ASIF.I/O

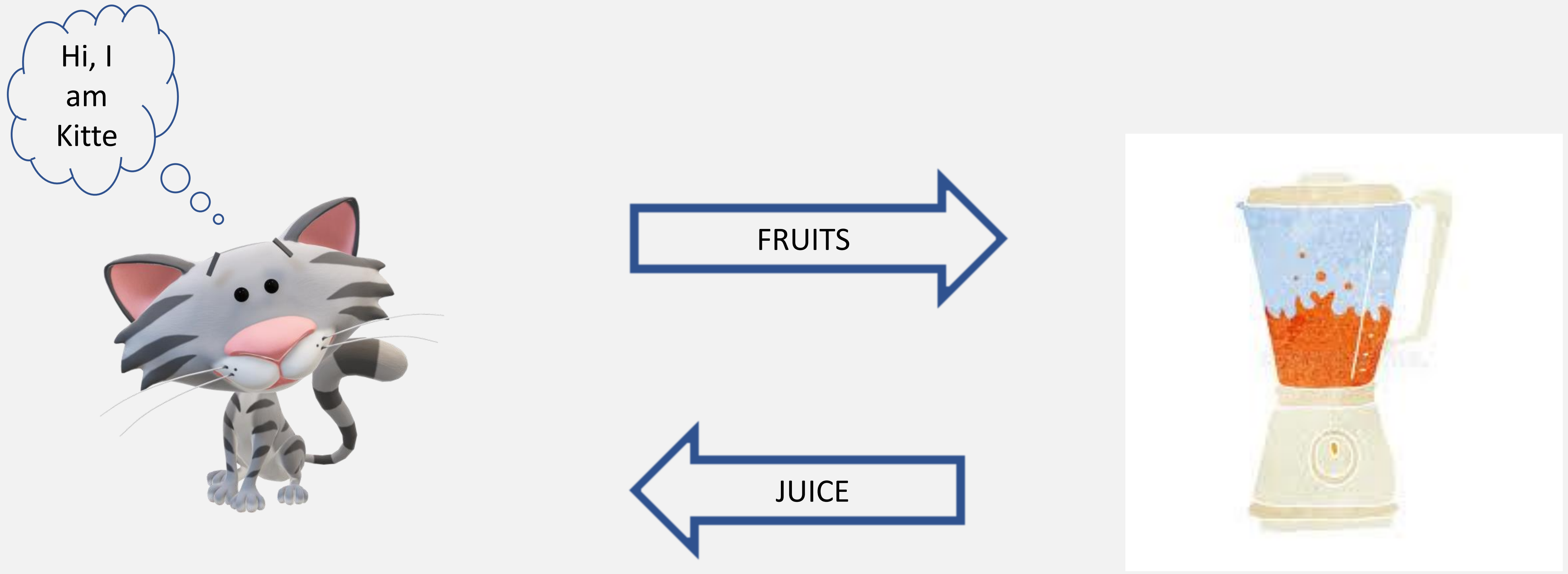
# Introduction to Methods

E-mail : **asif.io.edu@gmail.com**

## Methods

**Methods/Functions are block of codes which do some specific task and run only when it is called.**

**Methods/Function are block of codes which so some specific task and run only when called.**



# Method with Arguments

```
public void eat(){
    System.out.print("Eating something");
}


ob.eat();          //calling the method
ob.eat();
```

```
Eating something
Eating something
```

```
public void eat( String food){
    System.out.print("Eating "+food);
}


ob. eat("Pizza");       //calling the method
ob. eat("Biryani");
```

```
Eating Pizza
Eating Biryani
```

# Method with return value

```
public String eat(){
    System.out.print(“Feeling so Hungry”);
    return “Eating something”;
}


String result = ob.eat();      //calling the method
System.out.print(result);
```

```
Feeling so Hungry
Eating something
```

