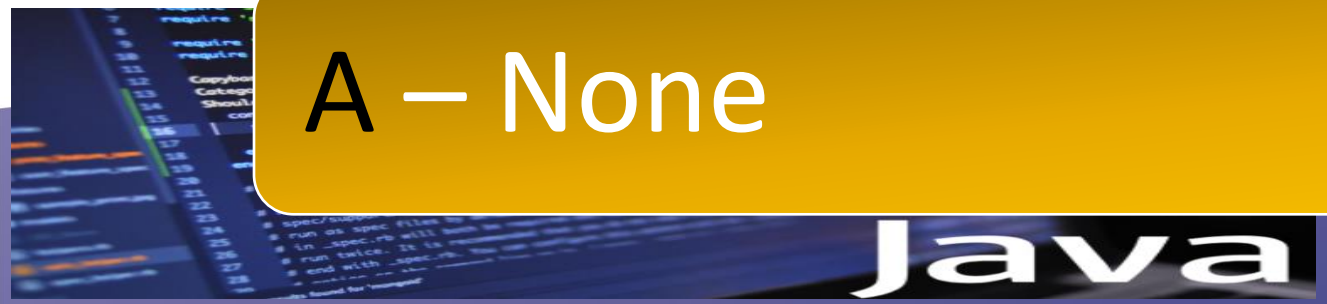
CJ Information

S – more on THIS keyword

IC – explanation + examples

HW – 2024.11.11 Mon – Java classes

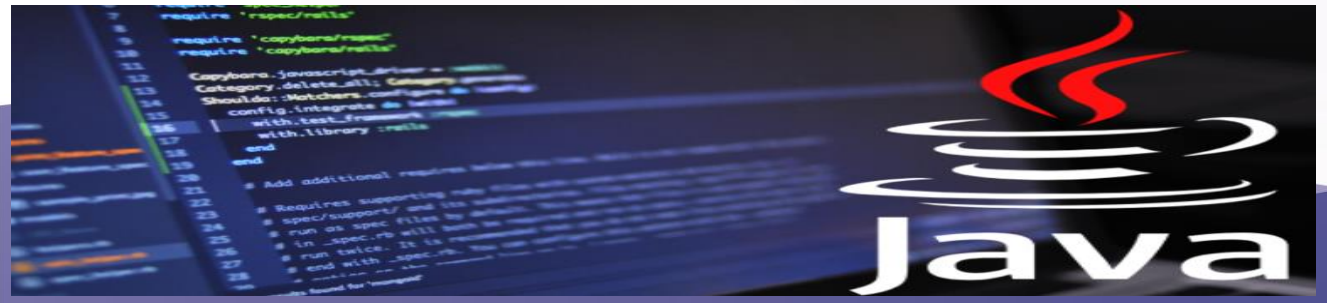
A – None



THIS keyword (…understanding it)

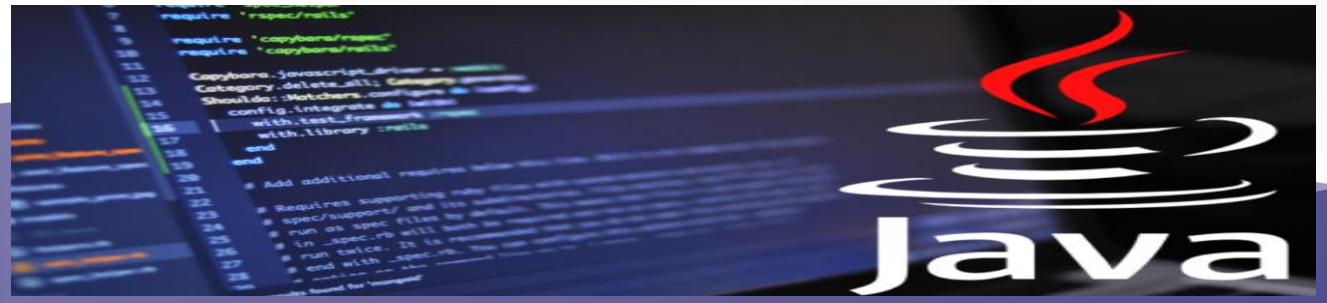
Outline :

- What is it ?
- What is it useful for?



What we learn today:

- How to use This
- Common Mistakes
- Example



THIS keyword

Usage of Java this Keyword

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

01 this can be used to refer current class instance variable.

