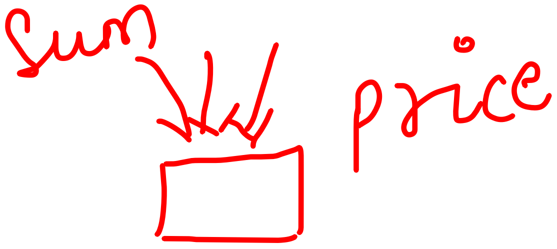
Functions → a piece of code which can be used in program whenever required.

add sum

Drawback. $\underbrace{(n!)/(n-r)}$ → n!, (n-r)!



✓

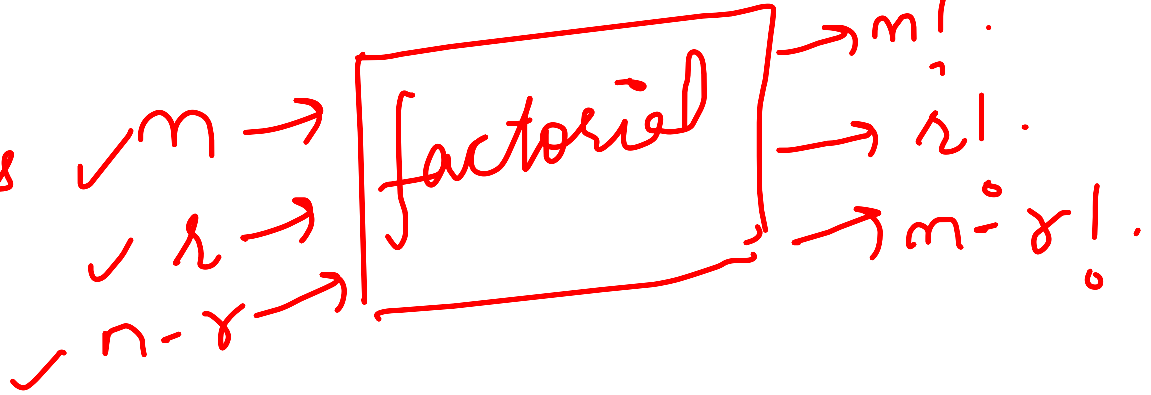
1) Repetation

2) Readability ✓

function

5 lines

20 lines w|o



Syntax of functions

o/p   i/p

```
public static return/output type func_name(parameters){

        // statement.

}
```

Calling a function

```
datatype variable name = func_name(arguments);

int ans = sum(a, b);
```

Return types of function

int → int

character → char

Double → double

Long → long

float → float

boolean → boolean

String → String

Array → int[]

Void → no o/p return

write return statement except void

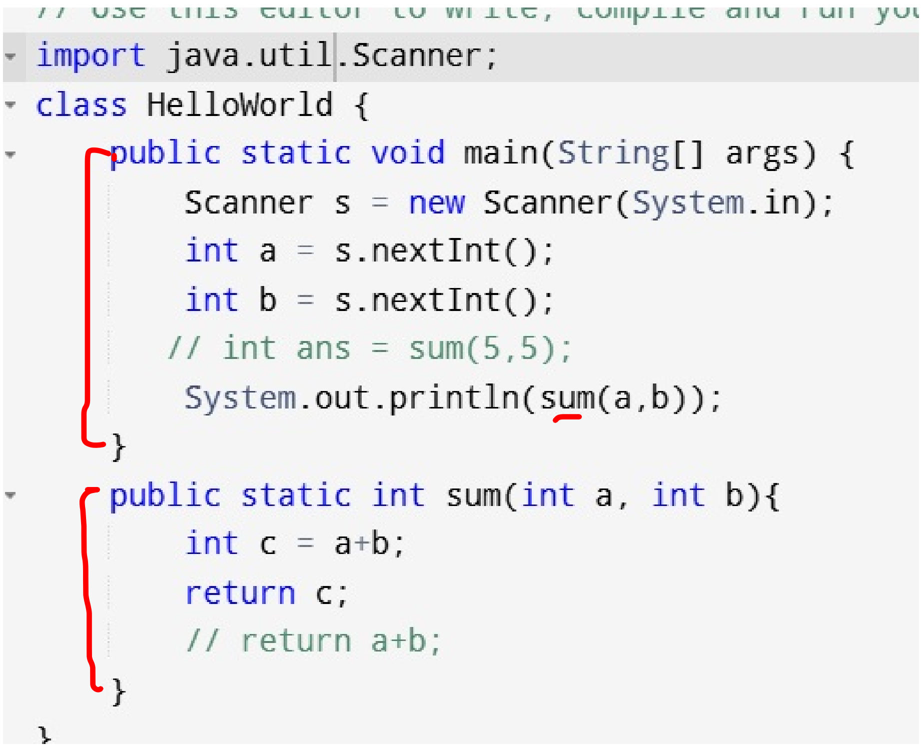
return a+b;