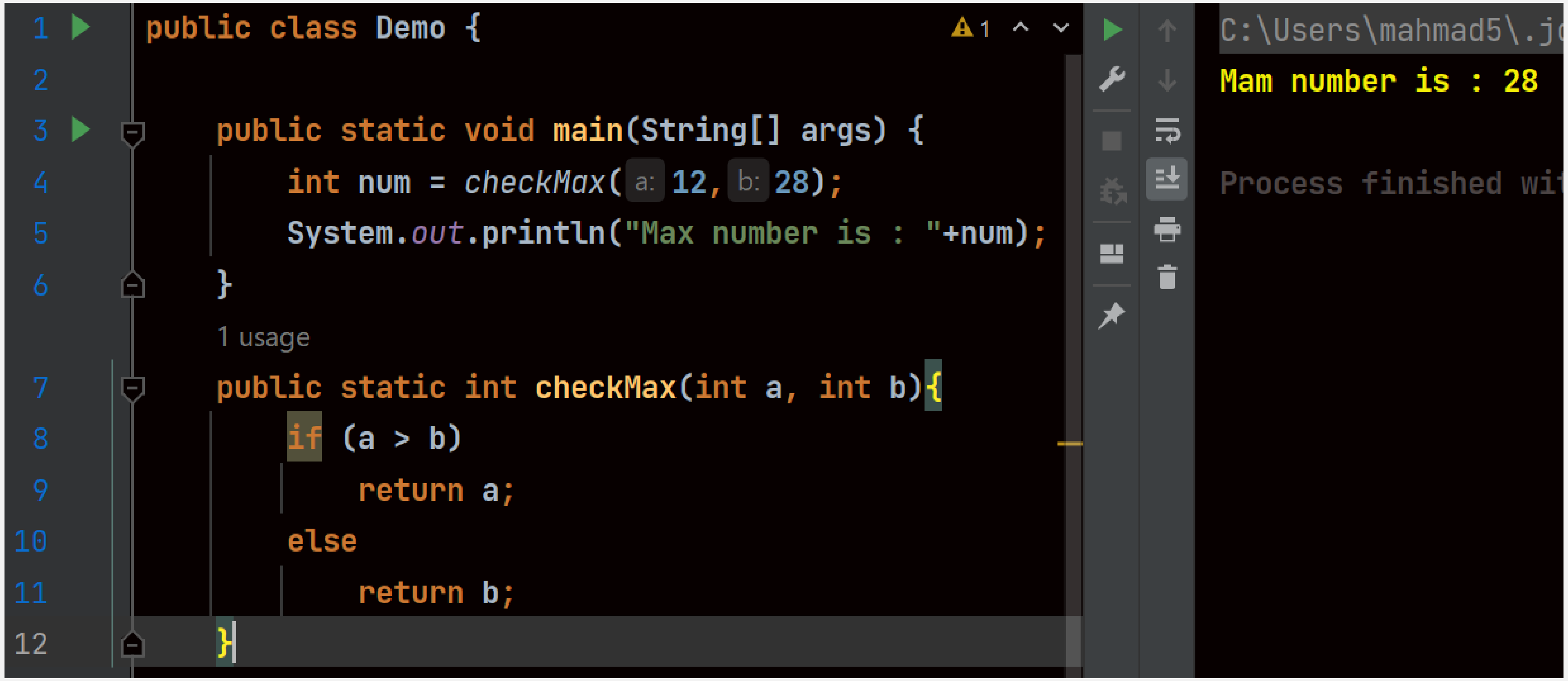
```
public String eat( String food){
    System.out.print(“Feeling so Hungry);
     return “Eating ”+food;
}


String result =ob.eat(“Pizza”);
System.out.print(result);
```

```
Feeling so Hungry
Eating Pizza
```

# Return Statement

- return keyword is used to terminate the method.
- No statement in a method block will be executed after the return statement.
- return statement can only return a single value from a method.
- In case of multiple return statement only one return statement will be executed.
- return type methods are also call pure methods