02 this can be used to invoke current class method (implicity)

03 this() can be used to invoke current class Constructor.

04 this can be passed as an argument in the method call.

05 this can be passed as argument in the constructor call.

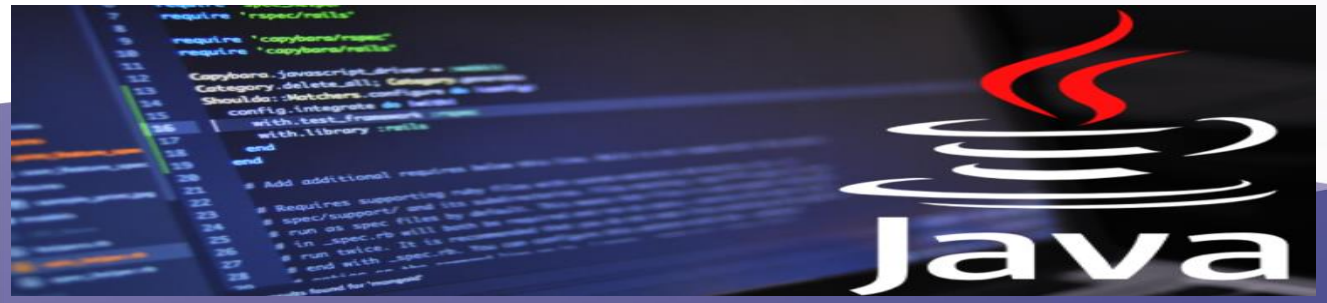
06 this can be used to return the current class instance from the method

THIS keyword

6 Usages of this keyword

Here is given the 6 usages of Java this keyword.

1. this keyword can be used to refer current class instance variable.
2. this keyword can be used to invoke current class method (implicitly)
3. this() can be used to invoke the current class constructor.
4. this keyword can be passed as an argument in the method call.
5. this keyword can be passed as an argument in the constructor call.
6. this keyword can be used to return the current class instance from the method.

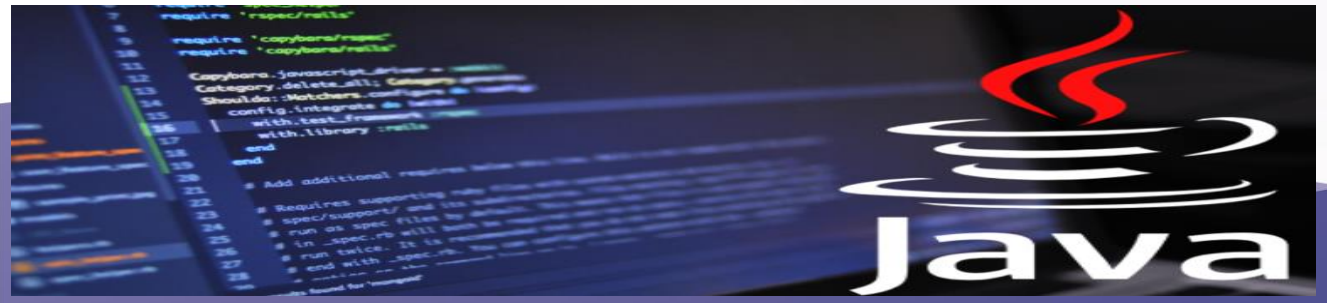


THIS keyword

1. this keyword can be used to refer to a 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.



THIS keyword

Understanding the problem without this keyword

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

```
public class User {
    private int id;
    private String firstName;
    private String lastName;
    private int age;

    public User(int id, String firstName, String lastName, int age) {
        id = id;
        firstName = firstName;
        lastName = lastName;
        age = age;
    }

    @Override
    public String toString() {
        return "User [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + ", age=" + age + "]";
    }

    public static void main(String[] args) {
        User user = new User(1, "Ramesh", "Fadatare", 28);
        System.out.println(user.toString());
    }
}
```

