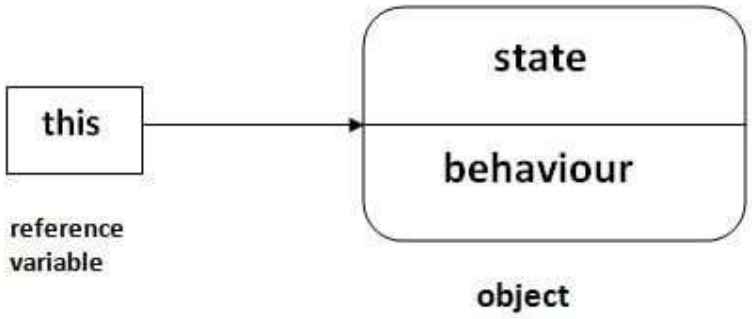
this keyword in java

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



Usage of java this keyword

Here is given the 6 usage of java this keyword.

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.

1) this: to refer current class instance variable

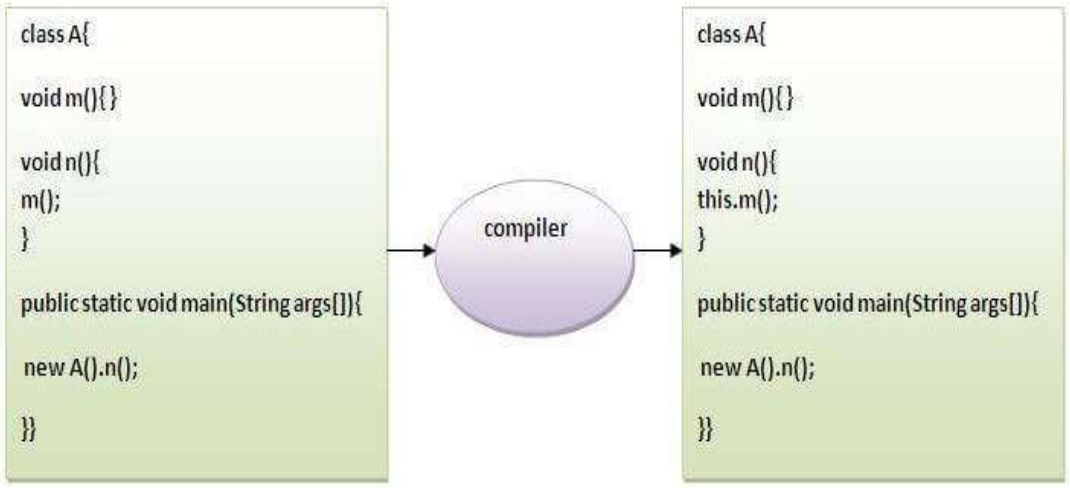
The this keyword can be used to refer current class instance variable. If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.

```
class Student
{
        introllno;
        String name;
        float fee;
        Student(introllno,String name,float fee)
        {
                this.rollno=rollno;
                this.name=name;
                this.fee=fee;
        }
        void display()
        {
```

```
                System.out.println(rollno+" "+name+" "+fee);
        }
}
classTestThis
{
        public static void main(String args[])
        {
                Student s1=new Student(111,"ankit",5000);
                Student s2=new Student(112,"sumit",6000);
                s1.display();
                s2.display();
        }
}
```

2) 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();
        }
}
class TestThis2
{
        public static void main(String args[])
        {
                A a=new A();
                a.n();
        }
}
```

3) 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 default constructor from parameterized constructor:

```
class A
{
        A()
        {
                System.out.println("hello a");
        }
        A(int x)
        {
                this();
                System.out.println(x);
        }
}
class TestThis3
{
        public static void main(String args[])
        {
                A a=new A(10);
        }
}
```

4) 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 TestThis4
{
	void m(TestThis4 obj)
	{
		System.out.println("method is invoked");
	}
	void p()
	{
		m(this);
	}
	public static void main(String args[])
	{
		TestThis4 s1 = new TestThis4();
		s1.p();
	}
}
```

5) 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
{
	TestThis5 obj;
	B(TestThis5 obj)
	{
		this.obj=obj;
	}
	void display()
	{
		System.out.println(obj.data);//using data member of A4 class
	}
}
class TestThis5
{
	int data=10;
	TestThis5()
	{
		B b=new B(this);
		b.display();
	}
	public static void main(String args[])
	{
		TestThis5 a=new TestThis5();
	}
}
```

6) this keyword can be used to return current class instance

We can return this keyword as an 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 this");
        }
}
class TestThis6
{
        public static void main(String args[])
        {
                new A().getA().msg();
        }
}
```