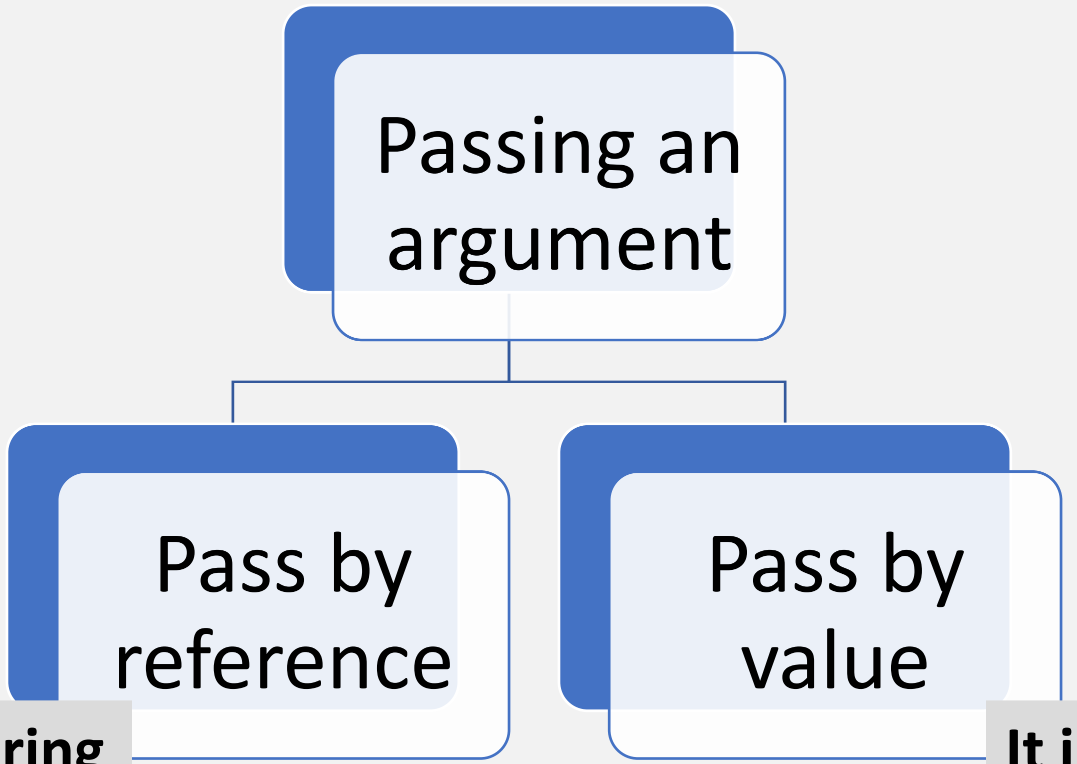
```java
public class Demo {

    public static void main(String[] args) {
        int num = checkMax( a: 12, b: 28);
        System.out.println("Max number is : "+num);
    }
    1 usage
    public static int checkMax(int a, int b){
        if (a > b)
            return a;
        else
            return b;
    }
```

```
C:\Users\mahmad5\.jc
Mam number is : 28

Process finished wi
```