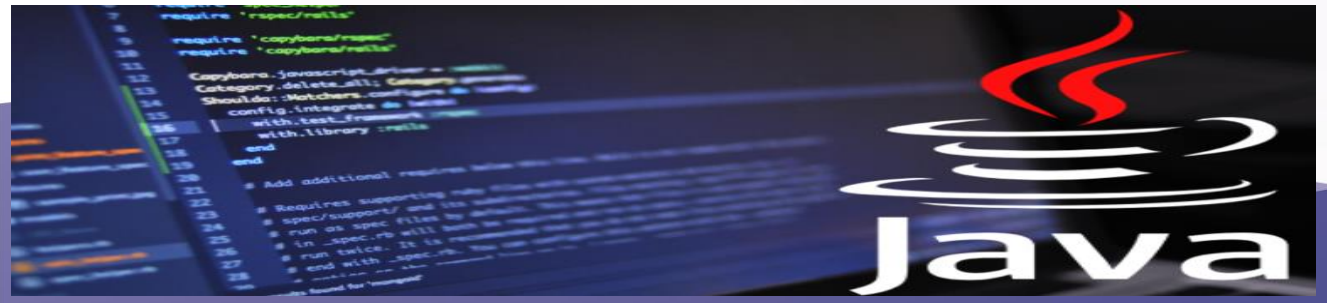
THIS keyword

Output:

```
User [id=0, firstName=null, lastName=null, age=0]
```

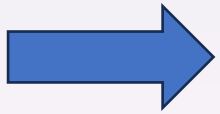
In the above example, parameters (formal arguments) and instance variables are the same hence we can see the output is wrong.

NOTE : this problem is called Instance Variable Hiding!!!!!
To solve this problem we have to use **THIS** keyword



THIS keyword

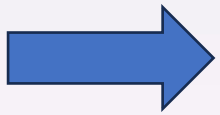

```java
public class Account
{
    int a ;
    int b ;

    public void setData (int a , int b)
    {
        a = a;
        b = b;
    }

    public void showData()
    {
        System.out.println("The value of a is : " + a);
        System.out.println("The value of b is : " + b);
    }


    public static void main(String[] args) {

        Account obj = new Account ();
        obj.setData(2,3);
        obj.showData();


    }
}
```



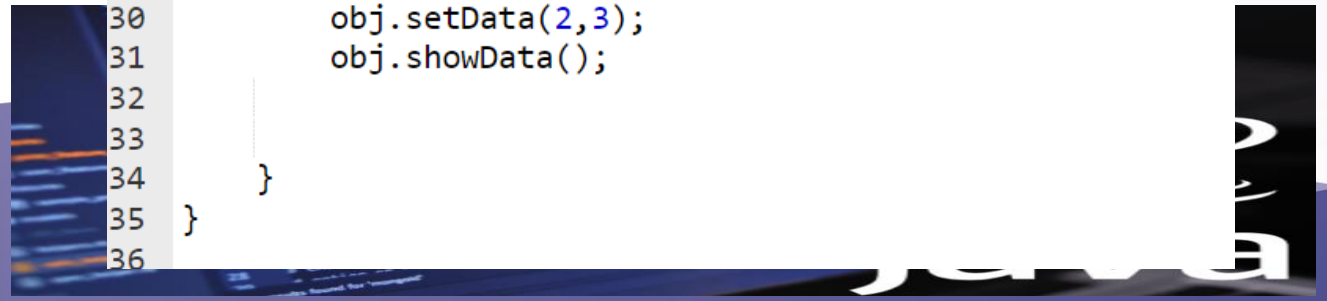
```
The value of a is : 0
The value of b is : 0
```

THIS keyword


```
 9  public class Account
10  {
11      int a ;
12      int b ;
13
14      public void setData (int a , int b)
15      {
16          this.a = a;
17          this.b = b;
18      }
19
20      public void showData()
21      {
22          System.out.println("The value of a is : " + a);
23          System.out.println("The value of b is : " + b);
24      }
25
26
27      public static void main(String[] args) {
28
29          Account obj = new Account ();
30          obj.setData(2,3);
31          obj.showData();
32
33
34      }
35  }
36
```

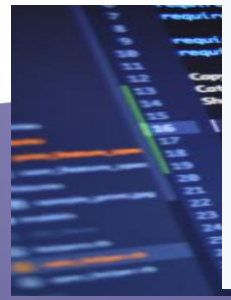


```
The value of a is : 2
The value of b is : 3
```



THIS keyword


```java
public class UserWithThisKeyword {
    private int id;
    private String firstName;
    private String lastName;
    private int age;

    public UserWithThisKeyword(int id, String firstName, String lastName, int age) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
    }

    @Override
    public String toString() {
        return "User [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + ", age=" + age + "]";
    }

    public static void main(String[] args) {
        UserWithThisKeyword user = new UserWithThisKeyword(1, "Ramesh", "Fadatare", 28);
        System.out.println(user.toString());
    }
}
```



