Module no : 7 Functions Date = 18/10/2023

Agenda :-

1. Definition and syntax
2. Arguments
3. Return types
4. Scope of a variable
5. Call stack & Heap Memory
6. Variable Arguments.

Introduction :-

Definition : Java functions are also known as methods. They are the blocks of code that are only executed when they are called.
We can pass values to methods known as parameters, and return values from them as well.

Benifits of functions / Methods: They allow us to divide our code in logical blocks.
They allow us to reuse code.

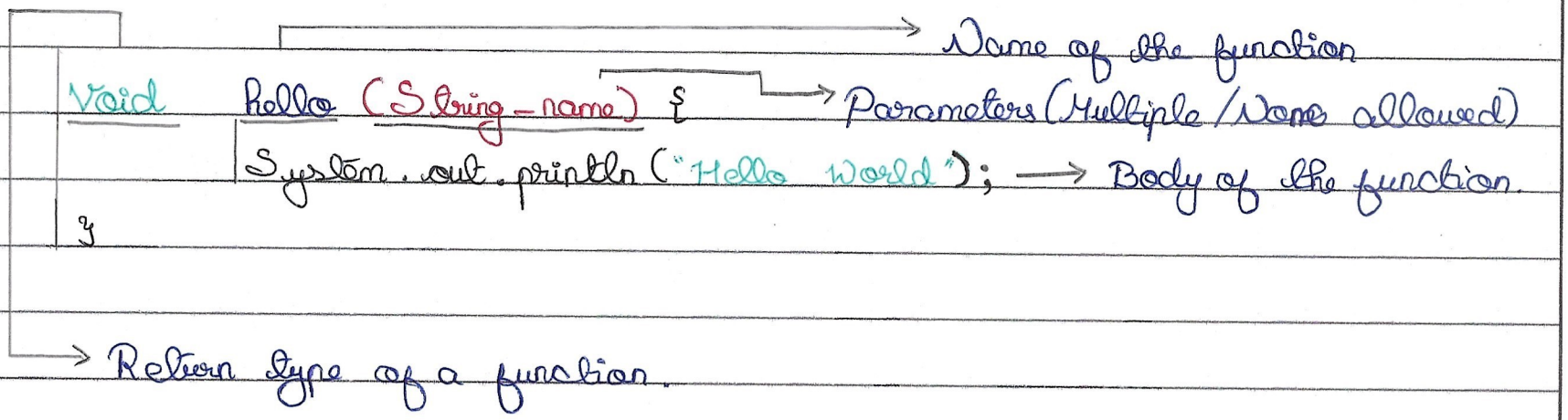
Syntax:

```
class Main {
    static void add() {
        // do Something
    }
    public static void main(String [] args) {
        int a = 10 ; b = 20 ; add(a, b);
    }
}
```

} function

→ calling the function.

Proper Syntax of a function :-



Functions in Java:

Program:

```
package com.company;
public class Main {
    // Creating a Tea Function
    static void tea(){
        System.out.println("Boil water with tea leaves and sugar");
        System.out.println("Add milk and boils for few minutes");
        System.out.println("Serve the milk tea");
    }
    // Creating a Coffee function
    static void coffee(){
        System.out.println("Boil some Milk");
        System.out.println("Put some coffee in the cup");
        System.out.println("Pour the milk into the cup");
    }
    public static void main(String args[]){
        // Calling the tea function
        System.out.println("Recipe for tea:");
        tea();
        //calling the coffee function
        System.out.println("Recipe for Coffee:");
        coffee();
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe"
Recipe for tea:
Boil water with tea leaves and sugar
Add milk and boils for few minutes
Serve the milk tea
Recipe for Coffee:
Boil some Milk
Put some coffee in the cup
Pour the milk into the cup

Process finished with exit code 0
```

Parameters :-

Definition :- Methods can accept values while being called. These values are stored in local variables known as parameters or arguments. A method can have multiples or none arguments. The data type of each argument must be defined in the function signature.

