**Java this keyword**

"**this**" keyword refers an object, to say, the **current object**.

" this " is java keyword/reference which is referring to current class instance or object so that can access the all members of current class using this "this" keyword with in the same class.

Writing this keyword is sometime optional / sometimes mandatory.

If there is no confusion between local variables and instance variables then writing this keyword is optional. In this case compiler will write this keyword and refer to instance variable.

But if there is a confusion between local variables and instance variables then writing this keyword is mandatory. In this case compiler will not write this keyword and refer to local variable. But if we want to refer to instance variable we have to write this keyword explicitly.

**Example**

***//wap to demo on this keyword***

*class Demo{*

*int a=10;*

*int b=20;*

*static int x=99;*

*void show(){*

*int a=30;*

*int c=40;*

*System.out.println(this.a);//this keyword is mandatory*

*System.out.println(a);//local variable*

*System.out.println(this.b);//this keyword is optional*

*System.out.println(c);//local variable*

*System.out.println(this.x);//this keyword is optional*

*this.display();//this keyword is optional*

*this.printing();//this keyword is optional*

*}*

*void display(){*

*System.out.println("this is display() method");*

*}*

*static void printing(){*

*System.out.println("this is printing method");*

*}*

*}*

*class ThisDemo1{*

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

*Demo d = new Demo();*

*d.show();*

*}*

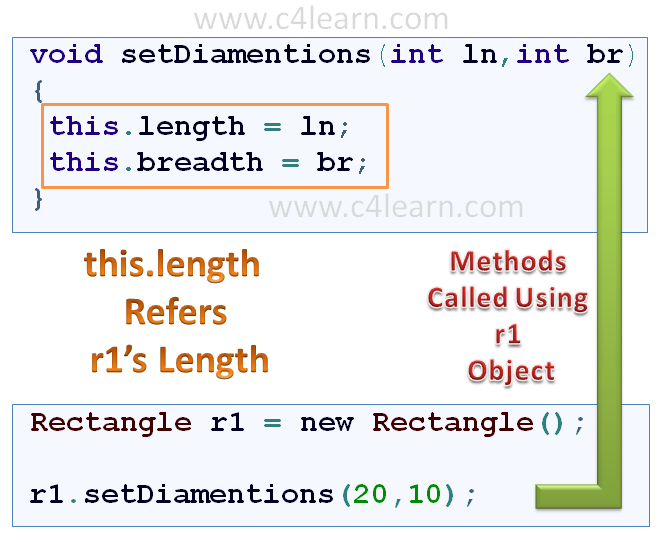
*}*

**this keyword : Refer Current Object in Java Programming**  
In java, this is a **reference variable** that refers to the current object.

1. **this** is keyword in Java.
2. We can use **this** keyword in[any method](http://www.c4learn.com/introducing-methods-in-java-class-class-concept-in-java.html) or constructor.
3. **this** keyword used to **refer current object**.
4. Use **this keyword** from any method or constructor to refer to the **current object that calls a method or invokes constructor** .

### *Syntax : this Keyword*

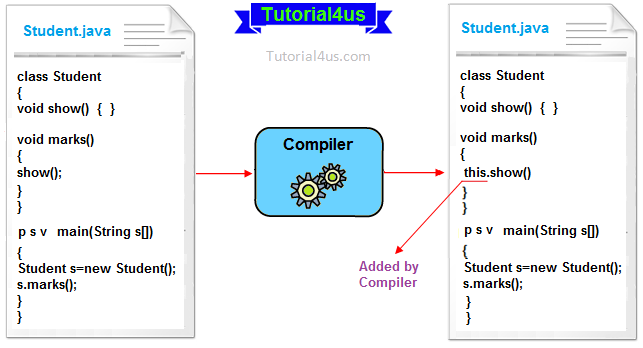
***this****.field*



## java this keyword

### this keyword used to invoke current class method (implicitly)

By using this keyword you can invoke the method of the current class. If you do not use the this keyword, compiler automatically adds this keyword at time of invoking of the method.



## Example of this keyword

**class** Student{

**void** show(){

System.**out**.println("You got A+");

}

**void** marks(){

**this**.show(); //no need to use this here because compiler does it.

}

**void** display(){

show(); //compiler act marks() as this.marks()

}

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

Student s = **new** Student();

s.display();

}

}

## Syntax

You got A+

**Live Example : this Keyword**

*class Rectangle {*

***int*** *length;*

***int*** *breadth;*

***void*** *setDiamentions(****int*** *ln,****int*** *br){*

***this****.length = ln;*

***this****.breadth = br;*

*}*

*}*

*class RectangleDemo {*

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

*Rectangle r1 = new Rectangle();*

*r1.setDiamentions(20,10);*

*System.out.println("Length of Rectangle : " + r1.length);*

*System.out.println("Breadth of Rectangle : " + r1.breadth);*

*}*

*}*

Output :

*java RectangleDemo*

*Length of Rectangle : 20*

*Breadth of Rectangle : 10*

**this Keyword is used to hide Instance Variable :**

***void*** *setDiamentions(****int*** *length,****int*** *breadth) {*

***this****.length = length;*

***this****.breadth = breadth;*

*}*

* length,breadth are the **parameters that are passed to the method**.
* Same names are given to the **instance variables of an object**.
* In order to hide instance variable we can use this keyword. above syntax will clearly make **difference between instance variable and parameter**.

