This keyword

this is a reference variable that refers to the current object. It is a keyword in java language represents current class object

Usage of this keyword

- It can be used to refer current class instance variable.
- this() can be used to invoke current class constructor.
- It can be used to invoke current class method (implicitly)
- It can be passed as an argument in the method call.
- It can be passed as argument in the constructor call.
- It can also be used to return the current class instance.

Why use this keyword in java ?

The main purpose of using this keyword is to differentiate the formal parameter and data members of class, whenever the formal parameter and data members of the class are similar then jvm get ambiguity (no clarity between formal parameter and member of the class)

To differentiate between formal parameter and data member of the class, the data member of the class must be preceded by "this".

"this" keyword can be use in two ways.

- this . (this dot)
- this() (this off)

this . (this dot)

which can be used to differentiate variable of class and formal parameters of method or constructor.

"this" keyword are used for two purpose, they are

- It always points to current class object.
- Whenever the formal parameter and data member of the class are similar and JVM gets an ambiguity (no clarity between formal parameter and data members of the class).

To differentiate between formal parameter and data member of the class, the data members of the class must be preceded by "this".

Syntax

this.data member of current class.

Note: If any variable is preceded by "this" JVM treated that variable as class variable.

Example without using this keyword

```
class Employee
{
  int id;
  String name;

  Employee(int id,String name)
 {
  id = id;
  name = name;
 }
 void show(){
  System.out.println(id+" "+name);
 }
  public static void main(String args[]){
   Employee e1 = new Employee(111,"Harry");
   Employee e2 = new Employee(112,"Jacy");
   e1.show();
   e2.show();
     }
  }
```

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.
Example of this keyword in java

```
class Employee{
  int id;
  String name;
  Employee(int id,String name){
  this.id = id;
  this.name = name;
 }
    void show(){
  System.out.println(id+" "+name);
 }
  public static void main(String args[]){
   Employee e1 = new Employee(111,"Harry");
   Employee e2 = new Employee(112,"Jacy");
   e1.show();
   e2.show();
  }
}
```

Output
111 Harry
112 Jacy

Note 1: The scope of "this" keyword is within the class.
Note 2: The main purpose of using "this" keyword in real life application is to differentiate variable of class or formal parameters of methods or constructor (it is highly recommended to use the same variable name either in a class or method and constructor while working with similar objects).

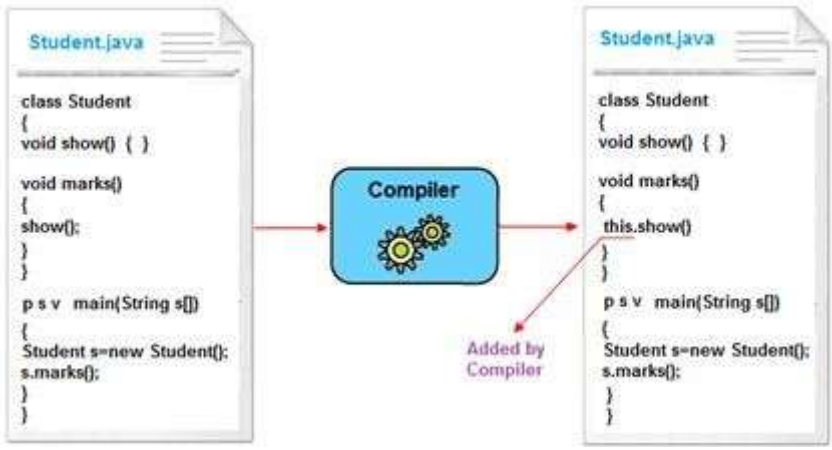
this ()

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
```

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

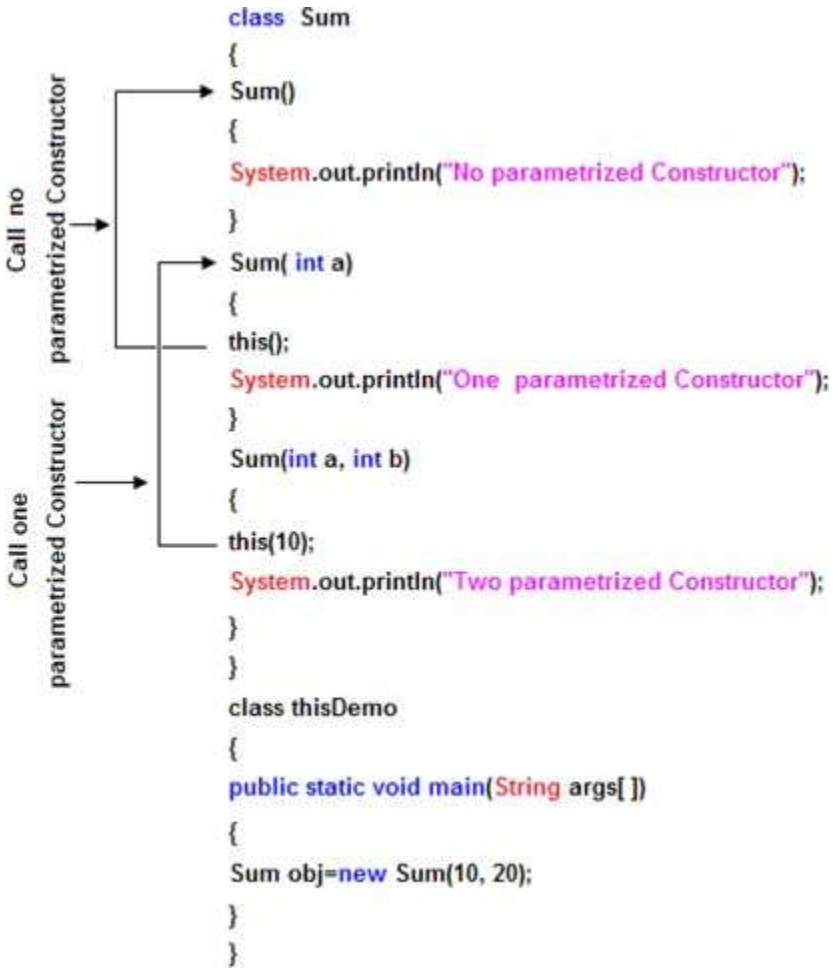
```
class Student
{
 void show()
  {
  System.out.println("You got A+");
  }
  void marks()
  {
  this.show(); //no need to use this here because compiler does it.
  }
  void display()
  {
  show(); //compiler act marks() as this.marks()
  }
  public static void main(String args[])
  {
  Student s = new Student();
  s.display();
  }
}
```

Syntax

You got A+

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().


Call no parametrized Constructor
Call one parametrized Constructor
class Sum
{
Sum()
{
System.out.println("No parametrized Constructor");
}
Sum(int a)
{
this();
System.out.println("One parametrized Constructor");
}
Sum(int a, int b)
{
this(10);
System.out.println("Two parametrized Constructor");
}
}
class thisDemo
{
public static void main(String args[ ])
{
Sum obj=new Sum(10, 20);
}
}