1) this keyword is used to refer current class instance variable

```
class Demo
{
	int i;//instance variable
	void setValue(int i)//local variable
	{
		this.i=i(15);//15
	}
	void show()
	{
		System.out.println(i);//instance	//default=0
	}
}
class DemoWthThis
{
		public static void main(String[] args)
		{
		Demo d=new Demo();
		d.setValue(15);
		d.show();
		}
}
```

2) this keyword can be used to invoke current class method(Implicitly).

```
class ThisDemo2
{
	void display()
	{
		System.out.println("Yash");
	}
	void show()
	{
		display();//this.display()
	}
	public static void main(String[] args)
	{
		ThisDemo2 t=new ThisDemo2();
		t.show();
	}

	//if you don't use this keyword,compiler automatically adds this keyword while invoking the method
}
```

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

```
class ThisDemo3
{
```

```
    ThisDemo3()
    {       //this(10);
            System.out.println("no argument constructor");
    }
    ThisDemo3(int a)
    {
            this();
            System.out.println("Parametrised constructor");
    }
    public static void main(String[] args)
    {
            ThisDemo3 t=new ThisDemo3();
    }
}
```

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

```
class ThisDemo4
{
    void y1(ThisDemo4 t)
    {
            System.out.println("Y1 method");
    }
    void y2()
    {
            y1(this);
```

```
    }

    public static void main(String[] args)
    {
        ThisDemo4 t=new ThisDemo4();
        t.y2();
    }
}
```

5. this can be used to pass as an argument in the Constructor call.

```
class t1
{
    t1(ThisDemo5 td)//constructor of t1
    {
        System.out.println("t1 class Constructor");
    }
}
class ThisDemo5
{
    void y1()
    {
        t1 t=new t1(this);//t1 class constructor as an argument(this)
    }
    public static void main(String[] args)
```

```
    {
        ThisDemo5 t=new ThisDemo5();
        t.y1();

    }
}
```

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

```
class ThisDemo6
{
    ThisDemo6 m1()
    {
        return this;
    }
    public static void main(String[] args)
    {
        ThisDemo6 t=new ThisDemo6();
        t.m1();
    }
}
```

HIERARCHICAL INHERITANCE

```
class A
{
    void displayA()
    {
```

```
            System.out.println("In A Class");
        }

}
class B extends A
{
        void displayB()
        {
            System.out.println("In B class");
        }
}
class C extends A
{
        void displayC()
        {
                System.out.println("In C class");
        }
        public static void main(String[] args)
        {
            A ob1=new A();
            ob1.displayA();
            //ob1.displayC();
            B ob2=new B();
            ob2.displayA();
            ob2.displayB();
            C ob3=new C();
```

```
            ob3.displayA();
            //ob3.displayB();
            ob3.displayC();
        }
}
```

ABSTRACTION EXAMPLE

```
abstract class Vehicle
{
        abstract void start();//50%
        void show()
        {
            System.out.println("In Abstract class");
        }

}
class Car extends Vehicle
{
        void start()
        {
            System.out.println("Starts by Key/Button");
        }

}
class Bike extends Car
{
```

```
        /*void start()
        {
                System.out.println("Starts with kick");
        }*/
        public static void main(String[] args)
        {
                //Vehicle v=new Vehicle();
                //Car c=new Car();
                //c.start();
                Bike b=new Bike();
                b.start();
                b.show();

        }
}

INTERFACE DEMO

interface A1
{
        public abstract void show();
        public static final int a=20;
}

class InterfaceDemo implements A1
```

```
{
        public void show()
        {
                System.out.println("In Demo Class");
        }
        public static void main(String[] args)
        {
                InterfaceDemo d=new InterfaceDemo();
                d.show();
        }
}
```

MULTIPLE INHERITANCE DEMO

```
interface A
{
        void show();
}
interface B
{
        void display();
}
class MultiDemo implements A,B
{
        public void show()
```

```
    {
        System.out.println("In Show method");
    }
    public void display()
    {
        System.out.println("In Display Method");
    }
    public static void main(String[] args)
    {
        MultiDemo md=new MultiDemo();
        md.show();
        md.display();

    }
}
```

OVERLOADING MAIN METHOD

```
class OverloadMain
{
    public static void main(String[] args)
    {
            System.out.println("Yash");
            OverloadMain o=new OverloadMain();
            o.main(10);
```

```
    }
    public static void main(int a)
    {
        System.out.println("Technologies");
    }
}
```

TYPE PROMOTION IN JAVA

```
class TypePro
{
    void show(Object a)
    {
        System.out.println("Object method");
    }
    void show(String b)
    {
        System.out.println("String Method");
    }
    public static void main(String[] args)
    {
        TypePro t=new TypePro();
        t.show("Jaynam");
    }
}
```

```
class TypeOver3
{
	void display(StringBuffer a)
	{
		System.out.println("StringBuffer Method");
	}
	void display(String a)
	{
		System.out.println("String method");
	}
	public static void main(String[] args)
	{
		TypeOver3 to=new TypeOver3();
		to.display("jaynam");
		to.display(new StringBuffer("yash"));

	}
}
	ENCAPSULATION
class Employee
{
	private int emp_id;//data hiding
	public void setEmpId(int emp_id)
	{
		this.emp_id=emp_id;
```

```
	}
	public int getEmpId()
	{
		return emp_id;
	}
}
class Organization
{
	public static void main(String[] args)
	{
		Employee e=new Employee();
		e.setEmpId(1000);
		System.out.println(e.getEmpId());
	}
}
```

SUPER CLASS DEMO

```
class A
{
int a=100;
}
class SuperDemo extends A
{
	int a=200;//instance variable
	void display(int a)//local variable
	{
```

```
            System.out.println(a);//300
            System.out.println(this.a);//200
            System.out.println(super.a);//100
        }
        public static void main(String[] args)
        {
            SuperDemo sd=new SuperDemo();
            sd.display(300);
        }
}
```

2) class A

```
{
        void display()
        {
            System.out.println("I am in class A-Display");
        }
}
class SuperDemo2 extends A
{
        void display()
        {
            System.out.println("I am in SuperDemo2-Display");
        }
        void show()
```

```
    {
        display();
        super.display();
    }
    public static void main(String[] args)
    {
        SuperDemo2 sd=new SuperDemo2();
        sd.show();
    }
}
```