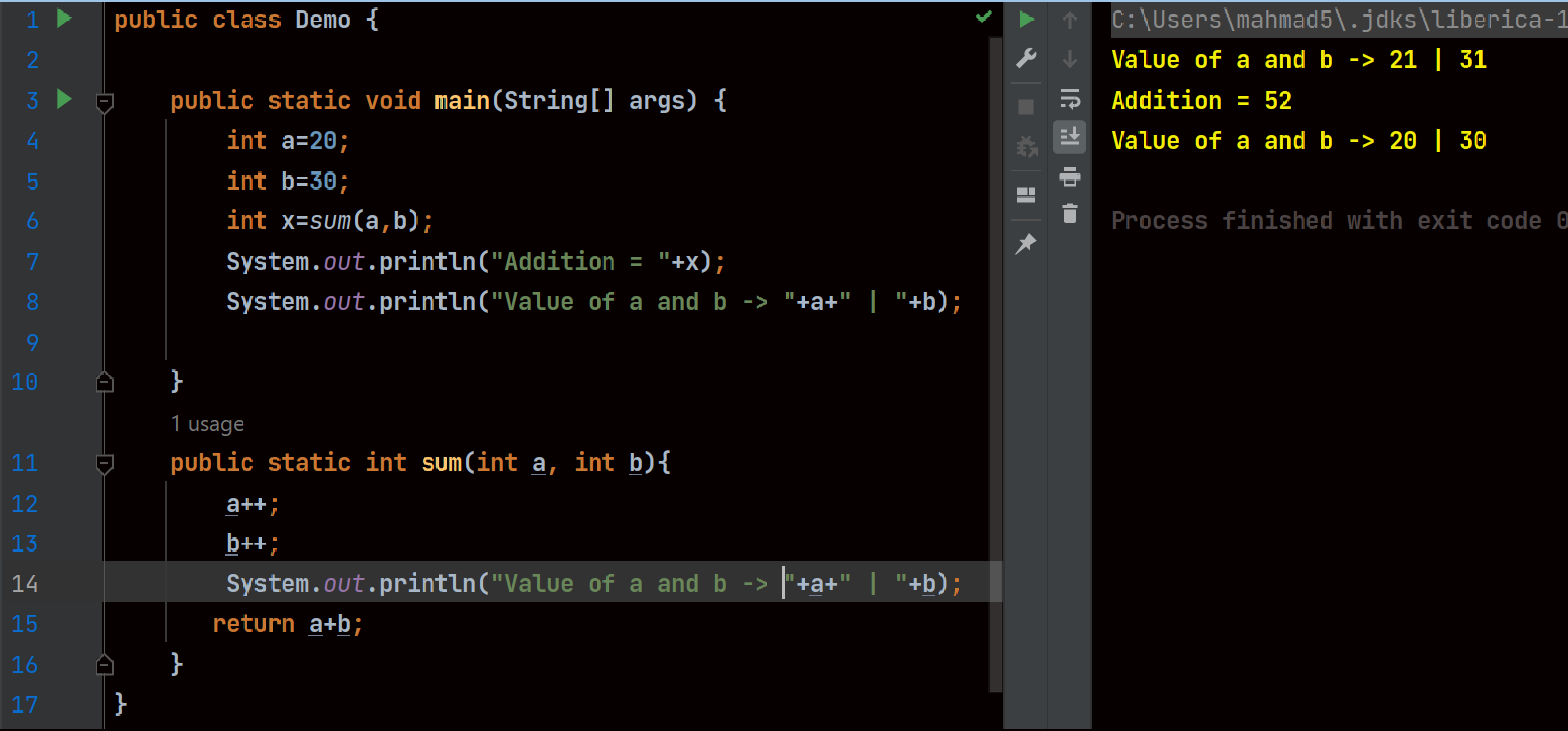
## Passing the argument



**It is the process of sharing address of actual parameters with the formal parameters.**

**It is a process of passing the copy of the actual parameters to the formal parameters.**

# Code - Pass by value



```java
public class Demo {

    public static void main(String[] args) {
        int a=20;
        int b=30;
        int x=sum(a,b);
        System.out.println("Addition = "+x);
        System.out.println("Value of a and b -> "+a+" | "+b);

    }
    1 usage
    public static int sum(int a, int b){
        a++;
        b++;
        System.out.println("Value of a and b -> "+a+" | "+b);
       return a+b;
    }
}
```