Sample Program :-

```
void introduce(String name, int age, String [] hobbies) {
        System.out.println("My name is " + name);
        System.out.println("I am" + age + " years old");
        System.out.println("My hobbies are :");
        for (String hobby : hobbies) {
                System.out.println("- " + hobby);
        }
}
```

Functions and Parameters / Arguments in Java:

Program:

```
package com.company;

public class Main {
    public static void main(String[] args) {
        String name = "Aswin";
        int age = 19;
        String []hobbies = {"Eat","Sleep","Code","Repeat"};
        introduce(name, age, hobbies);
    }
    static void introduce(String name, int age, String[] hobbies){
        System.out.println("My name is "+name);
        System.out.println("I am "+age+" years old");
        System.out.println("My hobbies are:");
        for (String hobby : hobbies){
            System.out.println("-"+hobby);
        }
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe"
My name is Aswin
I am 19 years old
My hobbies are:
-Eat
-Sleep
-Code
-Repeat

Process finished with exit code 0
```

Example : Square of a number :-

Task : Write a function which computes and prints the square of a number.

Sample input : N = 5

Sample Output : 25.

Functions in Java – Square of the Number:

Program:

```
package com.company;

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        System.out.println("Enter the number N:");
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        square(N);
    }
    static void square(int N){
        int result = N * N;
        System.out.println("The Square of "+N+" is: "+ result);
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe"
Enter the number N:
5
The Square of 5 is: 25

Process finished with exit code 0
```

Return Types :-

Definition :- Java methods can also optionally return some values back to the caller using the return statement. The data type of the value being returned must be specified in the method signature beforehand. The return type can be any valid data type in Java. In case no values to be returned, specify the return type as void.

Sample program :-

```
int square(int num) {
    return num * num;
}
```

→ Return type

→ Return Statement

→ The return type of this function is int

→ The function returns the square value of num

Functions in Java – Return Statement:

Program:

```
package com.company;

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        System.out.println("Enter the number N:");
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        System.out.println(square(N));
    }
    static int square(int N){ // int or string or anything which should match with the return value
        int result = N * N;
        return result;
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe"
Enter the number N:
100
10000

Process finished with exit code 0
```

Method Overloading :-

Definition :- Method Overloading is the act of having multiple methods having same name but different parameters. It increases the readability of the program.

Sample Program :-

```
class Main {
    static int add (int a, int b) {
        System.out.println ("Inside first add");
        return a+b;
    }
    static String add (String a, String b){
        System.out.println ("Inside Second add");
        return a+b;
    }
    public static void main (String [] args) {
        System.out.println (add (5,4));
        System.out.println (add ("Hello","World"));
    }
}
```

Functions in Java – Method Overlapping:

Program:

```
package com.company;

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        System.out.println(add(5,4));
        System.out.println(add("5","4"));
    }
    static int add(int a, int b){
        System.out.println("Inside the first add");
        return a+b;
    }
    static String add(String a, String b){
        System.out.println("Inside the second add");
        return a+b;
    }
}
```

Output:

