Lesson:

Java methods and scope of variables

PW SKILLS

Pre-Requisites

- Java classes and objects
- Basics of methods

List of Concepts Involved

- Scope of Variables in Java
- Method call by value
- Method call by reference

Variables are the most versatile part of the codes because they not only enable us to take values but also share it all across the program/application. It is therefore, extremely important to know some rules about the legality of access of variables across the program. Yes, you cannot use all kinds of variables freely in a large scale program.

To give you an insight to these rules of accessing a variable, we will start the lesson with the concept of 'scope'.

Topic: Scope of Variables In Java

Scope of a variable is the part of the program where the variable can be accessed. The various types of scopes that Java supports are as follows :

1. <u>Member Variables (Class Level Scope)</u>

- These are the variables that are declared inside the class
- These variables can be anywhere in class, but outside methods.
- Java allows us to access member variables outside the class with the access modifier rules:(will be taught in oops classes)

Example:

```
public class Main
{
  // All variables defined directly inside a class
  // are member variables
  int score;
  private String playerName;
  void showScore() {....}
  int getScore() {....}
}
```

2. Local Variables (Method Level Scope)

- Variables declared inside a pair of curly braces {} have block-level scope.
- Variables declared inside a method have method level scope
- These variables can't be accessed outside the method.
- These variables don't exist after the method's execution is over.

Example:

```
public class Main
{
   void game()
  {
  // Local variable (Method level scope)
  int score;
  }
}
```

3. Loop Variables (Block Scope)

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

Example:

```
public class Main
{
  public static void main(String args[])
  { for(int score=0;score<100;score++){
      // The variable score has scope within
      // brackets
      System.out.println(score);
    }
    // Uncommenting below line would produce
    // error since variable score is out of scope.
    // System.out.println(score);
  }
}
```

Note : Within the same scope, no two variables can be declared with the same name. However, it is possible for two variables to have the same name if they are declared in different scope.

Example: .

```
public static void main(String[] args)
  {
  int p = 50;
  double p = 60.70;
  System.out.println(p);
  }
```

For the above code, there are two variables with the same name p in the same scope of main () function. Hence, the code will not compile.

Before moving ahead to the next concept, let us first understand an important concept parameters that will be used to pass the variables in methods.

There are two types of parameters one is Formal parameters and the second is Actual parameters.

Formal parameters: These are the parameters that are defined during function/method definition.

Actual parameters: These are the parameters which are passed during the function/method call in other Function.

Let us look at the example below to understand this.

Example:

```
// Java program to show the difference
// between formal and actual parameters
import java.io.*;
class Main {
  static int diff(int p,int q) // Formal parameters
  {
    return p-q;
  }
  public static void main (String[] args) {
    int p=89;
    int q=9;
    System.out.println(diff(p,q));
  //This is actual parameters
    }
}
```

Output: 80

This parameter passing in methods is done in two ways, let us look at each of them.

Topic: Pass by Value and Pass by Reference in Java

Pass by Value and Pass by reference are the two ways by which we can pass a value through a variable in the function/method.

- **Pass by Value:** It is a process in which the function parameter values are simply copied to another variable(from actual to formal parameters). This is known as call by value.
- **Pass by Reference:** It is a process in which the copy of reference of actual parameter is passed to the function. This is called by call by reference.

Note: Java is officially always pass-by-value i.e when any variable is passed to a method in Java, the value of the variable on the stack is copied into a new variable inside the new method.

We will concentrate our attention on ‘pass by value’ for now.

Example: 1. Pass by value

```
public static void main(String[] args) {
  int a = 40;
  System.out.println(a); // prints "40"
  changeValue(a);
  System.out.println(a); // prints "40"
}
  public static void changeValue(int z) {
    ...
    z = 100; // changeValue has a copy of z, so it doesn't
      // overwrite the value of variable a used
      // in main() that called changeValue
}
```

Example 2. Pass by value

```
import java.util.Scanner;
public class Main
{
public static void main(String args[])
{
int a, b, ans;
Scanner sc = new Scanner(System.in);
System.out.print("Enter the first number: ");
a = sc.nextInt();
System.out.print("Enter the second number: ");
b = sc.nextInt();
ans = add(x, y);
System.out.println("The sum of two numbers a and b is: " + ans);
}
//method that calculates the sum
public static int add(int n1, int n2)
{
int ans = n1 + n2;
return ans;
}
}
```

In the above example, we are passing two values a and b entered by the user in a function add which is returning the sum of two numbers i.e a and b.

Q1.What is the output of the below code?

```
public static void decrease(int n1, int n2){
  n1--;
  n2 = n2 - 2;
  System.out.println(n1 + ":" + n2);
  // n1 and n2 are formal parameters
}
public static void main(String[] args) {
  int p = 26;
  int q = 13;
  decrease(p,q);
  System.out.println(p + ":" + q);
 // p and q are actual parameters.
}
```

Ans:
25: 11
26: 13

Explanation:
For the above code, changes in the values of n1 and n2 are not reflected to p and q because n1 and n2 are formal parameters and are local to function decrement so any changes in their values here won't affect variables p and q inside main.

Q2 What will be the output of the following code ?

```
public static void makeTwice(int p ){
  p = p * 2;
}
public static void main(String[] args) {
  int p = 24;
  makeTwice(p);
  System.out.println(p);
}
```

Ans : 24

Explanation:
For the above code, changes in the values of p is not reflected in main function
Here, we are calling the function makeTwice using pass by value.

Q3.Will the following code generate any error ?

```
public class Main {
  public static void temp(int p) {
    int q = p;
    q = q -100;
  }
  public static void main(String[] args) {
    int p = 890;
    temp(p);
    System.out.println(q);
  }
}
```

Ans: Yes,it will generate an error.

Explanation: Because in the main function there is no variable having name q,variable q has scope in temp function block only.

That is all for this lesson. See you in the next class !
Till then, happy learning !!

Upcoming Class Teasers

- Arrays

✓

21/08/2023