THIS keyword

Using the keyword THIS we have the correct result:

Output:

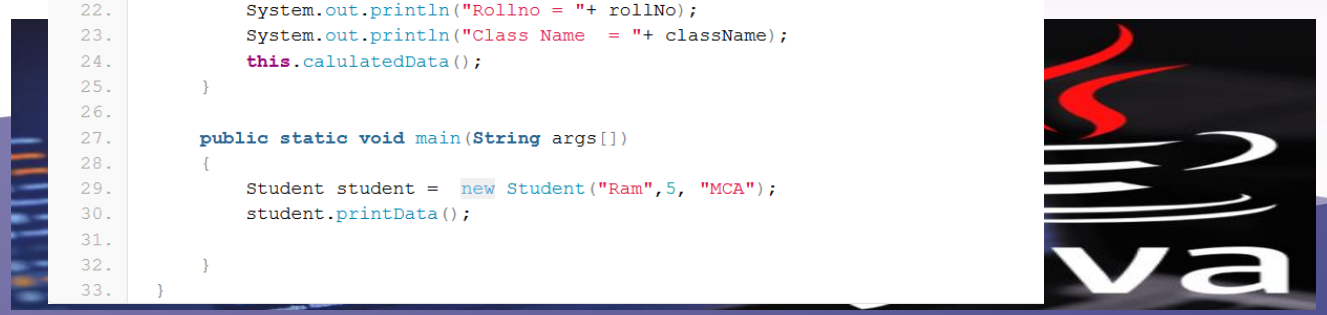
```
User [id=1, firstName=Ramesh, lastName=Fadatare, age=28]
```

THIS keyword

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

The below example demonstrates how to invoke the method of the current class by using `this` keyword.

```
1.  public class Student
2.  {
3.      String name;
4.      int rollNo;
5.      String className;
6.
7.      Student(String name, int rollNo, String className)
8.      {
9.          this.name = name;
10.         this.rollNo = rollNo;
11.         this.className = className;
12.     }
13.
14.     public void calulatedData()
15.     {
16.         System.out.println("This method is called by using of this keyword");
17.     }
18.
19.     public void printData()
20.     {
21.         System.out.println("Name = "+ name);
22.         System.out.println("Rollno = "+ rollNo);
23.         System.out.println("Class Name  = "+ className);
24.         this.calulatedData();
25.     }
26.
27.     public static void main(String args[])
28.     {
29.         Student student =  new Student("Ram",5, "MCA");
30.         student.printData();
31.
32.     }
33. }
```



NOTE : the keyword THIS is not necessary as the code works also without it.
It makes the code more clear

THIS keyword

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

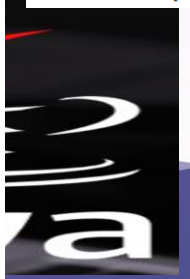
The below example demonstrates how to invoke the method of the current class by using `this` keyword.

An important point to refer current class variable

We can't use **this keyword** in a **static block** or **static method**. Because the **static block** or **static method** always binds with the class rather than objects.

```
1.  public class ExampleOfThisKeyword
2.  {
3.      int a = 10 ;
4.
5.      public static void setData(int a)
6.      {
7.          this.a = a;
8.      }
9.      public void showData()
10.     {
11.         System.out.println("Value of a = " + this.a);
12.     }
13.     public static void main(String arg[])
14.     {
15.         ExampleOfThisKeyword obj = new ExampleOfThisKeyword();
16.         setData(5);
17.         obj.showData();
18.     }
19. }
```

Output: *Exception in thread "main" java.lang.Error: Unresolved compilation problem: Cannot use this in a static context at ExampleOfThisKeyword.setData(ExampleOfThisKeyword.java:7) at ExampleOfThisKeyword.main(ExampleOfThisKeyword.java:16)*



