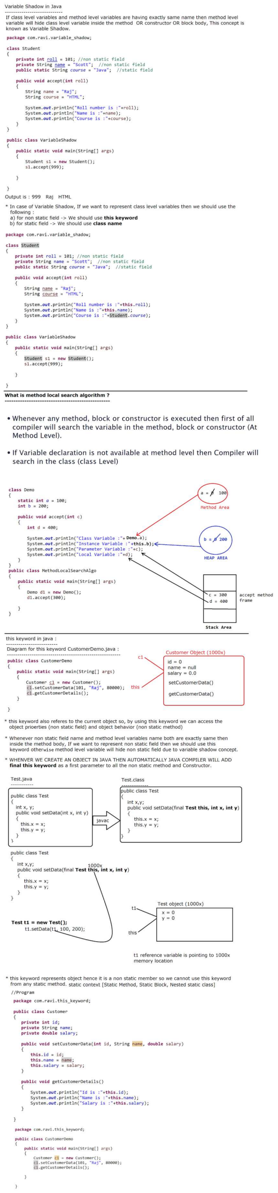

Variable Shadow in Java
-----------------------------

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.

```
package com.ravi.variable_shadow;

class Student
{
    private int roll = 101; //non static field
    private String name = "Scott";  //non static field
    public static String course = "Java";  //static field

    public void accept(int roll)
    {
        String name = "Raj";
        String course = "HTML";

        System.out.println("Roll number is :"+roll);
        System.out.println("Name is :"+name);
        System.out.println("Course is :"+course);
    }
}

public class VariableShadow
{
    public static void main(String[] args)
    {
        Student s1 = new Student();
        s1.accept(999);

    }

}
```

Output is : 999 Raj HTML

* In case of Variable Shadow, If we want to represent class level variables then we should use the following :
a) for non static field -> We should use **this keyword**
b) for static field -> We should use **class name**

```
package com.ravi.variable_shadow;

class Student
{
    private int roll = 101; //non static field
    private String name = "Scott";  //non static field
    public static String course = "Java";  //static field

    public void accept(int roll)
    {
        String name = "Raj";
        String course = "HTML";

        System.out.println("Roll number is :"+this.roll);
        System.out.println("Name is :"+this.name);
        System.out.println("Course is :"+Student.course);
    }
}

public class VariableShadow
{
    public static void main(String[] args)
    {
        Student s1 = new Student();
        s1.accept(999);

    }

}
```

**What is method local search algorithm ?**
-----------------------------------------------

- 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)

```
class Demo
{
    static int a = 100;
    int b = 200;

    public void accept(int c)
    {
        int d = 400;

        System.out.println("Class Variable :"+ Demo.a);
        System.out.println("Instance Variable :"+this.b);
        System.out.println("Parameter Variable :"+c);
        System.out.println("Local Variable :"+d);
    }
}
public class MethodLocalSearchAlgo
{
    public static void main(String[] args)
    {
        Demo d1 = new Demo();
        d1.accept(300);

    }
}
```

this keyword in java :
-------------------------

Diagram for this keyword CustomerDemo.java :
------------------------------------------------------

```
public class CustomerDemo
{
    public static void main(String[] args)
    {
        Customer c1 = new Customer();
        c1.setCustomerData(101, "Raj", 80000);
        c1.getCustomerDetails();
    }

}
```

* this keyword also referes to the current object so, by using this keyword we can access the object prioerties (non static field) and object behavior (non static method)

* Whenever non static field name and method level variables name both are exactly same then inside the method body, If we want to represent non static field then we should use this keyword otherwise method level variable will hide non static field due to variable shadow concept.

* WHENVER WE CREATE AN OBJECT IN JAVA THEN AUTOMATICALLY JAVA COMPILER WILL ADD **final this keyword** as a first parameter to all the non static method and Constructor.

Test.java
-----------

```
public class Test
{
  int x, y;
  public void setData(int x, int y)
  {
    this.x = x;
    this.y = y;
  }
}
```

Test.class
-------------

```
public class Test
{
   int x,y;
   public void setData(final Test this, int x, int y)
   {
      this.x = x;
      this.y = y;
   }
}
```

```
public class Test
{
   int x,y;
   public void setData(final Test this, int x, int y)
   {
      this.x = x;
      this.y = y;
   }
}
```

**Test t1 = new Test();**
t1.setData(t1, 100, 200);

t1 reference variable is pointing to 1000x memory location

* this keyword represents object hence it is a non static member so we cannot use this keyword from any static method. static context [Static Method, Static Block, Nested static class]

```
//Program

package com.ravi.this_keyword;

public class Customer
{
   private int id;
   private String name;
   private double salary;

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

   public void getCustomerDetails()
   {
      System.out.println("Id is :"+this.id);
      System.out.println("Name is :"+this.name);
      System.out.println("Salary is :"+this.salary);
   }

}

package com.ravi.this_keyword;

public class CustomerDemo
{
    public static void main(String[] args)
    {
        Customer c1 = new Customer();
        c1.setCustomerData(101, "Raj", 80000);
        c1.getCustomerDetails();

    }

}
```