this keyword in java - javatpoint

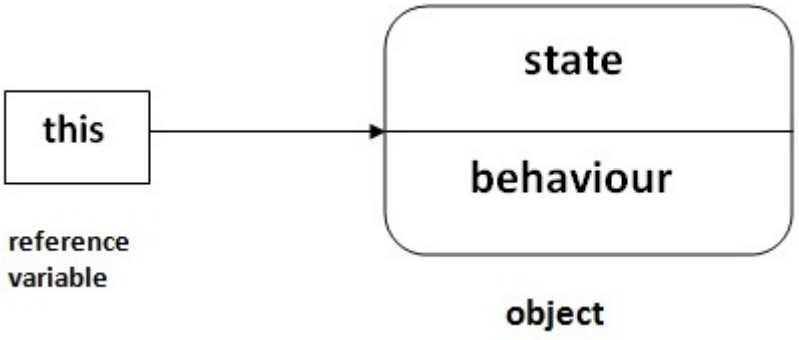
There can be a lot of usage of **java this keyword**. In java, this is a **reference variable** that refers to the current object.

Usage of java this keyword

Here is given the 6 usage of java this keyword.

1. this keyword can be used to refer current class instance variable.
2. this() can be used to invoke current class constructor.
3. this keyword can be used to invoke current class method (implicitly)
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 keyword can also be used to return the current class instance.

Suggestion: If you are beginner to java, lookup only two usage of this keyword.



1) The this keyword can be used to refer current class instance variable.

If there is ambiguity between the instance variable and parameter, this keyword resolves the problem of ambiguity.

Understanding the problem without this keyword

Let's understand the problem if we don't use this keyword by the example given below:

```
1. class Student10{
2.     int id;
3.     String name;
4.     Student10(int id,String name){
5.     id = id;
6.     name = name;
7.     }
8.     void display(){System.out.println(id+" "+name);}
9.     public static void main(String args[]){
10.    Student10 s1 = new Student10(111,"Karan");
11.    Student10 s2 = new Student10(321,"Aryan");
```

```
12.     s1.display();
13.     s2.display();
14.     }
15. }
```

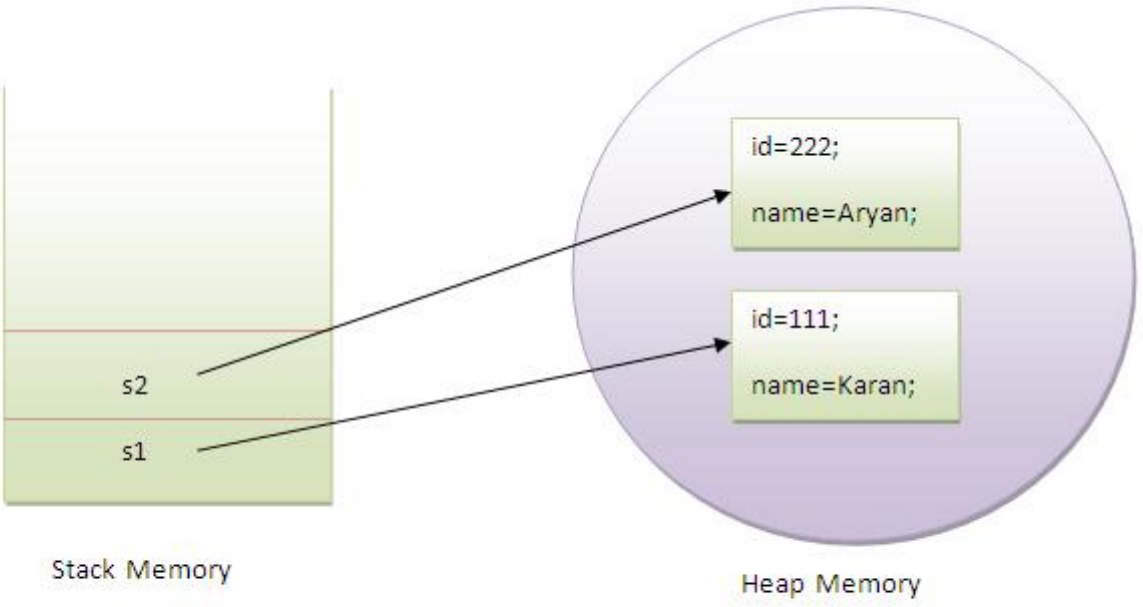
```
Output:0 null
       0 null
```

In the above example, parameter (formal arguments) and instance variables are same that is why we are using this keyword to distinguish between local variable and instance variable.

