2.8 this keyword in Java

❑ **What is *this***

- ***this*** is a keyword in Java. It can be used inside the M*ethod* or *constructor* of Class.
- It (***this)*** works as a reference to the current Object whose Method or constructor is being invoked.
- The ***this*** keyword can be used to refer to any member of the current object from within an instance Method or a constructor.

2.8.1 *this* keyword with field (Instance Variable)

- *this* keyword can be very useful in the handling of Variable Hiding.
- We cannot create two instance/local variables with the same name. However it is legal to create one instance variable & one local variable or Method parameter with the same name.
- In this scenario the local variable will hide the instance variable this is called Variable Hiding.
- **Example of Variable Hiding**

```
class JBT {
int variable = 5;
public static void main(String args[]) {
JBT obj = new JBT();

obj.method(20);
obj.method();
}
void method(int variable) {
variable = 10;
System.out.println("Value of variable :" + variable);
}
void method() {
int variable = 40;
System.out.println("Value of variable :" + variable);
}
}
```

Output:

```
Value of variable :10
Value of variable :40
```

- As you can see in the example above the instance variable is hiding and the value of the local variable (or Method Parameter) is displayed not instance variable.
- To solve this problem use ***this*** keyword with a field to point to the instance variable instead of the local variable.

Example of **this keyword** in Java for Variable Hiding

```
class JBT {
int variable = 5;
public static void main(String args[]) {
JBT obj = new JBT();
obj.method(20);
obj.method();
}
void method(int variable) {
variable = 10;
System.out.println("Value of Instance variable :" + this.variable);
System.out.println("Value of Local variable :" + variable);
}
void method() {
int variable = 40;
System.out.println("Value of Instance variable :" + this.variable);

System.out.println("Value of Local variable :" + variable);
}
}

Output:
Value of Instance variable :5
Value of Local variable :10
Value of Instance variable :5
Value of Local variable :40
```

2.8.2 *this* Keyword with Constructor

- ***"this"*** keyword can be used inside the constructor to call another overloaded constructor in the same Class. This is called the Explicit Constructor Invocation.
- This occurs if a Class has two overloaded constructors, one without argument and another with argument.
- Then the ***"this"*** keyword can be used to call constructor with argument from the constructor without argument. This is required as the constructor cannot be called explicitly.
- Example of ***this*** with Constructor

```
class JBT {
JBT() {
this("JBT");
System.out.println("Inside Constructor without parameter");
}
JBT(String str) {
System.out.println("Inside Constructor with String parameter as " +
str);
}
public static void main(String[] args) {
JBT obj = new JBT();
}
}
```

Output:

```
Inside Constructor with String parameter as JBT
Inside Constructor without parameter
```

As you can see "*this*" can be used to invoke an overloaded constructor in the same class.

Note*:

this keyword can only be the first statement in Constructor.

A constructor can have either ***this*** or ***super*** keyword but not both.

2.8.3 *this* Keyword with Method

this keyword can also be used inside Methods to call another Method from the same Class.

Example of this keyword with Method

```
class JBT {
public static void main(String[] args) {
JBT obj = new JBT();
obj.methodTwo();
}
void methodOne(){
System.out.println("Inside Method ONE");
}
void methodTwo(){
System.out.println("Inside Method TWO");
this.methodOne();   // same as calling methodOne()
}
}
```

Output:

```
Inside Method TWO
Inside Method ONE
```

Example of this keyword as Method parameter

```
public class JBTThisAsParameter {
public static void main(String[] args) {
JBT1 obj = new JBT1();
obj.i = 10;
obj.method();
}
}

class JBT1 extends JBTThisAsParameter {
int i;
void method() {
method1(this);
}
void method1(JBT1 t) {
System.out.println(t.i);
}
}
```