17/09/2021

Methods

Function / Methods (in JAVA):

- A method is block of code which only runs when it is called.
- to reuse code: define the code once and use it many times.

Syntax

```
public class Demo {
    static void myMethod () {
        // code
    }
}
```

(return type → void; name of method → myMethod)

Return Type

- A return statement causes the program flow to transfer back to caller of the method.

Pass by Value

Eg 1.

```
main () {
    name = "a";
    greet (name);
}

static void greet (naam) {
    print (naam);
}
```

(object value → "a"; name → a; naam → a)

since it is created inside func. it will not change original value

not changing original object just creating new object

__/__/____

Points to remember

1. primitive data type like int, short, char, byte, etc.
   ↳ just pass value

2. object & reference:
   ↳ passing value of reference ~~value~~ variable.

Eg-1

```
psvm () {
    a = 10;                    a → 10
    b = 20;                    b → 20
    swap (a, b);
}
swap (num1, num2) {
    temp = num1;               temp → 10
    num1 = num2;               num1 → 20
    num2 = temp;               num2 → 10
}
```

at fn scope they are swapped.

Eg-2

arr ⇒ [1, 2, 3, 4, 5]

nums ↗

num[0] = 99. [now value of 0th position will change in arr & nums too]

__/__/____

Scopes

- **function scope**: variable declared a method / function scope (inside the method) cannot be accessed outside the method.

- **block scope**:

```
psvm () {
    int a = 10;
    int b = 20;
    {
        int a = 5; ✗
        a = 5;
        int c = 20;
    }
    c = 10 ✗
    int c = 10;
    a = 50;
}
```

← variables initialized outside the block can be updated inside the box

← variable initialized inside the block cannot be updated outside the box but can be reinitialized outside the box.

↓ variable like 'a' here is declared outside the block, updated inside the block and can also be updated outside the block.

- **loop scope**: variable declared inside loop are having loop scope.

__/__/____

Shadowing

Shadowing in JAVA is the practice of using variable in overlapping scopes with same name where the variable in low level scope overrides the variable of high level scope. Hence variable at high level scope is shadowed by low level scope variable.

Eg

```
psvm {
    static int x = 90;
public class shadowing {
    static int x = 90;
    psvm () {
        sopln (x);          // 90
        x = 50;             // Here high level scope is
        sopln (x);          // shadowed by low level scope.
    }                       // 50
}
```

Variable Argument: Variable argument is used to take a variable number of argument. A method that take variable number of argument is called varang method.

Syntax

```
static void fun (int ... a) {
    // method body
}
```

__/__/____

Method Overloading

Function overloading happen when two function have same name.

```
fun (int a) {
        // code
  }
fun ( int a, int b) {
    // code
  }
```

* at compile time it decide which function to run