Solution of the above problem by this keyword

```
1. //example of this keyword
2. class Student11{
3.     int id;
4.     String name;
5.     Student11(int id,String name){
6.     this.id = id;
7.     this.name = name;
8.     }
9.     void display(){System.out.println(id+" "+name);}
10.    public static void main(String args[]){
11.    Student11 s1 = new Student11(111,"Karan");
12.    Student11 s2 = new Student11(222,"Aryan");
13.    s1.display();
14.    s2.display();
15. }
16. }
```

```
Output111 Karan
       222 Aryan
```



If local variables(formal arguments) and instance variables are different, there is no need to use this keyword like in the following program:

Program where this keyword is not required

```
1. class Student12{
2.     int id;
3.     String name;
4.     Student12(int i,String n){
5.     id = i;
6.     name = n;
7.     }
8.     void display(){System.out.println(id+" "+name);}
9.     public static void main(String args[]){
10.    Student12 e1 = new Student12(111,"karan");
11.    Student12 e2 = new Student12(222,"Aryan");
12.    e1.display();
13.    e2.display();
14. }
15. }
```

```
Output:111 Karan
       222 Aryan
```

2) this() can be used to invoked current class constructor.

The this() constructor call can be used to invoke the current class constructor (constructor chaining). This approach is better if you have many constructors in the class and want to reuse that constructor.

```
1. //Program of this() constructor call (constructor chaining)
2. class Student13{
3.     int id;
4.     String name;
5.     Student13(){System.out.println("default constructor is invoked");}
6.     Student13(int id,String name){
7.     this ();//it is used to invoked current class constructor.
8.     this.id = id;
9.     this.name = name;
10.    }
11.    void display(){System.out.println(id+" "+name);}
12.    public static void main(String args[]){
13.    Student13 e1 = new Student13(111,"karan");
14.    Student13 e2 = new Student13(222,"Aryan");
15.    e1.display();
16.    e2.display();
17.   }
18. }
```

```
Output:
       default constructor is invoked
       default constructor is invoked
       111 Karan
       222 Aryan
```

Where to use this() constructor call?

The this() constructor call should be used to reuse the constructor in the constructor. It maintains the chain between the constructors i.e. it is used for constructor chaining. Let's see the example given below that displays the actual use of this keyword.

```
1. class Student14{
2.     int id;
3.     String name;
4.     String city;
5.     Student14(int id,String name){
6.     this.id = id;
7.     this.name = name;
8.     }
9.     Student14(int id,String name,String city){
10.    this(id,name);//now no need to initialize id and name
11.    this.city=city;
12.    }
13.    void display(){System.out.println(id+" "+name+" "+city);}
14.    public static void main(String args[]){
15.    Student14 e1 = new Student14(111,"karan");
```

```
16.     Student14 e2 = new Student14(222,"Aryan","delhi");
17.     e1.display();
18.     e2.display();
19.    }
20. }
```

```
Output:111 Karan null
       222 Aryan delhi
```

