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.](https://www.javatpoint.com/this1)
2. [this can be used to invoke current class method (implicitly)](https://www.javatpoint.com/this2)
3. [this() can be used to invoke current class constructor.](https://www.javatpoint.com/this3)
4. [this can be passed as an argument in the method call.](https://www.javatpoint.com/this4)
5. [this can be passed as argument in the constructor call.](https://www.javatpoint.com/this5)
6. [this can be used to return the current class instance from the method.](https://www.javatpoint.com/this6)

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.tch

Understanding the problem without this keyword

Problem if we don't use this keyword by the example given below:

**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** TestThis1{

**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

In the above example, parameters (formal arguments) and instance variables are same. So, we are using this keyword to distinguish local variable and instance variable.

Solution of the above problem by this keyword

**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** TestThis2{

**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:**

111 ankit 5000.0

112 sumit 6000.0

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

**class** Student{

**int** rollno;

String name;

**float** fee;

Student(**int** r,String n,**float** f){

rollno=r;

name=n;

fee=f;

}

**void** display(){System.out.println(rollno+" "+name+" "+fee);}

}

**class** TestThis3{

**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:**

111 ankit 5000.0

112 sumit 6000.0

It is better approach to use meaningful names for variables. So we use same name for instance variables and parameters in real time, and always use this keyword.

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** TestThis4{

**public** **static** **void** main(String args[]){

A a=**new** A();

a.n();

}}

**Output:**

hello n

hello m

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** TestThis5{

**public** **static** **void** main(String args[]){

A a=**new** A(10);

}}

**Output:**

hello a

10

**Calling parameterized constructor from default constructor:**

**class** A{

A(){

**this**(5);

System.out.println("hello a");

}

A(**int** x){

System.out.println(x);

}

}

**class** TestThis6{

**public** **static** **void** main(String args[]){

A a=**new** A();

}}

**Output:**

5

hello a

Real usage of this() constructor call

The this() constructor call should be used to reuse the constructor from 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.

**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** TestThis7{

**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 0.0

112 sumit java 6000.0

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

**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**.fee=fee;

**this**(rollno,name,course);//C.T.Error

}

**void** display(){System.out.println(rollno+" "+name+" "+course+" "+fee);}

}

**class** TestThis8{

**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:**

Compile Time Error: Call to this must be first statement in constructor

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** S2{

**void** m(S2 obj){

  System.out.println("method is invoked");

  }

**void** p(){

  m(**this**);

  }

**public** **static** **void** main(String args[]){

  S2 s1 = **new** S2();

  s1.p();

  }

}

**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. It is used to reuse one object in many methods.

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{

  A4 obj;

  B(A4 obj){

**this**.obj=obj;

  }

**void** display(){

    System.out.println(obj.data);//using data member of A4 class

  }

}

**class** A4{

**int** data=10;

  A4(){

   B b=**new** B(**this**);

   b.display();

  }

**public** **static** **void** main(String args[]){

   A4 a=**new** A4();

  }

}

Output:10

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:

Syntax of this that can be returned as a statement

return\_type method\_name(){

**return** **this**;

}

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

**class** A{

A getA(){

**return** **this**;

}

**void** msg(){System.out.println("Hello java");}

}

**class** Test1{

**public** **static** **void** main(String args[]){

**new** A().getA().msg();

}

}

**Output:**

Hello java

Proving this keyword

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.

**class** A5{

**void** m(){

System.out.println(**this**);//prints same reference ID

}

**public** **static** **void** main(String args[]){

A5 obj=**new** A5();

System.out.println(obj);//prints the reference ID

obj.m();

}

}

**Output:**

A5@22b3ea59

A5@22b3ea59