This keyword

Definition:

The ***this*** keyword refers to the current object in a method or constructor.

Need for ***this*** keyword:

This keyword is used to resolve ambiguity between the instance variables and parameters. The most common use of ***this*** keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter).

Sample program without ***this*** keyword:

```
class Student
{
    int rollno;
    String name;
    float fee;

    Student(int rollno,String name,float fee) {
        rollno=rollno;
        name=name;
        fee=fee;
    }
    void display() {
        System.out.println(rollno +" "+ name +" "+fee);
    }
}
class Main
```

```
{
    public static void main(String args[]) {
        Student s1=new Student(111,"ankit",5000f);
        Student s2=new Student(112,"sumit",6000f);
        s1.display();
        s2.display();
    }
}
```

Output:

0 null 0.0
0 null 0.0

Explanation:

The Base class constructor has the same instance variable name and parameter argument name. So a confusion occurs. Thus output is the default value.

Usage of this keyword:

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

This keyword used to:

1. Refer current class instance variable:

```
class Student
```

```
{
    int rollno;
    String name;
    float fee;
    Student(int rollno,String name,float fee)
    {
        this.rollno=rollno;
        this.name=name;
        this.fee=fee;
    }
    void display()
    {
        System.out.println(rollno +" "+ name +" "+fee);
    }
}

class Main
{
    public static void main(String args[])
    {
        Student s1=new Student(111,"raj",500);
        Student s2=new Student(112,"bharat",600);
        s1.display();
        s2.display();
    }
}
```

Output:

111 raj 500.0

112 Bharat 600.0

Explanation:

The Student class constructor has the same instance variable name and parameter argument name. If ***this*** keyword is omitted the output will be 0 null 0.0 (as above), because of confusion between class attribute and parameter (having same name). To avoid the confusion ***this*** keyword is used.

2. Invoke current class method:

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

Output:

hello n
hello m
hello m

Explanation:

Current class method can be invoked with or without using ***this*** keyword.

```
this.m();
m();
```

Both these methods are the same. It invokes the current class(class Base) method(m()).

3. Invoke current class constructor:

```
class A
{
    A()
    {
      System.out.println("hello a");
    }
    A(int x)
    {
     this();//calling non-parameterized constructor
     System.out.println(x);
    }
}
class Main
{
    public static void main(String args[])
    {
      A a=new A(10);  // parameterized
    }
}
```

Output:

hello a
10

Explanation:

As the object is created with a parameter, parameterized constructor starts executing. In the first line of the parameterized constructor ***this()*** is used which calls the current class constructor. As there is no parameter inside ***this(),*** a non parameterized constructor is called and executed(which displays hello a as output in the above example).

★ Parameterized constructor can also be called by using this keyword.(Refer last example).

4. Pass as an argument in method call:

```
class A
{
    int x = 100;
    void m(A obj)
    {
        System.out.println("x:" + x);
    }
    void p()
    {
        m(this);
    }
}
class Main{
    public static void main(String args[]){
    A s1 = new A();
    s1.p();
    }
}
```

Output:

x:100

Explanation:

Parameter of the function *m()* is an object. Inside the function *p()* ***this*** is passed which passes the current object. Thus ***this*** keyword is used to pass current class object as an argument to a method. It can access the members of the class. In this example, **int x** is accessed after passing it as an object using *this* keyword.

5. Pass as an argument to constructor call:

```
class A
{
```

```
    B obj;
    A(B obj)
    {
      this.obj=obj;
    }
    void display()
    {
      System.out.println(obj.data);//using data member of B class
    }
}

class B
{
    int data=10;
    B()
    {
     A a=new A(this);//passing B class object as argument
     a.display();
    }
}

class Main
{
    public static void main(String args[])
    {
        B b=new B();//creating object for B class
    }
}
```

Output:

10

Explanation:

Inside the constructor *B()* an object is created for class A in which object is passed as an argument to the constructor of class. The constructor of class A accesses object B as it is passed as argument.

6. Return current class instance:

```
class A
{
    Base getA()
    {
        return this;//returning current object
    }
    void msg()
    {
        System.out.println("Hello java");
    }
}
class Main
{
    public static void main(String args[])
    {
        new A().getA().msg();
    }
}
```

Output:

Hello java

Explanation:

Constructor of class A returns the instance object, thus creating an object for class A is not required.

Real time usage of ***this*** keyword:

```
class Student
{
    int rollno;
    String name,course;
    float fee;

    Student(int rollno,String name,String course)
    {
        this.rollno=rollno;
        this.name=name;
        this.course=course;
    }

   Student(int rollno,String name,String course,float fee)
    {
        this(rollno,name,course);//reusing constructor
        this.fee=fee;
    }
    void display()
    {
        System.out.println(rollno +" "+ name +" "+ course +" "+fee);
    }
}
class Main
{
    public static void main(String args[])
    {
        Student s1=new Student(111,"ankit","java");
        Student s2=new Student(112,"sumit","java",6000f);
        s1.display();
        s2.display();
    }
}
```

Output:

111 ankit java null
112 sumit java 6000

Explanation:

this keyword is used to reuse constructor from constructor. It maintains the constructor chain.

Other important points:

1. It is a better approach to use meaningful names for variables. So we use the same name for instance variables and parameters in real time, and always use ***this*** keyword.
2. Call to ***this()*** must be the first statement in constructor.

```
class A
{
    A(int x)
    {
        System.out.println("a:" + x);
    }
    A()
    {
      int a = 25;
      this(a);
    }
}
class Main
{
    public static void main(String args[])
    {
      A a = new A();  // parameterized
    }
```

```
}
```

Output:

```
error: call to this must be first statement in constructor
```

Explanation:

As per rule *this()* should be called as the first statement, or else it will be an error.

To avoid this error:

this() is called as first statement

```
class A
{
    A()
    {
      this(25); // calling parameterized constructor
    }
    A(int x)
    {
        System.out.println("a:" + x);
    }
}
class Main
{
    public static void main(String args[])
    {
      A a = new A();  // non-parameterized
    }
}
```

Output:

a:25

Explanation:

When an object is created it calls a non-parameterized constructor. Which calls parameterized constructor passing 25 as value.

Practice problems: