Multiple Methods – Definition of Scope in Java

In previous examples, variables have been declared within the method, this however affects the usage of the variables in other methods within the class, effectively if a variable is declared within the method it can only be used within that method.

In fact if a variable is declared in a sub-block such as a for loop it can only be accessed purely within the braces it has been defined in.

To improve the access (or scope) for other methods, the Developer can choose to define the variables at class level (termed member variables or 'fields'), in the following example both methods can access the same variable defined at class level, note that because the methods are static, the variables must be made static too …

Multiple Methods – Definition of Scope in Java

```
private static int value;    // Making a field private means only this Class can use them
private static String text; // This is a Member variable or 'field'

public static void find_length()
{

    value = text.length();

 }

 public static void main(String[] args)
 {

   text = "How long is a piece of string?";
   System.out.println("The length returned is "+value);

 }
```

Multiple Methods – Definition of Scope in Java

There is a problem with scope, in that the compiler will allow a class level and a variable in a method defined with the same name, this means that 90 will always be printed in this example ...

```
private static int value;
private static String text;

public static void find_length()
        { value = text.length(); }

public static void main(String[] args)
        {
          int value = 90;
          text = "How long is a piece of string?";
          System.out.println("The length returned is "+value);
        }
```