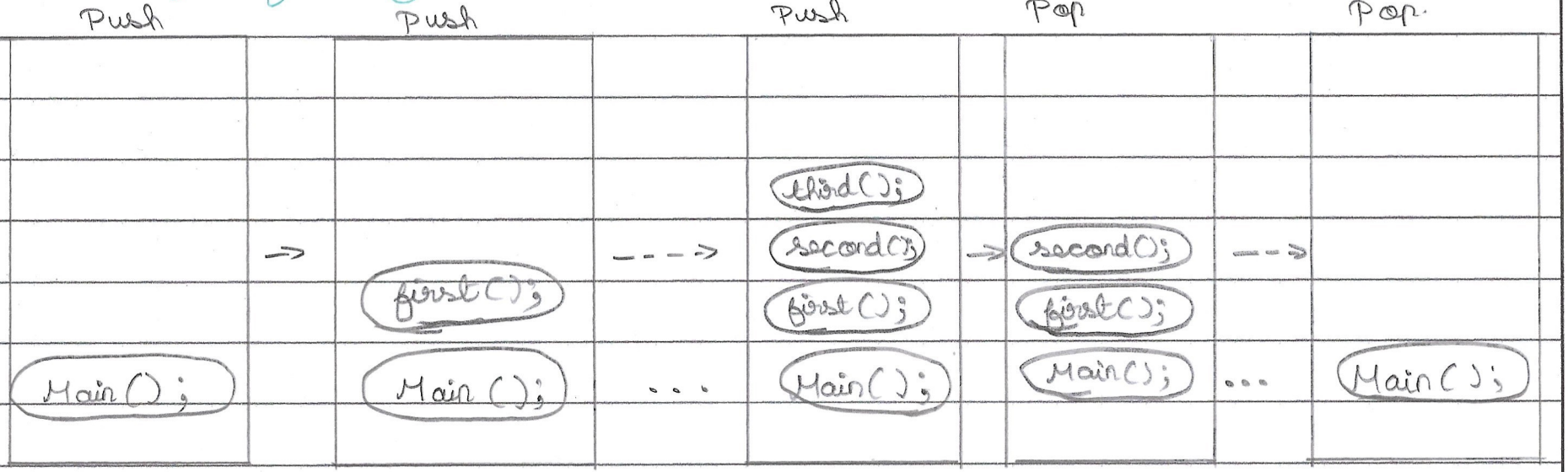
```
"C:\Program Files\Java\jdk-21\bin\java.exe"
Inside the first add
9
Inside the second add
54

Process finished with exit code 0
```

Call Stack :-

Definition :- The call stack is what a program uses to keep track of method calls. The call stack is made up of stack frames (i.e.,) one for each method call.

Call Stack for following Example :-



Functions in Java – Call Stacks:

Program:

```
package com.company;

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        int x = 0;
        System.out.println("Inside main(), x = "+ x);
        first();
    }
    static void first(){
        int x = 10;
        System.out.println("Inside first(), x = "+ x);
        second();
    }
    static void second(){
        int x = 20;
        System.out.println("Inside second(), x = "+ x);
        third();
    }
    static void third(){
        int x = 30;
        System.out.println("Inside third(), x = "+ x);
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe"
Inside main(), x = 0
Inside first(), x = 10
Inside second(), x = 20
Inside third(), x = 30

Process finished with exit code 0
```

Scope of a Variable :-

Definition :- The scope of a variable is the region of the program where it is accesible. The types of scope levels in Java are class level scope and Block level scope.

Sample program :-

```
class Main {
    public static void main (String [] args) {
        // int a = 5 ;
        if (true) {
            a = 10;
            System.out.println("Inside if, a=" + a);
        }
        System.out.println("outside if, a = " + a);
    }
}
```

block scope.

gets error. ←

to correct it uncomment "int a = 5;"

Variable Arguments - Varargs :-

Definition :- In java, an argument of a method can accept arbitary number of values. This argument that can accept variable number of values is called varargs.

Syntax :-

```
return_type  methodName(dataType... args) {
        // body
}
```

Functions in Java – varargs – Variable Arguments:

Program:

```
package com.company;

public class Main {
    public static void main(String[] args) {
        float avg1 = getAvg(2,3,4,5,6,7);
        float avg2 = getAvg(2,3,4);
        System.out.println("avg1 = "+avg1);
        System.out.println("avg2 = "+avg2);
    }
    static float getAvg(float ... varargs){ // varargs can be renamed as optional
        float total = 0;
        for (float num :varargs){
            total += num;
        }
        return total;
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-21\bin\java.exe"
avg1 = 27.0
avg2 = 9.0

Process finished with exit code 0
```