return → o/p print, function destroy.

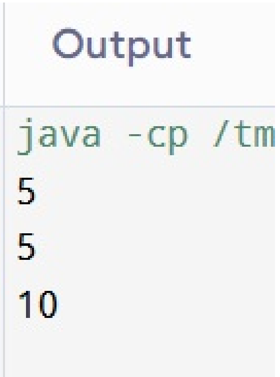
- we can have many return statements, but only one will get executed.

Parameters - defined during func. create.
defines data types as well.

Arguments - defines during function call
no data types defined.



```
// Use this editor to write, compile and run you
import java.util.Scanner;
class HelloWorld {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int a = s.nextInt();
        int b = s.nextInt();
       // int ans = sum(5,5);
        System.out.println(sum(a,b));
    }
    public static int sum(int a, int b){
        int c = a+b;
        return c;
        // return a+b;
    }
}
```



Output

```
java -cp /tm
5
5
10
```

main function → caller (main)

function which is being called is callee (sum)

Some points

→ No. of parameters and arguments matters.

→ sequence of arguments.

→ i/p can be of different data types.

→ calling a func. and collecting result of func. are different.

sum(5,5); → calling a func.

int ans = sum(5,5);
syso(ans);
} → collecting result.

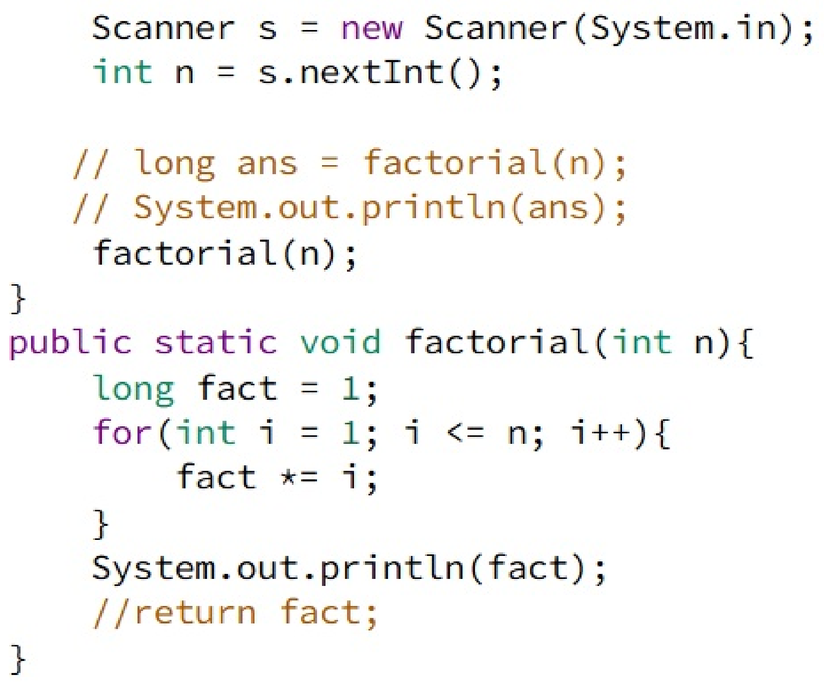
→ i/p can be empty also and the function can also be empty, then write void instead of int.