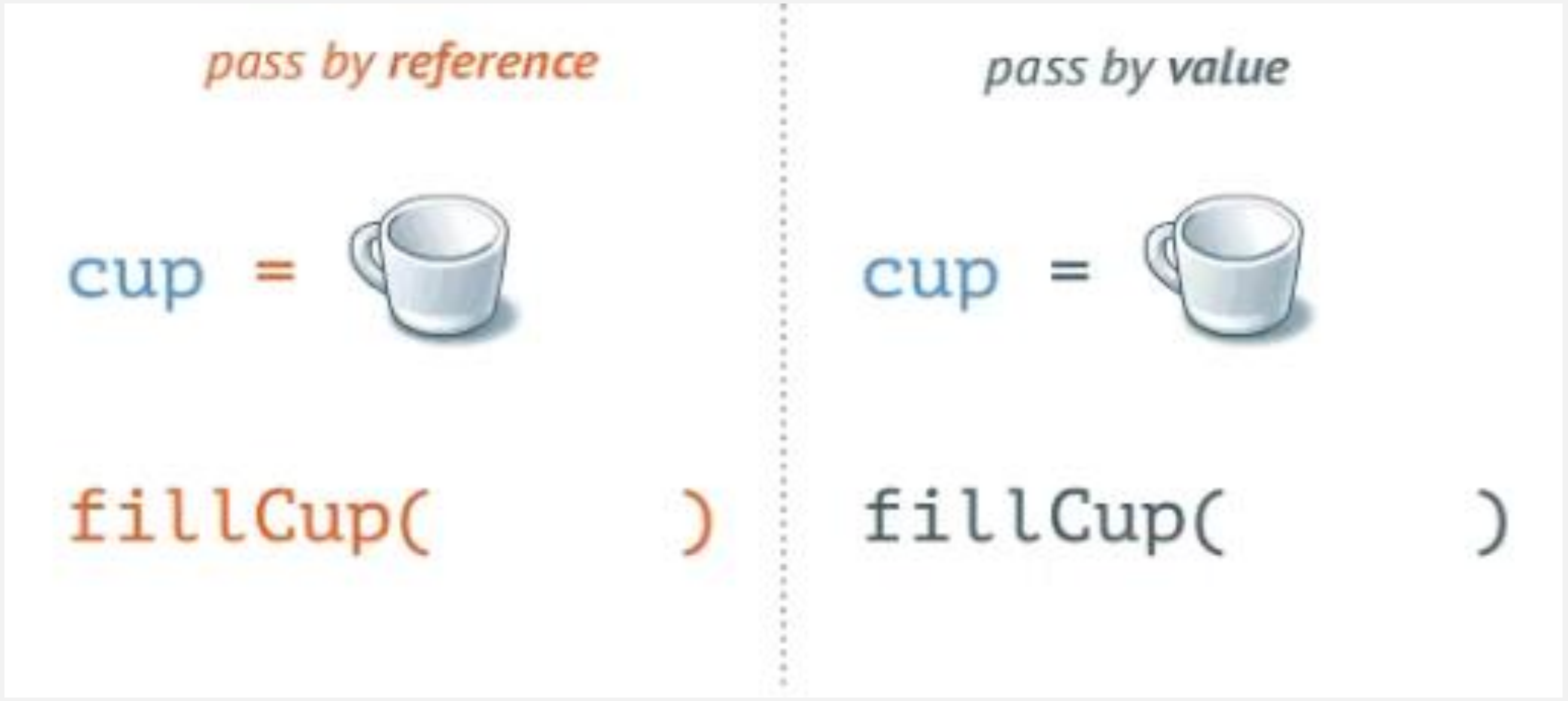
```
C:\Users\mahmad5\.jdks\liberica-1
Value of a and b -> 21 | 31
Addition = 52
Value of a and b -> 20 | 30

Process finished with exit code 0
```

# Passing the argument

It is the process of sharing address of actual parameters with the formal parameters.

It is a process of passing the copy of the actual parameters to the formal parameters.