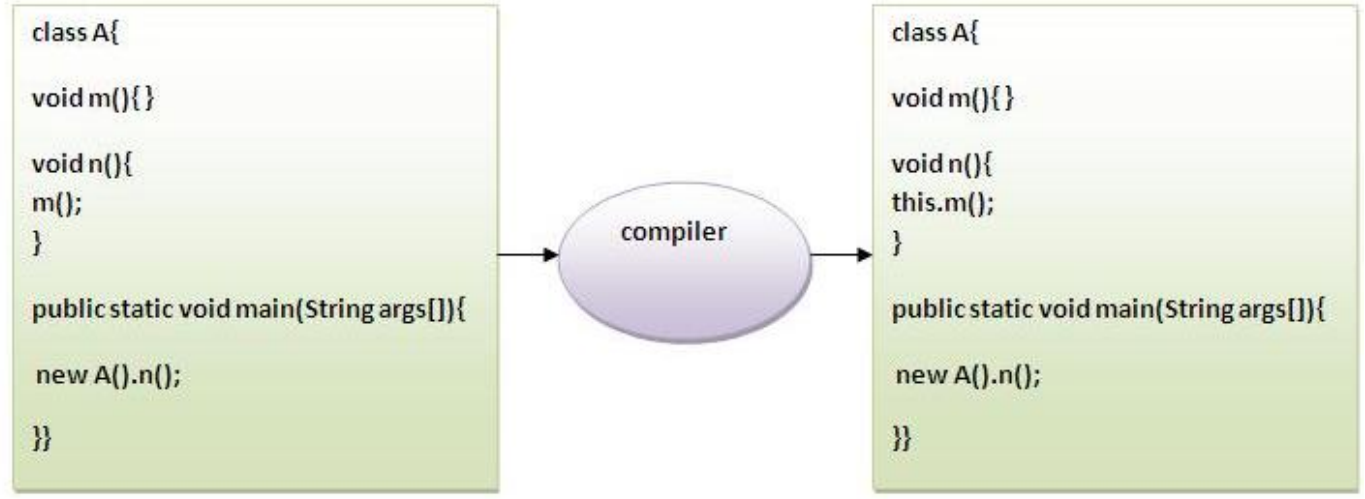
Rule: Call to this() must be the first statement in constructor.

```
1. class Student15{
2.     int id;
3.     String name;
4.     Student15(){System.out.println("default constructor is invoked");}
5.     Student15(int id,String name){
6.     id = id;
7.     name = name;
8.     this ();//must be the first statement
9.     }
10.    void display(){System.out.println(id+" "+name);}
11.    public static void main(String args[]){
12.    Student15 e1 = new Student15(111,"karan");
13.    Student15 e2 = new Student15(222,"Aryan");
14.    e1.display();
15.    e2.display();
16.   }
17. }
```

```
Output:Compile Time Error
```

3)The this keyword can be used to invoke current class method (implicitly).

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




```
1. class S{
2.   void m(){
3.   System.out.println("method is invoked");
4.   }
5.   void n(){
6.   this.m();//no need because compiler does it for you.
7.   }
8.   void p(){
9.   n();//complier will add this to invoke n() method as this.n()
10.  }
11.  public static void main(String args[]){
12.  S s1 = new S();
13.  s1.p();
14.  }
15. }
```

```
Output:method is invoked
```

4) this keyword can be passed 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:

```
1. class S2{
2.   void m(S2 obj){
3.   System.out.println("method is invoked");
4.   }
5.   void p(){
6.   m(this);
7.   }
8.   public static void main(String args[]){
9.   S2 s1 = new S2();
10.  s1.p();
11.  }
```

```
12. }
```

```
Output:method is invoked
```

Application of this that can be passed as an argument:

In event handling (or) in a situation where we have to provide reference of a class to another one.

5) The this keyword can be passed 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:

```
1. class B{
2.   A4 obj;
3.   B(A4 obj){
4.     this.obj=obj;
5.   }
6.   void display(){
7.     System.out.println(obj.data);//using data member of A4 class
8.   }
9. }
10. class A4{
11.   int data=10;
12.   A4(){
13.    B b=new B(this);
14.    b.display();
15.   }
16.   public static void main(String args[]){
17.    A4 a=new A4();
18.   }
19. }
```

```
Output:10
```

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

We can return the 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:

Syntax of this that can be returned as a statement

```
1. return_type method_name(){
2. return this;
3. }
```

Example of this keyword that you return as a statement from the method

```
1. class A{
2. A getA(){
3. return this;
4. }
5. void msg(){System.out.println("Hello java");}
6. }
7. class Test1{
8. public static void main(String args[]){
9. new A().getA().msg();
10. }
11. }
```

```
Output:Hello java
```

Proving this keyword

Let's prove that this keyword refers to the current class instance variable. In this program, we are printing the reference variable and this, output of both variables are same.

```
1. class A5{
2. void m(){
3. System.out.println(this);//prints same reference ID
4. }
5. public static void main(String args[]){
6. A5 obj=new A5();
7. System.out.println(obj);//prints the reference ID
8. obj.m();
9. }
10. }
```