

# Scope of Variables In Java

Difficulty Level : Easy Last Updated : 28 Jun, 2021

Scope of a variable is the part of the program where the variable is accessible. Like C/C++, in Java, all identifiers are lexically (or statically) scoped, i.e. scope of a variable can be determined at compile time and independent of function call stack.

Java programs are organized in the form of classes. Every class is part of some package. Java scope rules can be covered under following categories.

## Member Variables (Class Level Scope)

These variables must be declared inside class (outside any function). They can be directly accessed anywhere in class. Let's take a look at an example:

```
public class Test
{
    // All variables defined directly inside a class
    // are member variables
    int a;
    private String b;
    void method1() {....}
    int method2() {....}
    char c;
}
```

- We can declare class variables anywhere in class, but outside methods.
- Access specified of member variables doesn't affect scope of them within a class.
- Member variables can be accessed outside a class with following rules

Modifier	Package	Subclass	World
public	Yes	Yes	Yes
protected	Yes	Yes	No
Default (no modifier)	Yes	No	No

private	No	No	No
---------	----	----	----

### Local Variables (Method Level Scope)

Variables declared inside a method have method level scope and can't be accessed outside the method.

```
public class Test
{
    void method1()
    {
        // Local variable (Method level scope)
        int x;
    }
}
```

**Note :** Local variables don't exist after method's execution is over.

Here's another example of method scope, except this time the variable got passed in as a parameter to the method:

```
class Test
{
    private int x;
    public void setX(int x)
    {
        this.x = x;
    }
}
```

The above code uses this keyword to differentiate between the local and class variables.

As an exercise, predict the output of following Java program.

---

```
public class Test
{
    static int x = 11;
```

```
private int y = 33;
public void method1(int x)
{
    Test t = new Test();
    this.x = 22;
    y = 44;

    System.out.println("Test.x: " + Test.x);
    System.out.println("t.x: " + t.x);
    System.out.println("t.y: " + t.y);
    System.out.println("y: " + y);
}

public static void main(String args[])
{
    Test t = new Test();
    t.method1(5);
}
}
```

## Output:

```
Test.x: 22
t.x: 22
t.y: 33
y: 44
```

## Loop Variables (Block Scope)

A variable declared inside pair of brackets “{” and “}” in a method has scope within the brackets only.

---

```
public class Test
{
    public static void main(String args[])
    {
        {
            // The variable x has scope within
            // brackets
            int x = 10;
            System.out.println(x);
        }
    }
}
```

```
// Uncommenting below line would produce
// error since variable x is out of scope.

// System.out.println(x);
}
```

### Output:

10

As another example, consider following program with a for loop.

---

```
class Test
{
    public static void main(String args[])
    {
        for (int x = 0; x < 4; x++)
        {
            System.out.println(x);
        }

        // Will produce error
        System.out.println(x);
    }
}
```

### Output:

```
11: error: cannot find symbol
      System.out.println(x);
```

The right way of doing above is,

---

```
// Above program after correcting the error
class Test
{
    public static void main(String args[])
    {
        int x;
        for (x = 0; x < 4; x++)
        {
            System.out.println(x);
        }

        System.out.println(x);
    }
}
```

## Output:

0  
1  
2  
3  
4

Let's look at tricky example of loop scope. Predict the output of following program. You may be surprised if you are regular C/C++ programmer.

---

```
class Test
{
    public static void main(String args[])
    {
        int a = 5;
        for (int a = 0; a < 5; a++)
        {
            System.out.println(a);
        }
    }
}
```

## Output :

```
6: error: variable a is already defined in method go(int)
    for (int a = 0; a < 5; a++)
        ^
```

1 error

Note:- In C++, it will run. But in java it is an error because in java, the name of the variable of inner and outer loop must be different.

A similar program in C++ works. See [this](#).

As an exercise, predict the output of the following Java program.

---

```
class Test
{
    public static void main(String args[])
    {
        {
            int x = 5;
            {
                int x = 10;
                System.out.println(x);
            }
        }
    }
}
```

Q. From the above knowledge, tell whether the below code will run or not.

---

```
class Test {
    public static void main(String args[])
    {
        for (int i = 1; i <= 10; i++) {
            System.out.println(i);
        }
        int i = 20;
    }
}
```

```
        System.out.println(i);  
    }  
}
```

### Output :

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
20
```

Yes, it will run!

See the program carefully, inner loop will terminate before the outer loop variable is declared. So the inner loop variable is destroyed first and then the new variable of same name has been created.

### Some Important Points about Variable scope in Java:

- In general, a set of curly brackets { } defines a scope.
- In Java we can usually access a variable as long as it was defined within the same set of brackets as the code we are writing or within any curly brackets inside of the curly brackets where the variable was defined.
- Any variable defined in a class outside of any method can be used by all member methods.
- When a method has the same local variable as a member, “this” keyword can be used to reference the current class variable.
- For a variable to be read after the termination of a loop, It must be declared before the body of the loop.

This article is contributed by **Rishabh Mahrsee**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to [review-team@geeksforgeeks.org](mailto:review-team@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above