→ this keyword can be passed as argument in Constructor calling.

→ this keyword can be used to return the current class instance

Example:-

```
class Test{
    int id;
    Test(int id)
    {
        id = id;
    }
    void display()
    {
        System.out.println("id has value :"+id);
    }
    public static void main(String args[])
    {
        Test t = new Test(10);
        t.display();
    }
}
```

this.id

10

In above example, to resolve the ambiguity, we have to use this.id in place of id because class Test has variable named id and constructor also has the variable id as argument.

If we are not using `this.id=id;` then the output value printed will be '0'.

But if we use `this.id=id` in the Constructor then the value of id printed as output is 10.

Now, we see that `this()` can be used to call current class constructors as :—

```
class Test {
        Test (int x, int y, int z)
        { this(x,y); //calling Constructor with 2 Args
System.out.println("The value of z is: "+z);
        }
        Test (int x, int y)
        { this(x); //calling Constructor with 1 Arg,
System.out.println("The value of y is:"+y);
        }
        Test (int x)
        {
System.out.println("The value of x is:"+x);
        }
}
```

Notes

```
class DemoThis
{
    public static void main(String args[])
    {
        Test obj = new Test(110,120,130);
    }
}
```

// Calling Constructor with three argument.

Let us see the output :-

```
javac DemoThis.java
java DemoThis
```

```
The value of x is : 110
The value of y is : 120
The value of z is : 130
```

As we create object obj of Test class with three parameters 110, 120, 130, constructor of Test is called having three parameters.

This constructor's first line of code `this(x,y);` calls the second constructor of Test having two arguments.

This constructor's first line of code `this(x);` further calls the third constructor of Test having single argument.

Thus the first line execution Code is :- The value of x is : 110

Then second line of execution is The value of y is : 120

And lastly The value of z is : 130.

Thus in the above program, the value of x is printed from the Constructor having Single argument.
The value of y is being printed from the Constructor having two arguments. and
the value of z is being printed from the Constructor having three arguments.

Now: example of using this for calling methods.

```
class Test
{   int a=10;
    void m1()
    {
System.out.println("Display from m1");
System.out.println("The value of a is:"+this.a);
        this.m2(); // method called
    }   void m2(){System.out.println("Display from m2");
                  System.out.println("The val of a:"+this.a);
                  }
}
class DemoThis1
{
    public static void main(String []args)
    {
        Test t = new Test();
        t.m1();
    }
}
```

Output of the above program is :-

```
Display from m1()
The value of a is: 10
Display from m2()
The value of a : 10
```

Example:-

```
class Test {
    int a, b;
    Test(int a, int b)
    { this.a = a; //class variable a is storing
      this.b = b; //local variable's value of a.
    }
    void m1()
    { System.out.println("-- Display from m1()");
      System.out.println("The value of a:" + this.a);
    }
    void m2()
    { System.out.println("Display from m2()");
      System.out.println("The value of b:" + this.b);
    }
}
class DemoThis2
{
    public static void main(String args[])
    {
        Test t1 = new Test(15, 16);
        t1.m1();
        t1.m2();
    }
}
```

FEBRUARY

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

MARCH

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Output will be

Display from m1()
The value of a : 15
Display from m2()
The value of b : 16

Whenever, we have same variable names inside a class and inside any method or constructor, then we have to use this.variable for variables inside the class (instance variables).

Since, instance variables gets ~~created while~~ memory while object creation inside the heap of the object's memory.

Example:-

```
class Test{
    int a,b;
    Test(int x, int y)
    { a = x;
      b = y;
    }
void m1()
  { System.out.println("Val of a:"+a);
    System.out.println("Val of b:"+b);
  }
}
```

Notes

25 February
Wednesday

```
class DemoThis3{
    public static void main(String args[])
    {
        Test t1=new Test(5,7);
        t1.m1();
    }
}
```

In above program, a and b are instance variables of class Test. In order to access it we have not given this.a but JVM automatically performs this task for us. Since there is no any ambiguity between instance variable name and local variable names used in the program.