→ can't collect function if it is void function.

Function Overloading

Depends on no. of types of arguments, with same fun. name, calling it many times, called function overloading.

Factorial of N



```
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();

       // long ans = factorial(n);
       // System.out.println(ans);
        factorial(n);
    }
    public static void factorial(int n){
        long fact = 1;
        for(int i = 1; i <= n; i++){
            fact *= i;
        }
        System.out.println(fact);
        //return fact;
    }
```

Find ^nC_r

formula $\frac{n!}{(n-r)!\, r!}$ ⇒ → n! → r! → (n-r)!

$^3C_2$

$n=2$

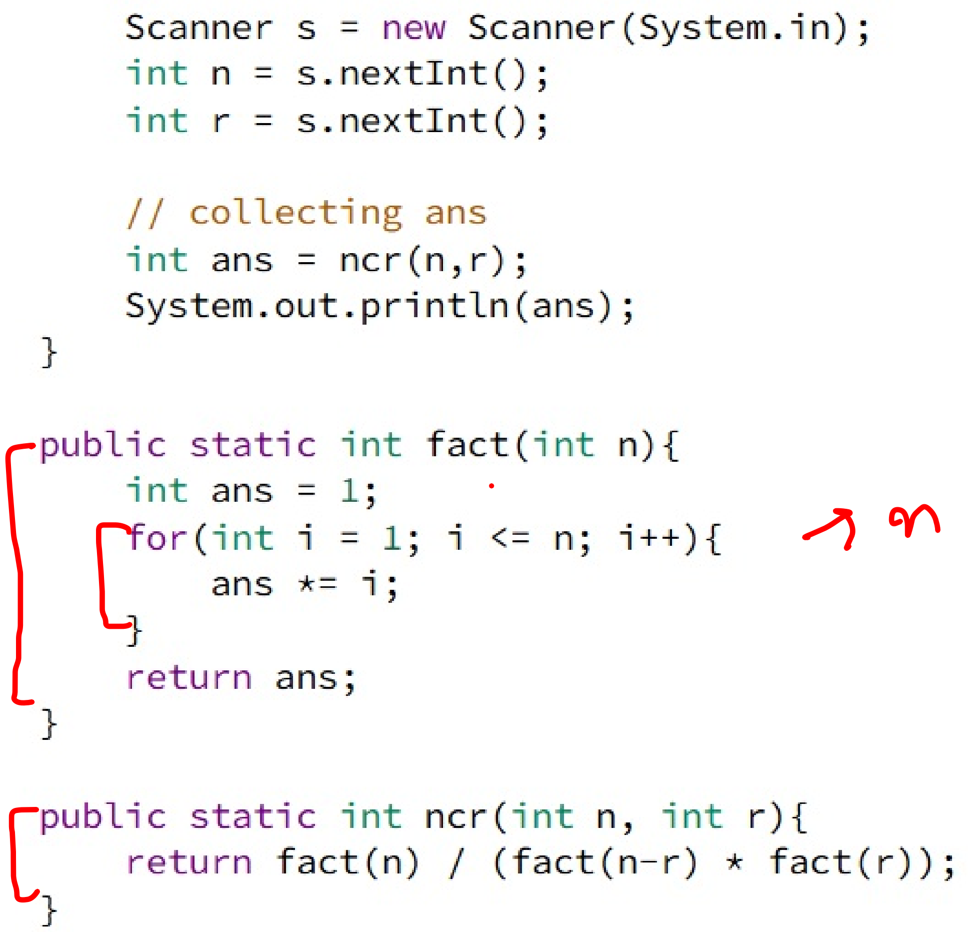
$r=4$

$\frac{2!}{4! \times (-2)!}$

$n=-2$

$i=1;\ \le -2$

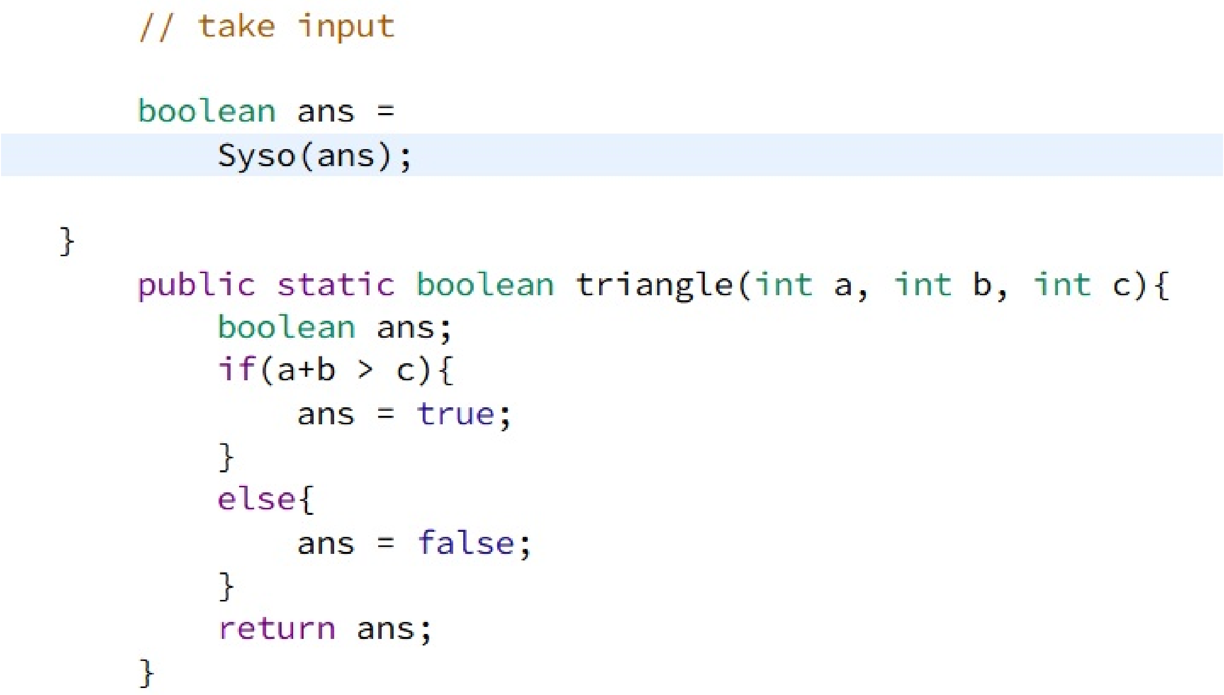
$$\Rightarrow \frac{2\times 1}{4\times 3\times 2\times 1\times -2\times -1\times 0\times 1} = \frac{2}{0} = 0$$



```
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        int r = s.nextInt();

        // collecting ans
        int ans = ncr(n,r);
        System.out.println(ans);
    }

    public static int fact(int n){
        int ans = 1;
        for(int i = 1; i <= n; i++){
            ans *= i;
        }
        return ans;
    }

    public static int ncr(int n, int r){
        return fact(n) / (fact(n-r) * fact(r));
    }
```

→ n

HW_If triangle is possible.



```
    // take input

    boolean ans =
        Syso(ans);

}
    public static boolean triangle(int a, int b, int c){
        boolean ans;
        if(a+b > c){
            ans = true;
        }
        else{
            ans = false;
        }
        return ans;
    }
```