MOD -02 -> OOPs

this Keyword

this keyword

this Keyword

"*this*" is a keyword in java which refers to the current instance of the class.

1. *Every constructor* and all non-static methods in java have this as an implicitly parameter.
2. When n numbers of parameters are passed in a method then from JRE point of view, n+1 number of parameters are passed in the method. One additional parameter is *this*.
3. If *this* keyword is used in *constructor* or *method* then *this* must be the first statement.

Use of this keyword in java

1. this can be used to differentiate between instance variable and local variable.
2. this can be used in constructor chaining. this() and super() must be the first statement.
3. this can be used to invoke current class method implicitly.
4. this can be used to return current instance of the class.
5. this can be used as a parameter in constructor call.
6. this can be used as a parameter in method call.

Note:

1. this can be used inside any method to refer to the current object.
2. this is always a reference to the object on which the method was invoked.

1. this can be used to differentiate between instance variable and local variable.

Instance Variable Hiding

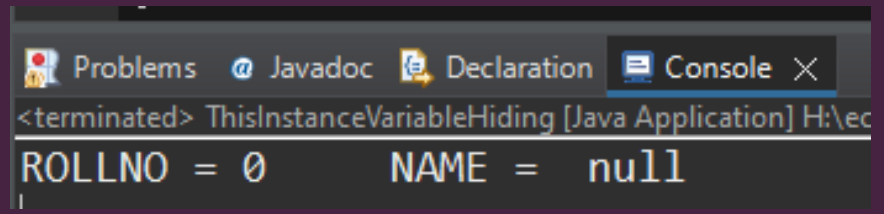
when a local variable has the same name as an instance variable, the local variable hides the instance variable (ie: compiler will not be able to distinguish them). “this” can be used to differentiate between instance variable and local variable. “this” lets you refer directly to the object.

```java
package thispack;

class Student {
    // instance variables
    int rollNo;
    String name;
    Student(int rollNo, String name) {
        rollNo = rollNo;
        name = name;
    }
    public void displayStudentDetails() {
        System.out.println("ROLLNO = " + rollNo + " NAME = " + name);
    }
}
public class ThisInstanceVariableHiding {

    public static void main(String[] args) {
        Student student = new Student(23, "Arun");
        student.displayStudentDetails();
    }
}
```

Output



How to avoid Instance Variable Hiding by using “this” keyword

```java
package thispack;
class Student {
    // instance variables
    int rollNo;
    String name;

    Student(int rollNo, String name) {
        this.rollNo = rollNo;
        this.name = name;
    }

    public void displayStudentDetails() {
        System.out.println("ROLLNO = " + rollNo + " NAME = " + name);
    }
 }
 public class ThisInstanceVariableHidingSolution {
    public static void main(String[] args) {
        Student student = new Student(23, "Arun");
        student.displayStudentDetails();
    }
}
```

2. this can be used in constructor chaining.

Constructor chaining is a process of calling one constructor from other constructor of a class on an object. Helps Cleaner, centralized initialization logic.

```
package thispack;

class Student {
    int rollNo;
    String name;

    Student() {
        this(23, "Arun");
    }

    Student(int rollNo) {
        this(rollNo, "Priya");
    }

    Student(int rollNo, String name) {
        this.rollNo = rollNo;
        this.name = name;
    }
```

```
    void showDetails() {
        System.out.println("Student Details:");
        System.out.println("Roll No: " + rollNo);
        System.out.println("Name: " + name);

    }

}

public class ThisConstructorChaining {

    public static void main(String[] args) {

        Student obj1 = new Student();
        obj1.showDetails();

        Student obj2 = new Student(50);
        obj2.showDetails();

        Student obj3 = new Student(60, "Bindu");
        obj3.showDetails();

    }
}
```

3. this can be used to invoke current class method implicitly.

Method chaining is a process of calling multiple methods on an object in a single statement.

```
package thispack;

class CurrentClass {
    public void displayRollNo() {
        System.out.println(23);
        //this.displayName(); //or
        displayName();
    }
    void displayName() {
        System.out.println("Bindu");
        //this.displayStream();
        displayStream();
    }
    private void displayStream() {
        System.out.println("BTech Civil");
    }
}
```

```java
public class ThisMethodChaining {


    public static void main(String[] args) {
        CurrentClass obj1=new CurrentClass();
        obj1.displayRollNo();
    }


}
```

4. this can be used to return current instance of the class

```
package thispack.old;


class Current {
    public Current getDisplay() {
        return this;// return the current class instance
    }
    public void print() {
        System.out.println("hi");
    }
}
public class ThisReturnCurrentInstance {


    public static void main(String[] args) {
        Current obj=new Current();
        obj.getDisplay().print();
}


}
```

5. this can be used as a parameter in constructor call.

```
package thispack;
class Demo {
    Demo(Test test) {
        System.out.println(test);
    }
}

class Test {
    Test() {
        // System.out.println("hi"); //allowed
        Demo d = new Demo(this);
    }
 }
public class ThisParameterInConstructorCall {
    public static void main(String[] args) {
        Test obj = new Test();
    }
}
```

6. this can be used as a parameter in method call.

```
package thispack.old;
class Trail {
    public void display() {
        System.out.println("hi");
        print(this);
    }
    public void print(Trail obj) {
        System.out.println(obj);
    }
}
public class ThisParameterMethodCall {
    public static void main(String[] args) {
        Trail trail=new Trail();
        trail.display();
    }
}
```

Thank you ☺ Happy coding ☺