Course Name - Object Oriented Programming using Java

Lecture 12 – This keyword, use of objects as parameter & methods returning objects

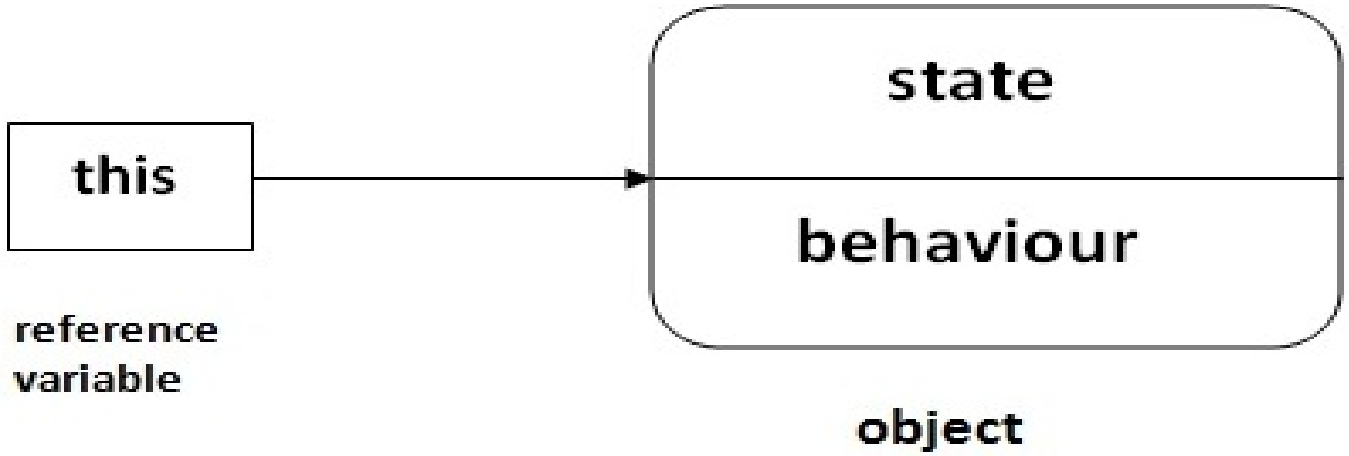
Presented By
Dr. Sudipta Sahana
Asso. Prof.
Dept. of CSE
UEM - Kolkata

Topic of Interest

- this keyword in java
- this – to invoke current class method
- this() – to invoke current class constructor
- this – to pass as an argument in the method
- this – to pass as argument in the constructor call
- this keyword can be used to return current class instance
- Methods returning objects

this keyword in java:

1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly)
3. this() can be used to invoke current class constructor.
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this can be used to return the current class instance from the method.



this – to invoke current class method:

You may invoke the method of the current class by using the this keyword. If you don't use the this keyword, compiler automatically adds this keyword while invoking the method. Let's see the example:

```
class A{
void m(){System.out.println("hello m");}
void n(){
System.out.println("hello n");
m(); //same as this.m()
//this.m();
}
}
public class TestThis4{
public static void main(String args[]){
A a=new A();
a.n();
}}
```

Output –
hello n
hello m

this() – to invoke current class constructor

The this() constructor call can be used to invoke the current class constructor. It is used to reuse the constructor. In other words, it is used for constructor chaining.

Calling no argument constructor from parameterized constructor

```
class A{
A(){System.out.println("hello a");}
A(int x){
this();
System.out.println(x);
}
}
class TestThis5{
public static void main(String args[]){
A a=new A(10);
}}
```

Output-

hello a
10

this – to pass as an argument in the method:

The this keyword can also be passed as an argument in the method. It is mainly used in the event handling. Let's see the example

```
class S2{
  void m(S2 obj){
  System.out.println("method is invoked");
  }
  void p(){
  m(this);
  }
  public static void main(String args[]){
  S2 s1 = new S2();
  s1.p();
  }
}
```

Output -

method is invoked

this – to pass as argument in the constructor call

We can pass the this keyword in the constructor also. It is useful if we have to use one object in multiple classes. Let's see the example:

```
class B{
  A4 obj;
  B(A4 obj){
    this.obj=obj;
  }
  void display(){
    System.out.println(obj.data);//using data member of A4
class
  }
}
 class A4{
  int data=10;
  A4(){
   B b=new B(this);
   b.display();
  }
  public static void main(String args[]){
   A4 a=new A4();
  }
}
```

Output -

10

this keyword can be used to return current class instance

We can return this keyword as a statement from the method. In such case, return type of the method must be the class type (non-primitive). Let's see the example:

```
class A{
A getA(){
return this;
}
void msg(){System.out.println("Hello java");}
}
public class Test1{
public static void main(String args[]){
new A().getA().msg();
/* A a = new A();
A b = a.getA();
b.msg(); */
}
}
```

Output -

Hello java

this keyword as local variable suppressor

In show() method, the local variable x suppresses the instance variable x. the console text output will be 10 and 940. “this” keyword refers to the current object, i.e. the object that is invoking the method.

```
public class A
{int x=940;
 void show(){
  int x=10;
System.out.println(x);
System.out.println(this.x);}
public static void main (String []args){
    A a = new A();
    a.show();
}
}
```

Output -

10
940

this keyword can be used to return current class instance

Proving this keyword

Let's prove that this keyword refers to the current class instance variable. In this program, we are printing the reference variable and this, output of both variables are same.

```
class A5{
void m(){
System.out.println(this);//prints same reference ID
}
public static void main(String args[]){
A5 obj=new A5();
System.out.println(obj);//prints the reference ID
obj.m();
}
}
```

Output

```
A5@6d06d69c
A5@6d06d69c
```

Methods returning objects

In java, a method can return any type of data, including objects. For example, in the following program, the incrByTen( ) method returns an object in which the value of a (an integer variable) is ten greater than it is in the invoking object.

```
class ObjectReturnDemo{
    int a;
     ObjectReturnDemo(int i)
    {
        a = i;
    }
     // This method returns an object
    ObjectReturnDemo incrByTen() {
        ObjectReturnDemo temp = new
ObjectReturnDemo(a+10);
        return temp;
    }}
// Driver class
public class Test
{
    public static void main(String args[])
    {
        ObjectReturnDemo ob1 = new
ObjectReturnDemo(2);
        ObjectReturnDemo ob2;
         ob2 = ob1.incrByTen();
         System.out.println("ob1.a: " + ob1.a);
        System.out.println("ob2.a: " + ob2.a);
    }}
```

Output

ob1.a: 2

ob2.a: 12

Thank You