***** What is Variable Shadow in Java ?**

If class level variables and method level variables are having exactly same name then method level variable will **hide** class level variable **inside the method OR constructor OR block body.** This concept is known as Variable Shadow.

Case 1 :

```
package com.nit.variable_shadow;

class Test
{
    static int a = 100;  //Static Field
    int b = 200;         //Non static field

    public void printData(int a) //a is Parameter Variable
    {
        int b = 400; //Local Variable

        IO.println(a); //300
        IO.println(b); //400
    }
}

public class VariableShadow
{
    public static void main(String[] args)
    {
        Test t1 = new Test();
        t1.printData(300);
    }

}
```

Case 2 :

From the above program, It is clear that inside the method body always method level variables are having more priority. If we want to represent **class level fields inside the method OR block OR constructor in case of Variable hiding then we should use**

a) For Static field : We should use **class name**
b) For non static field : We should use **this keyword**

```
package com.nit.variable_shadow;

class Test
{
    static int a = 100;  //Static Field
    int b = 200;         //Non static field

    public void printData(int a) //a is Parameter Variable
    {
        int b = 400; //Local Variable

        IO.println(Test.a); //100
        IO.println(this.b); //200
    }
}

public class VariableShadow
{
    public static void main(String[] args)

    {
        Test t1 = new Test();
        t1.printData(300);
    }

}
```

***** What is Method Local Search algorithm in Java :**

- Whenever any method, block or constructor is executed then first of all compiler will search the variable in the method, block or constructor (At Method Level).

- If **Variable declaration** is not available at method level then Compiler will search in the class (class Level)

//Program + Diagram

```
package com.nit.variable_shadow;

class Test
{
    static int a = 100;  //ST
    int b = 200;  //HAV

    public void printData(int c) //c is parameter variable
    {
        int d = 400; //d is local variable

        IO.println("Static Field :"+Test.a);
        IO.println("Non Static Field :"+this.b);
        IO.println("Parameter Variable :"+c);
        IO.println("Local Variable :"+d);

    }
}

public class Method{cal}Search&Algo
{
    void main()
    {
        Test t1 = new Test();
        t1.printData(300);
    }
}
```

---

this keyword in Java : [In the platform (coderlib) + Online compiler, The OLC class should be in the top]

* this is a very special keyword in java which is used to refer the **current Object** as shown in the diagram.

```
public class ThisKeywordDemo
{
    public static void main(String[] args)
    {
        Customer scott = new Customer();
        scott.setCustomerData(111, "Scott");
        scott.getCustomerData();
    }
}
```

* By using this keyword, We can refer the non-static field & non-static method anywhere in the BLC class.

* this keyword, we **cannot** use from static context (From Static Method OR Static Block OR Static Nested class)

* IN JAVA, WHENEVER WE CREATE AN OBJECT THEN AUTOMATICALLY JAVA COMPILER WILL ADD final this to the very first parameter to all the non static method and constructor.

Customer.java

```
public class Customer
{
  private int id;

  public void setData(int id)
  {
    this.id = id;
  }

  public void getData()
  {
    IO.println(id);
  }
}
```

javac

Customer.class

```
public class Customer
{
  private int id;

  public void setData(final Customer this, int id)
  {
    this.id = id;
  }

  public void getData(final Customer this)
  {
    IO.println(this.id);
  }
}
```

```
package com.nit.this_keyword;

class Customer
{
    private int id;
    private String name;

    public void setCustomerData(int id, String name)
    {
        this.id = id;
        this.name = name;

    }

    public void getCustomerData()
    {
        IO.println("Customer Id is :"+this.id);
        IO.println("Customer Name is :"+this.name);
    }
}

public class ThisKeywordDemo
{
    public static void main(String[] args)
    {
        Customer scott = new Customer();
        scott.setCustomerData(111, "Scott");
        scott.getCustomerData();
    }
}
```