‘this’ reference in Java

Last Updated : 08 Jan, 2024

In Java, ‘this’ is a reference variable that refers to the current object, or can be said “this” in Java is a keyword that refers to the current object instance. It can be used to call current class methods and fields, to pass an instance of the current class as a parameter, and to differentiate between the local and instance variables. Using “this” reference can improve code readability and reduce naming conflicts.

Java this reference Example

In Java, this is a reference variable that refers to the current object on which the method or constructor is being invoked. It can be used to access instance variables and methods of the current object.

Below is the implementation of this reference:

Java

```java
// Java Program to implement
// Java this reference

// Driver Class
public class Person {

    // Fields Declared
    String name;
    int age;

    // Constructor
    Person(String name, int age)
    {
        this.name = name;
        this.age = age;
    }

    // Getter for name
    public String get_name() { return name; }

    // Setter for name
    public void change_name(String name)
    {
        this.name = name;
    }

    // Method to Print the Details of
    // the person
    public void printDetails()
    {
        System.out.println("Name: " + this.name);
        System.out.println("Age: " + this.age);
        System.out.println();
    }

    // main function
    public static void main(String[] args)
    {
        // Objects Declared
        Person first = new Person("ABC", 18);
        Person second = new Person("XYZ", 22);

        first.printDetails();
        second.printDetails();

        first.change_name("PQR");
        System.out.println("Name has been changed to: "
                           + first.get_name());
    }
}
```

Output

```
Name: ABC
Age: 18

Name: XYZ
Age: 22
```

```
Name has been changed to: PQR
```

Explanation

In the above code, we have defined a Person class with two private fields name and age. We have defined the Person class constructor to initialize these fields using this keyword. We have also defined getter and setter methods for these fields which use this keyword to refer to the current object instance.

In the printDetails() method, we have used this keyword to refer to the current object instance and print its name, age, and object reference.

In the Main class, we have created two Person objects and called the printDetails() method on each object. The output shows the name, age, and object reference of each object instance.

Methods to use 'this' in Java

Following are the ways to use the 'this' keyword in Java mentioned below:

- Using the 'this' keyword to refer to current class instance variables.
- Using this() to invoke the current class constructor
- Using 'this' keyword to return the current class instance
- Using 'this' keyword as the method parameter
- Using 'this' keyword to invoke the current class method
- Using 'this' keyword as an argument in the constructor call

1. Using 'this' keyword to refer to current class instance variables

Java

```
// Java code for using 'this' keyword to
// refer current class instance variables
class Test {
    int a;
    int b;

    // Parameterized constructor
    Test(int a, int b)
    {
        this.a = a;
        this.b = b;
    }

    void display()
    {
        // Displaying value of variables a and b
        System.out.println("a = " + a + "  b = " + b);
    }

    public static void main(String[] args)
    {
        Test object = new Test(10, 20);
        object.display();
    }
}
```

Output

```
a = 10  b = 20
```

2. Using this() to invoke current class constructor

Java

```
// Java code for using this() to
// invoke current class constructor
class Test {
    int a;
    int b;

    // Default constructor
    Test()
    {
        this(10, 20);
        System.out.println(
            "Inside  default constructor \n");
    }

    // Parameterized constructor
    Test(int a, int b)
    {
```

```java
        this.a = a;
        this.b = b;
        System.out.println(
            "Inside parameterized constructor");
    }

    public static void main(String[] args)
    {
        Test object = new Test();
    }
}
```

Output

```
Inside parameterized constructor
Inside  default constructor
```

3. Using ‘this’ keyword to return the current class instance

Java

```java
// Java code for using 'this' keyword
// to return the current class instance
class Test {
    int a;
    int b;

    // Default constructor
    Test()
    {
        a = 10;
        b = 20;
    }

    // Method that returns current class instance
    Test get() { return this; }

    // Displaying value of variables a and b
    void display()
    {
        System.out.println("a = " + a + "  b = " + b);
    }
    public static void main(String[] args)
    {
        Test object = new Test();
        object.get().display();
    }
}
```

Output

```
a = 10  b = 20
```

4. Using 'this' keyword as a method parameter

Java

```
// Java code for using 'this'
// keyword as method parameter
class Test {
    int a;
    int b;

    // Default constructor
    Test()
    {
        a = 10;
        b = 20;
    }

    // Method that receives 'this' keyword as parameter
    void display(Test obj)
    {
        System.out.println("a = " + obj.a
                           + "  b = " + obj.b);
    }

    // Method that returns current class instance
    void get() { display(this); }

    // main function
    public static void main(String[] args)
    {
        Test object = new Test();
        object.get();
    }
}
```

Output

```
a = 10  b = 20
```

5. Using ‘this’ keyword to invoke the current class method

Java

```java
// Java code for using this to invoke current
// class method
class Test {

    void display()
    {
        // calling function show()
        this.show();

        System.out.println("Inside display function");
    }

    void show()
    {
        System.out.println("Inside show function");
    }

    public static void main(String args[])
    {
        Test t1 = new Test();
        t1.display();
    }
}
```

Output

```
Inside show function
Inside display function
```

6. Using ‘this’ keyword as an argument in the constructor call

Java

```java
// Java code for using this as an argument in constructor
// call
// Class with object of Class B as its data member
class A {
    B obj;

    // Parameterized constructor with object of B
    // as a parameter
    A(B obj)
    {
        this.obj = obj;

        // calling display method of class B
        obj.display();
    }
}

class B {
    int x = 5;

    // Default Constructor that create an object of A
    // with passing this as an argument in the
    // constructor
    B() { A obj = new A(this); }

    // method to show value of x
    void display()
    {
        System.out.println("Value of x in Class B : " + x);
    }

    public static void main(String[] args)
    {
        B obj = new B();
    }
}
```

Output

```
Value of x in Class B : 5
```

Advantages of using ‘this’ reference

There are certain advantages of using ‘this’ reference in Java as mentioned below:

1. It helps to distinguish between instance variables and local variables with the same name.
2. It can be used to pass the current object as an argument to another method.
3. It can be used to return the current object from a method.
4. It can be used to invoke a constructor from another overloaded constructor in the same class.

Disadvantages of using ‘this’ reference

Although ‘this’ reference comes with many advantages there are certain disadvantages of also:

1. Overuse of this can make the code harder to read and understand.
2. Using this unnecessarily can add unnecessary overhead to the program.
3. Using this in a static context results in a compile-time error.
4. Overall, this keyword is a useful tool for working with objects in Java, but it should be used judiciously and only when necessary.

This article is contributed by **Mehak Narang** and **Amit Kumar**.

Start your **Java programming** journey today with our Java Programming Online Course, designed for both beginners and advanced learners. With self-paced lessons covering everything from **basic syntax to advanced concepts**, you’ll gain the skills needed to excel in the world of programming.

Take the **Three 90 Challenge**! Complete **90% of the course** in **90 days**, and **earn a 90% refund.** Track your progress and stay