Object Oriented Programming Structure

Uses of “this” keyword

1. “this” keyword can be used to refer current class instance variable.

```
class ThisDemo
{
    int i;
    void setValue(int i)
    {
        this.i=i;
    }
    void show()
    {
        System.out.println(i);
    }
    public static void main(String[] args)
    {
        ThisDemo td=new ThisDemo();
        td.setValue(100);
        td.show();
    }
}
```

2. “this” keyword can be used to invoke current class method (implicitly).

```
class ThisDemo
{
    void display()
    {
        System.out.println("hello");
    }
    void show()
    {
        display();
    }
    public static void main(String[] args)
    {
        ThisDemo td=new ThisDemo();
        td.show();
    }
}
```

If you don't use the this keyword, compiler automatically adds this keyword while invoking the method.

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

```
class ThisDemo
{
    ThisDemo()
    {
        System.out.println("no arg constructor");
    }
    ThisDemo(int a)
    {
        this();
        System.out.println("parametrised constructor");
    }
    public static void main(String[] args)
    {
        ThisDemo td=new ThisDemo(10);
    }
}
```

```
D:\javaprograms>java ThisDemo
parametrised constructor

D:\javaprograms>javac ThisDemo.java

D:\javaprograms>java ThisDemo
no arg constructor
parametrised constructor

D:\javaprograms>
```

```
class ThisDemo
{
    ThisDemo()
    {
        this(10);
        System.out.println("no arg constructor");
    }
    ThisDemo(int a)
    {
        System.out.println("parametrised constructor");
    }
    public static void main(String[] args)
    {
        ThisDemo td=new ThisDemo();
    }
}
```

```
D:\javaprograms>java ThisDemo
parametrised constructor

D:\javaprograms>javac ThisDemo.java

D:\javaprograms>java ThisDemo
no arg constructor
parametrised constructor

D:\javaprograms>javac ThisDemo.java

D:\javaprograms>java ThisDemo
parametrised constructor
no arg constructor

D:\javaprograms>
```

4. “this” can be used to pass as an argument in the method call.

```
class ThisDemo
{
    void m1(ThisDemo td)
    {
        System.out.println("im in m1 method");
    }
    void m2()
    {
        m1(this);
    }
    public static void main(String[] args)
    {
        ThisDemo td=new ThisDemo();
        td.m2();
    }
}
```

```
D:\javaprograms>java ThisDemo
parametrised constructor

D:\javaprograms>javac ThisDemo

D:\javaprograms>java ThisDemo
no arg constructor
parametrised constructor

D:\javaprograms>javac ThisDemo

D:\javaprograms>java ThisDemo
parametrised constructor
no arg constructor

D:\javaprograms>javac ThisDemo

D:\javaprograms>java ThisDemo
im in m1 method
```

5. “this” can be used to pass as an argument in the constructor call.

```
class Test
{
    Test(ThisDemo td)
    {
        System.out.println("test class constructor");
    }
}
class ThisDemo
{
    void m1()
    {
        Test t=new Test(this);
    }
    public static void main(String[] args)
    {
        ThisDemo t=new ThisDemo();
        t.m1();
    }
}
```

6. “this” can be used to return the current class instance from the method.

```
class ThisDemo
{
    ThisDemo m1()
    {
        return this;
    }
    public static void main(String[] args)
    {
        ThisDemo t=new ThisDemo();
        t.m1();
    }
}
```

Uses of “super” keyword

1. “super” keyword can be used to refer immediate parent class instance variable.

```
class A
{
    int i=10;
}
class B extends A
{
    int i=20;
    void show(int i)
    {
        System.out.println(i);          //30
        System.out.println(this.i);     //20
        System.out.println(super.i);    //10
    }
    public static void main(String[] args)
    {
        B ob=new B();
        ob.show(30);
    }
}
```

2. “super” keyword can be used to invoke immediate parent class method.

```
class A
{
    void m1()
    {
        System.out.println("i m in class A");
    }
}
class B extends A
{
    void m1()
    {
        System.out.println("i m in class B");
    }
    void show()
    {
        m1();
        super.m1();
    }
    public static void main(String[] args)
    {
        B ob=new B();
        ob.show();
    }
```

```
D:\javaprograms>java B
i m in class B
i m in class A
```

3. “super( )” can be used to invoke immediate parent class constructor.

```
class A
{
    A()
    {
        System.out.println("i m in class A");
    }
}
class B extends A
{
    B()
    {
        System.out.println("i m in class B");
    }
    public static void main(String[] args)
    {
        B ob=new B();
    }
}
```

```
D:\javaprograms>java B
i m in class A
i m in class B
```

Implicitly super( ) invokes constructor of class A

```
class A
{
    A()
    {
        System.out.println("i m in class A");
    }
}
class B extends A
{
    B()
    {
        super();
        System.out.println("i m in class B");
    }
    public static void main(String[] args)
    {
        B ob=new B();
    }
}
```

```
D:\javaprograms>java B
i m in class A
i m in class B
```

Using super( ) explicitly does the same thing.