# Pure vs Impure Functions

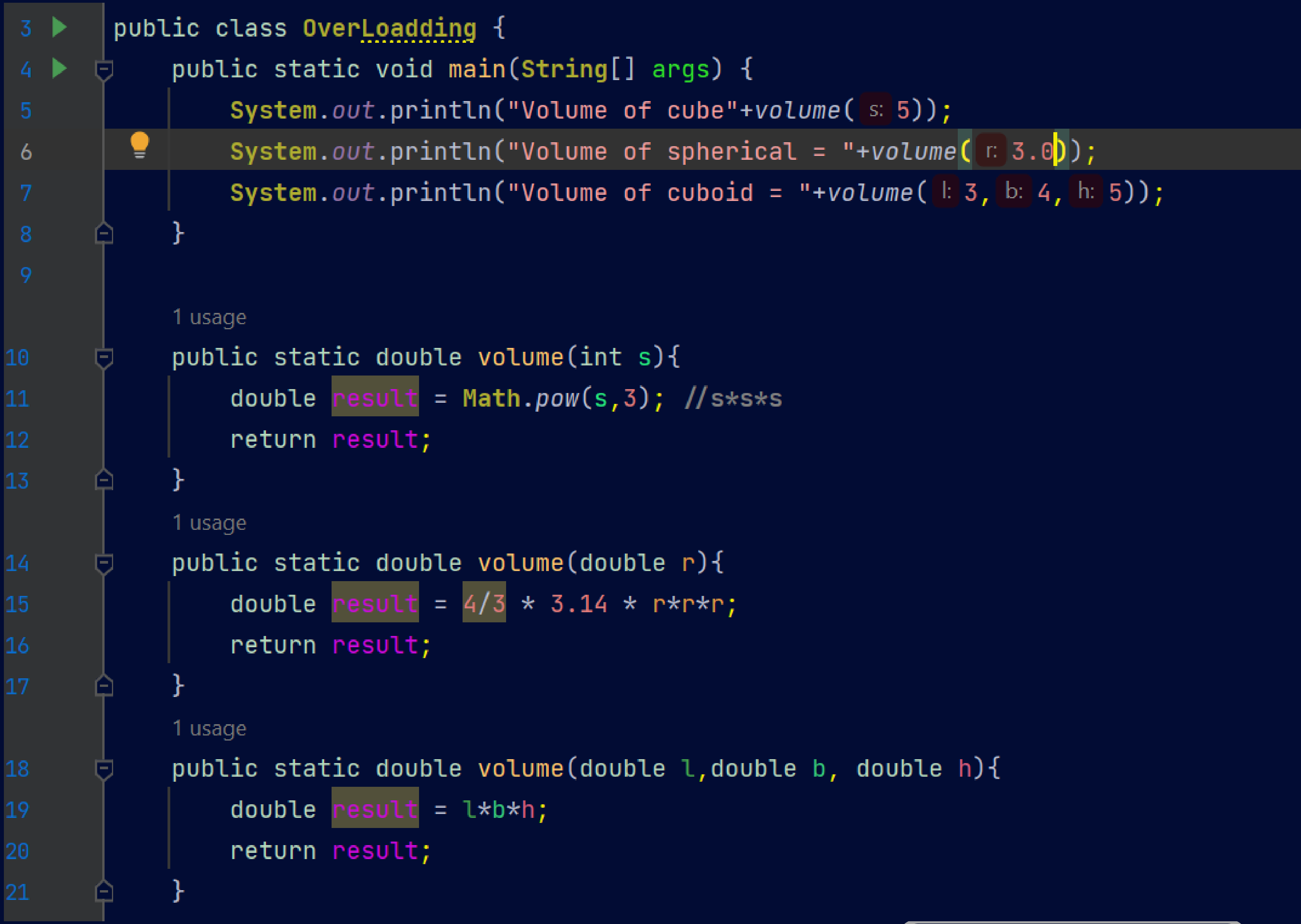
A function which **return** a value to its caller and **do not change the state** of object is called **Pure Function**

A function which **may or may not return** a value but **change the state of object** is called **Impure Function**

## Function Overloading

Function overloading is the process of defining functions/methods with the same function name but with different **number of parameter** and **type of parameter**.

