Page No.: Date:

FUNCTIONS/METHODS

Function/Method-
A method is a block of code which only runs when it is called
To reuse code: define the code once, use it many times

Syntax-
```
access_modifier return_type method_name()
{
    //code
    return statement; ⟶ function ends here
}
```

The type of data returned by a method must be compatible with the return type specified by the method.

Pass by value-
Eg1-
```
class Main{
    psvm(){
        name = "a";                    name ⟶ a
                                       naam ⟶
        greet(name);  ⟶ calling function
    }                 creating copy of value of name
                      ie passing value by ref.
    static void greet(naam){ ⟶ name= naam= "a"
        print(naam)
    }                 O/p - a
}
```
O/P- a

Eg2-
```
class Main{
    psvm(){
        name= "a";
        change(name); //calling function
        print(name)
    }
```

name
| creating copy

$$\begin{tikzcd} \text{name} \arrow[r] & a \\ \text{naam} \arrow[ur] & \end{tikzcd}$$

```
static void change(naam) {
    naam = "b"; //creating new object
}
}
```

$$\begin{tikzcd} \text{name} \arrow[r] & a \\ \text{naam} \arrow[r] & b \end{tikzcd}$$

O/p a

Since it is created inside function it will not change orignal value.

Primitive datatypes	Just pass value
Object and Reference	passing value of reference variable

Eg:
```
public static void main(String[] args) {
        int []arr = {1, 2, 3};
        change(arr);
        System.out.println(Arrays.toString(arr));
}
static void change(int[] nums) {
        nums[0] = 99; // If you make a change to the object
  }                   via this ref variable same object will be
                      changed
```
O/P 99, 2, 3

$$\begin{tikzcd} \text{arr} \arrow[r] & {[\,1,\ 2,\ 3\,]} \arrow[r, "99"] & \\ \text{nums} \arrow[ur] & \text{nums[0]} & \end{tikzcd}$$

- Function scope – Variables declared inside function/method cant be accessed outside the method.
- Block scope

```
psvm() {
    int a = 10; → Var. initialized outside block can be updated inside
    {
        int a = 5; X      Var. initialized inside the block cannot be
        a = 100; ✓        updated outside the block but can be
        int c = 20;       reinitialized outside the block.
    }
    c = 10; X
    int c = 15; ✓
    a = 50; ✓ → declared outside can be update outside.
}
```

- Shadowing -
Shadowing in Java is the practice of using variables in overlapping scopes with the same name where the variables in low-level scope overrides the variable at high level scope.
Eg-
```
public class Shadowing{
        static int x=90;
          public static void main(String[] args){
              System.out.println(x); → 90
              x = 50;
       // here high level scope is shadowed by low-level scope
              System.out.println(x); → 50
          }
}
```
O/P 90
50

- Variable Arguments
Variable Arguments is used to take a variable number of arguments. A method that takes a argument of variable number is a varargs method.
Syntax-
```
    static void sub (int ... a) {
        //method body
    }
```
Here would be array of type int[] parameters