## Example:

*class Employee{*

*int empno;*

*String ename;*

*double salary;*

*Employee(int empno,String ename,double salary){*

*this.empno=empno;*

*this.ename=ename;*

*this.salary=salary;*

*this.display();*

*}*

*void display(){*

*System.out.println("EMPNO:"+empno);*

*System.out.println("ENAME:"+ename);*

*System.out.println("SALARY:"+salary);*

*}*

*}*

*class Constructor2{*

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

*Employee e = new Employee(1,"sehwagh",120000.0);*

*}*

*}*

## Observe the following program and pay attention to **this.salary**.

*public class Officer3 {*

*int salary;*

*public void display(int salary){*

*this.salary = salary;*

*}*

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

*Officer3 o1 = new Officer3();*

*o1.display(5000);*

*System.out.println(o1.salary); // 5000*

*}*

*}*



In the **display()** method, local variable and the instance variable are same, **salary**. That is, local variable clashes with instance variable. If "**this**" is removed in **display()**method, **o1.salary** in the main() method prints 0. "**this**" refers the object **o1** as with **o1** the **display()** method is called. Now we can make one rule, **if a method is called with an object, the instance variables inside the method are linked with the object implicitly**.

Now, **this.salary** becomes **o1.salary** internally and one memory location is created. If "**this**" is removed, both salaries are treated as local variables and thereby the compiler does not create any memory location. Use always "**this**" keyword to differentiate a local from instance variable when they clash. C++ people, call "this keyword" as "this pointer".

**Note:**

With this keyword we can access both instance and static members

We can use this keyword inside the constructor,inside the instance methods of the class but we cannot use this keyword inside the static methods of the class

## *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**.**

### 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: |

***class*** *Student10{*

***int*** *id;*

*String name;*

*Student10(****int*** *id,String name){*

*id = id;*

*name = name;*

*}*

***void*** *display(){*

*System.out.println(id+" "+name);*

*}*

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

*Student10 s1 =* ***new*** *Student10(111,"Karan");*

*Student10 s2 =* ***new*** *Student10(321,"Aryan");*

*s1.display();*

*s2.display();*

*}*

*}*

**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

***//Example of this keyword***

***class*** *Student11{*

***int*** *id;*

*String name;*

*Student11(****int*** *id,String name){*

***this****.id = id;*

***this****.name = name;*

*}*

***void*** *display(){*

*System.out.println(id+" "+name);*

*}*

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

*Student11 s1 =* ***new*** *Student11(111,"Karan");*

*Student11 s2 =* ***new*** *Student11(222,"Aryan");*

*s1.display();*

*s2.display();*

*}*

*}*

**Output**:  
 111 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

**class** Student12{

**int** id;

String name;

Student12(**int** i,String n){

id = i;

name = n;

}

**void** display(){

System.*out*.println(id+" "+name);

}

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

Student12 e1 = **new** Student12(111,"karan");

Student12 e2 = **new** Student12(222,"Aryan");

e1.display();

e2.display();

}

}

**Output:**  
*111 Karan*

*222 Aryan*

**Example:**

***class*** *Student12{*

***int*** *id;*

*String name;*

*Student12(****int*** *i,String n){*

*id = id;*

*name = n;*

*}*

***void*** *display(){*

*System.out.println(id+" "+name);*

*}*

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

*Student12 e1 =* ***new*** *Student12(111,"karan");*

*Student12 e2 =* ***new*** *Student12(222,"Aryan");*

*e1.display();*

*e2.display();*

*}*

*}*

**Output:**

0 karan

0 Aryan

### *2) this() can be used to invoked current class constructor:*

this() which can be used to call one constructor within the another constructor without creation of objects multiple time for the same class.