# Code - Overloading



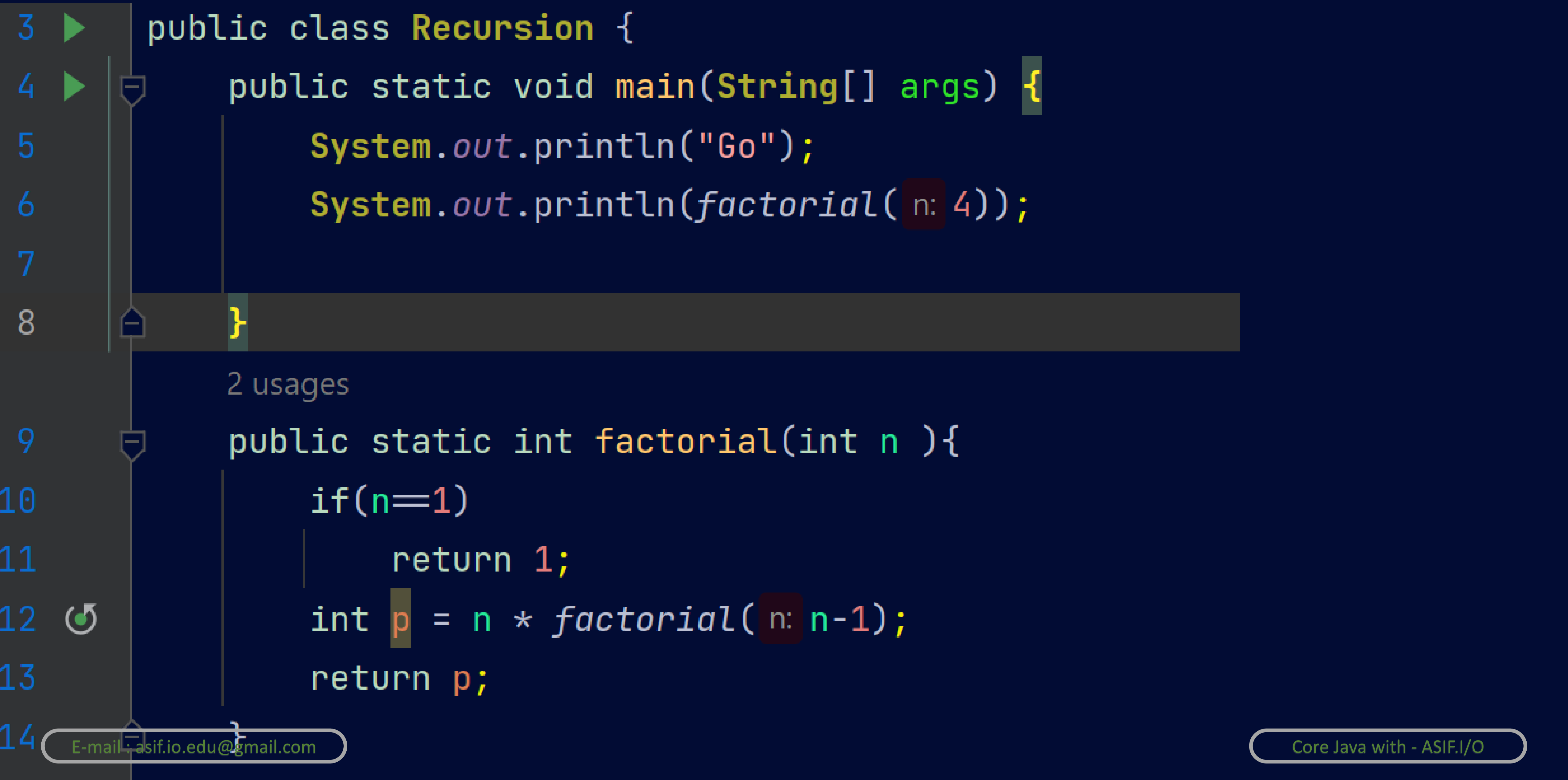
```java
public class OverLoadding {
    public static void main(String[] args) {
        System.out.println("Volume of cube"+volume( s: 5));
        System.out.println("Volume of spherical = "+volume( r: 3.0));
        System.out.println("Volume of cuboid = "+volume( l: 3, b: 4, h: 5));
    }

    1 usage
    public static double volume(int s){
        double result = Math.pow(s,3); //s*s*s
        return result;
    }
    1 usage
    public static double volume(double r){
        double result = 4/3 * 3.14 * r*r*r;
        return result;
    }
    1 usage
    public static double volume(double l,double b, double h){
        double result = l*b*h;
        return result;
    }
```

## Recursion

The process in which a function calls itself directly or indirectly is called **recursion**.

# Recursion Code



```
public class Recursion {
    public static void main(String[] args) {
        System.out.println("Go");
        System.out.println(factorial( n: 4));

    }
    2 usages
    public static int factorial(int n ){
        if(n==1)
            return 1;
        int p = n * factorial( n: n-1);
        return p;
    }
```