THIS keyword

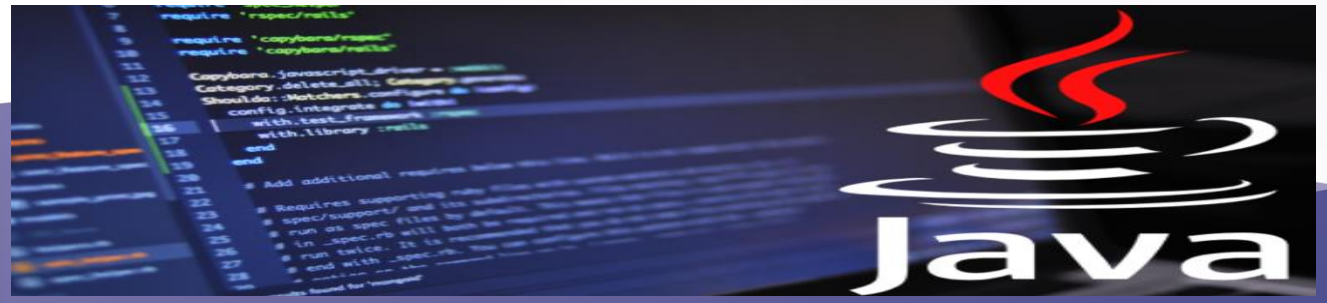
3. For calling other constructor of same class

"This" keyword in java can be used to call other constructor of same class.

Some times we have different constructors with different parameters then you can reuse existing constructor using "this" keyword.
We can call any type of constructor within any constructor. That is we can call default in parameterized, parameterized in defatul or parameterized in parameterized constructor.

Syntax: `this() or this(args)`

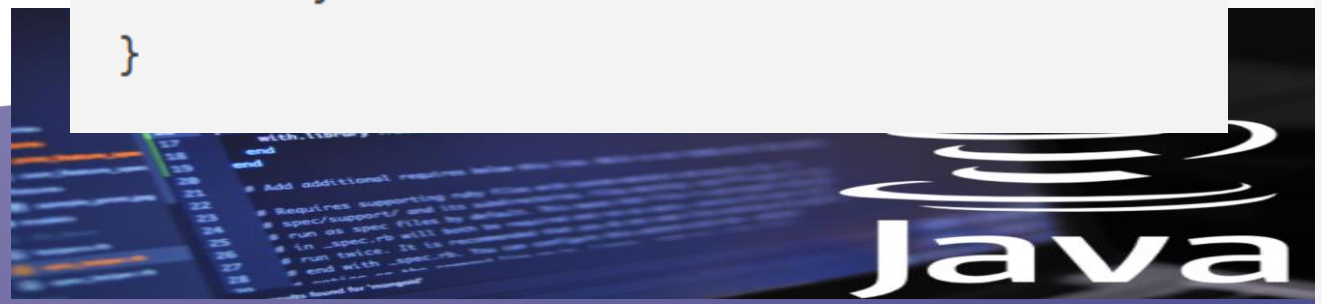
The most important thing to note is that the **use of "this" for calling other constructor should be the first line in constructor**.



THIS keyword

Example 1

```
public class SomeClass{
        private int variable ;

        public SomeClass(){
                // some code...
        }
        public SomeClass(int variable ){
                this()
                this.variable = variable ;
        }
}
```



Example 2

```
public class Shape{
        private int length ;
        private int breadth;

        public Shape(int length ,int breadth){
                this.length = length ;
                this.breadth = breadth;
        }

        public Shape(int length ){
                this(length,length);
        }
}
```

CHINA

THIS keyword

4. Using "this" keyword as return value in java

If you need to return current object instance we can use "this" keyword.

Builder design pattern is one of the most common example of this case.

```java
public class Person {
    private String name;

    // Constructor to initialize the name
    public Person(String name) {
        this.name = name;
    }

    // Method to set the name, returns 'this' for method chaining
    public Person setName(String name) {
        this.name = name;
        return this; // return the current instance of Person
    }

    // Method to get the name
    public String getName() {
        return name;
    }
```

```java
    public static void main(String[] args) {
        // Create a new Person object and use method chaining to set the name
        Person person = new Person("John")
                .setName("Alice");  // Chain the setName method

        // Print the updated name
        System.out.println("Person's name: " + person.getName()); // Output: Alice
    }
}
```

THIS keyword

5. Passing "this" keyword as argument to method

```
class MyClass {
    String name;

    MyClass(String name) {
        this.name = name;
    }

    void printThis() {
        System.out.println(this); // Print the reference to the current object
    }

    @Override
    public String toString() {
        return "MyClass[name=" + name + "]";
    }
}

public class Main {
    public static void main(String[] args) {
        MyClass obj = new MyClass("Alice");
        obj.printThis();  // Output: MyClass[name=Alice]
    }
}
```

- In the `printThis()` method, `this` refers to the current instance of `MyClass`.
- The `toString()` method is overridden to provide a human-readable representation of the object, so printing `this` will show something like `MyClass[name=Alice]`.