## Syntax

**this**(); // call no parametrized or default constructor

**this**(value1,value2,.....) //call parametrize 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.

//**Program of this() constructor call (constructor chaining)**

***class*** *Student13{*

***int*** *id;*

*String name;*

*Student13(){*

*System.out.println("default constructor is invoked");*

*}*

*Student13(****int*** *id,String name){*

***this*** *();//it is used to invoked current class constructor.*

***this****.id = id;*

***this****.name = name;*

*}*

***void*** *display(){*

*System.out.println(id+" "+name);*

*}*

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

*Student13 e1 =* ***new*** *Student13(111,"karan");*

*Student13 e2 =* ***new*** *Student13(222,"Aryan");*

*e1.display();*

*e2.display();*

*}*

*}*

**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.**

***class*** *Student14{*

***int*** *id;*

*String name;*

*String city;*

*Student14(****int*** *id,String name){*

***this****.id = id;*

***this****.name = name;*

*}*

*Student14(****int*** *id,String name,String city){*

***this****(id,name);//now no need to initialize id and name*

***this****.city=city;*

*}*

***void*** *display(){*

*System.out.println(id+" "+name+" "+city);*

*}*

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

*Student14 e1 =* ***new*** *Student14(111,"karan");*

*Student14 e2 =* ***new*** *Student14(222,"Aryan","delhi");*

*e1.display();*

*e2.display();*

*}*

*}*

**Output**:

111 Karan null

222 Aryan delhi

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

***class*** *Student15{*

***int*** *id;*

*String name;*

*Student15(){*

*System.out.println("default constructor is invoked");*

*}*

*Student15(****int*** *id,String name){*

*id = id;*

*name = name;*

***this ();//must be the first statement***

*}*

***void*** *display(){*

*System.out.println(id+" "+name);*

*}*

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

*Student15 e1 =* ***new*** *Student15(111,"karan");*

*Student15 e2 =* ***new*** *Student15(222,"Aryan");*

*e1.display();*

*e2.display();*

*}*

*}*

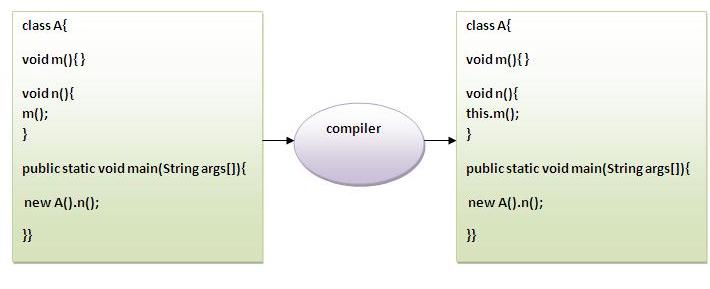
**Output**:

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**



***class*** *S{*

***void*** *m(){*

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

*}*

***void*** *n(){*

***this****.m();//no need because compiler does it for you.*

*}*

***void*** *p(){*

*n();//complier will add this to invoke n() method as this.n()*

*}*

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

*S s1 =* ***new*** *S();*

*s1.p();*

*}*

*}*

**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:**

***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. |

### 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:**

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

|  |
| --- |
| We can return this keyword as a 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(){*  *System.out.println("getA()");*  ***return******this****;*  *}*  ***void*** *msg(){*  *System.out.println("msg()");*  *}*  *}*  ***class*** *Test1{*  ***public******static******void*** *main(String args[]){*  *System.out.println("Before new A().getA().msg()");*  ***new*** *A().getA().msg();*  *System.out.println("After new A().getA().msg()");*  *}*  *}*  **Output**:  Before new A().getA().msg()  getA()  msg()  After new A().getA().msg() 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.  **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@e48e1b  A5@e48e1b  **What about local variables is memory created for them too ?**   1. Local variables memory exist as long as the method executes.   **this():**  this() is used to invoke and access the 0 parametrized constructor of the same class inside another constructor.  **this(...)**  this(...) is used to invoke and access the parameterized constructor of the same class inside another constructor.  **//program to demo on this(), this(...)**  *class Demo{*  *Demo(){*  *this(10);*  *System.out.println("this is 0 parameterized constructor");*  *}*  *Demo(int x){*  *System.out.println("this is 1 parameterized constructor");*  *}*  *Demo(int x,int y){*  *this();*  *System.out.println("this is 2 parameterized constructor");*  *}*  *}*  *class ThisDemo2{*  *public static void main(String args[]){*  *Demo d = new Demo(10,20);*  *}*  *}*  **o/p:**  *this is 1 parameterized constructor*  *this is 0 parameterized constructor*  *this is 2 parameterized constructor*  **Rules**  1. Calling this() or this(...) must be the first statements in the constructor.  2. Calling this() or this(...) must not be in side the regular method  3. We can call this() or this(...) at most 1 time  4. Calling this() or this(...) must not provide a recursive constructor invocation  **Example:** |
| |  | | --- | | *class AnyClass{*  *int i;*  *AnyClass(){*  *System.out.println("First Constructor");*  *}*  *AnyClass(int j){*  *this();    //****calling statement to First Constructor***  *System.out.println("Second Constructor");*  *}*  *void methodOne(){*  *System.out.println("From method one");*  *}*  *void methodTwo(){*  *System.out.println(this.i);****//Accessing same class field***  *this.methodOne();      //****Accessing same class method***  *}*  *}* |   this() is the calling statement to same class constructor. It must be used within constructor only. If it is used, it must be the first statement in the constructor. |

## Rules to use this()

**this()** always should be the first statement of the constructor. One constructor can call only other single constructor at a time by using **this()**.

**)**.

