"this" keyword

⇒ "this" keyword is the "reference variable".
⇒ this keyword refers to the current class object.
⇒ Uses :-
   1. Used to refer the current class instance variable
   2. Used to refer the current class method
   3. Used to refer the current class constructor
   4. Used as an argument in the method or constructor
   5. Used to return the current class reference

```
class MainApp
{
    MainApp ma = new MainApp();  // reference variable
}
```

⇒ 1. Used to refer the current class instance variable

```
public class MainApp1
{

    int no = 10;

    void m1(int no)
    {
        System.out.println("No. 1 : "+no); //20
        System.out.println("No. 2 : "+this.no);  //10
    }

    public static void main(String[] args)
    {
        MainApp1 obj = new MainApp1();
        obj.m1(20);
    }
}
```

```
public class MainApp2
{
    int no;  //100

    MainApp2(int no)
    {
        this.no = no;
    }

    public static void main(String[] args)
    {
        MainApp2 obj = new MainApp2(100);
        System.out.println(obj.no);
    }
}
```

⇒ 2. Used to refer the current class object

```
public class MainApp3
{
    void m1()
    {
        System.out.println("m1 method");
        m1();  // this statement will be converted into this.m1(); by compiler
    }

    void m1()
    {
        System.out.println("m2 method");
    }

    public static void main(String[] args)
    {
        MainApp3 obj = new MainApp3();
        obj.m1();
    }
}
```


⇒ 3. Used to refer the current class constructor

```
public class MainApp4
{
    MainApp4()
    {
        this(10);

        //this(10);   -- this keyword should be the first statement in the constructor.

        System.out.println("no-argument constructor");

        //this(10);   -- this keyword should be the first statement in the constructor.
    }

    MainApp4(int no)
    {
        this();

        System.out.println("parameterized constructor : "+no);
    }

    public static void main(String[] args)
    {
        new MainApp4(10);
    }
}
```

⇒ 4. Used as an argument in the method or constructor

```
public class MainApp5
{
    void m1()
    {
        System.out.println("m1 method : ");

//      MainApp5 obj = new MainApp5();
//      m2(obj);

//      m2(new MainApp5());

        m2(this);
    }

    void m2(MainApp5 obj)
    {
        System.out.println("m2 method : "+obj);
    }

    public static void main(String[] args)
    {
        MainApp5 obj = new MainApp5();
        obj.m1();
    }
}
```

⇒ 5. Used to return the current class reference

```
public class MainApp6
{

    MainApp6 m1()
    {
//        return new MainApp6();
        return this;
    }

    public static void main(String[] args)
    {
        MainApp6 obj = new MainApp6();
        System.out.println(obj.m1());
    }
}
```

"super" Keyword

⇒ "super" keyword is the reference variable
⇒ It refers to the parent class object
⇒ Uses :
   1. Used to refer the parent class instance variable
   2. Used to refer the parent class method
   3. Used to refer the parent class constructor

⇒ 1. Used to refer the parent class instance variable

```
class Test1
{
    int no = 111;
}
public class SuperKeyword1 extends Test1
{
    int no = 10;

    void m1(int no)
    {
        System.out.println("No. 1 : "+no);
        System.out.println("No. 2 : "+this.no);
        System.out.println("No. 3 : "+super.no);
    }
    public static void main(String[] args)
    {
        SuperKeyword1 obj = new SuperKeyword1();
        obj.m1(100);
    }
}
```

⇒ 2. Used to refer the parent class method

```
class Test2
{
    void m1()
    {
        System.out.println("Test2 class m1 method");
    }
}
public class SuperKeyword2 extends Test2
{
    void m1()
    {
        m1();  //super.m1();
    }

    public static void main(String[] args)
    {
        SuperKeyword2 obj = new SuperKeyword2();
        obj.m1();
    }
}
```

⇒ 3. Used to refer the parent class constructor

```
class Test3
{
    Test3()
    {
        System.out.println("Test3 constructor");
    }
}
public class SuperKeyword3 extends Test3
{
    SuperKeyword3()
    {
        super();
    }

    public static void main(String[] args)
    {
        new SuperKeyword3();
    }
}
```

"final" Keyword

⇒ "final" keyword is used for restrictions
⇒ If we use final keyword with :
   -> variable, then its value cannot be changed.
   -> method, then that method cannot override
   -> class, then we cannot inherit that class

1. final variable value cannot be changed.

```
public class FinalKeyword1
{
    public static void main(String[] args)
    {
        final int no = 10;

        //no = 20;      -- final variable value cannot be changed

        System.out.println(no);
    }
}
```

2. final method cannot override

```
class A1
{
    final void m1()
    {
        System.out.println("m1 method in class A1");
    }
}
public class FinalKeyword2 extends A1
{
//    if we override the m1() method then it will provide an error
//    @Override
//    void m1()
//    {
//        System.out.println("m1 method in class FinalKeyword2");
//    }
}
```

3. final class cannot be inherited

```
final class A3
{

}
public class FinalKeyword3 //extends A3  error
{

}
```

Interview Questions

⇒ this Keyword
1. What is the purpose of the this keyword in Java?
2. Can we call another constructor of the same class using this()?
3. How does this help resolve variable shadowing?
4. Can we use this inside a static method? Why or why not?
5. How can this be used to return the current class instance?
6. Can this be passed as an argument to a method or constructor? If yes, how? (extra)
7. What happens if we use this in a constructor? (extra)

⇒ super Keyword
8. What is the super keyword in Java, and why is it used?
9. Can we call a superclass constructor explicitly? How?
10. What happens if we do not use super() in a subclass constructor?
11. Can super be used inside a static method? Why or why not?
12. How can super be used to access parent class methods and variables?
13. Can we use both this() and super() in the same constructor? Why or why not? (extra)
14. Can we call a specific overloaded constructor of the parent class using super()? (extra)

⇒ final Keyword
15. What is the final keyword in Java, and what are its three main uses?
16. Can we declare a local variable as final? What does it mean?
17. What happens if we try to reassign a final variable?
18. Can a final method be overridden in a subclass? Why or why not?
19. Why is it useful to declare parameters as final in a method?
20. What happens if we try to extend a final class?
21. Can a final method be overloaded? Why or why not?
22. What is the difference between finally and final in Java?
23. What is the effect of declaring a class variable as both static and final?
24. Can we modify a reference variable declared as final? If yes, how?
25. What is a blank final variable? How is it initialized? (extra)
26. What is a static blank final variable? (extra)
27. Why are all variables inside an interface implicitly public static final? (extra)