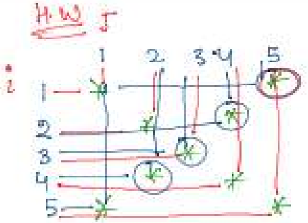
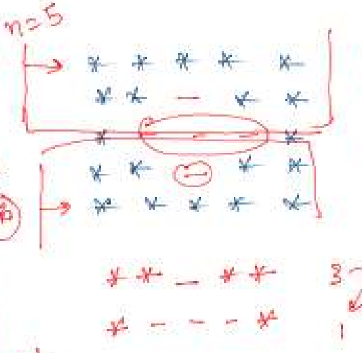
H.W



(i+j==n+1)

```
if((i==j) || (i+j==n+1){
  Sypo("* ");
}
else{
  Sypo("  ");
}
```



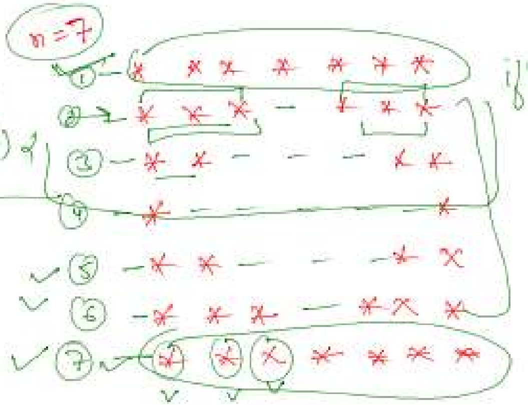
upper ①

lower

```
if(row<n/2){ ②
  nst--;
  nsp+=2;
}else{
  nst++;
  nsp-=2
}
```

```
nst = n/2 => 2 1
nsp = ~~1~~ 3

for(row 1 to n){
  [ stars)
  space()
  [ stars();
  nsp--
  ns ~~~~ +2;
}
```



```
row
if(i==1 ||
   row==n){
  sth
}
```

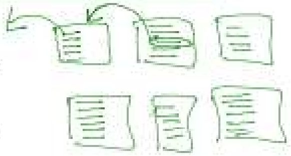
```
if(row<=n/2+1){
}else{
}
```

nst
nsp

```
for(row ---- ){
  [ col
    loop   ---- nsp   ){     * * * * * * *
           }
  [ col
    loops  for (nst){
           }
}
```

functions →

many functions



```
public Static void main(String[] Args){

}
```

public static void function nam(parameter){

access modifier

}

Class Bird{

variable ← int weight;
String color;

function ← p.s.v. fly(){
}

}

eagle
eagle.fly()

dodo
dodo.

Class String{

public ✓
private; ✓
default; ✓
protected;

```
— p. s. char charAt(int i){
}
— p. s. int length( ){
}
```

str.length();

String str = new String("abcd");

str.length()
str.charAt(i);

}

}

① with return

public Static int sum( ){

char
String

int a, int a

int ans = a+b;
return ans;

}

parameter
arguments.

Static variable

static int a=0

non static variable

int a = 0;

② without return type

public static void sum(int a, int b){

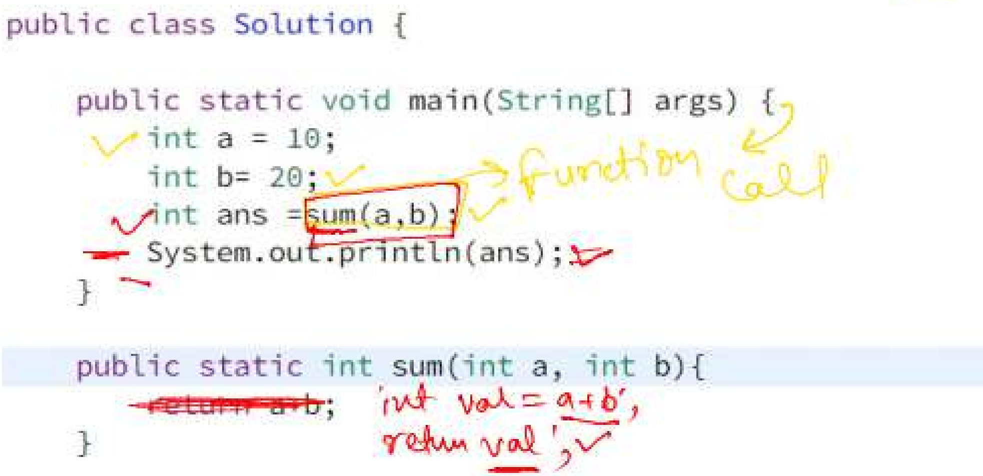
return type

Syso(a+b);

}

[whatever function on top of stack it will execute first.]

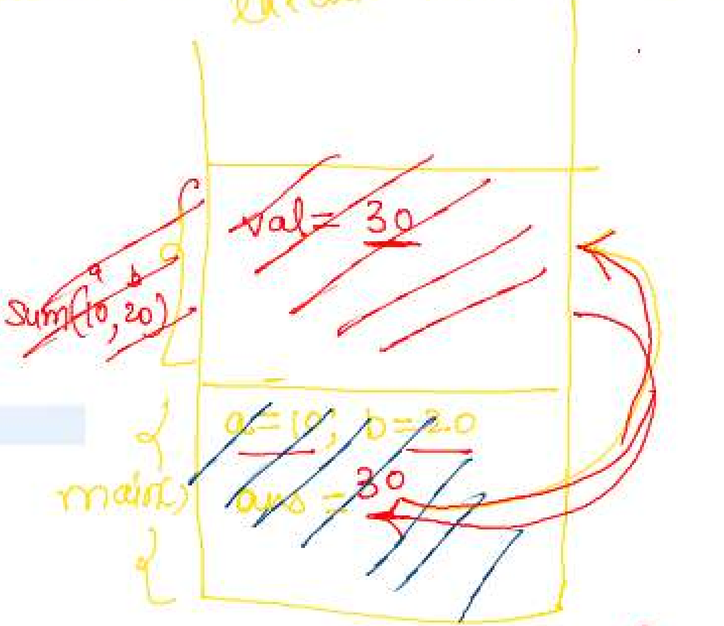
[* whatever on top of stack it will execute first.]



```
public class Solution {

    public static void main(String[] args) {
        int a = 10;
        int b= 20;
        int ans = sum(a,b);
        System.out.println(ans);
    }

    public static int sum(int a, int b){
        ~~return a+b~~;
    }
```

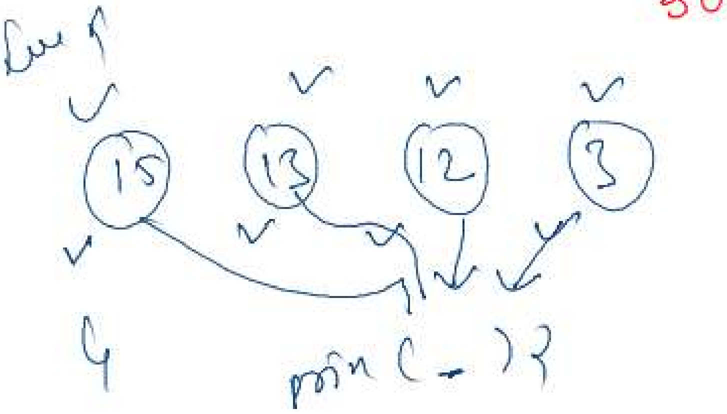


30

Two ways to wipe/remove function from stack

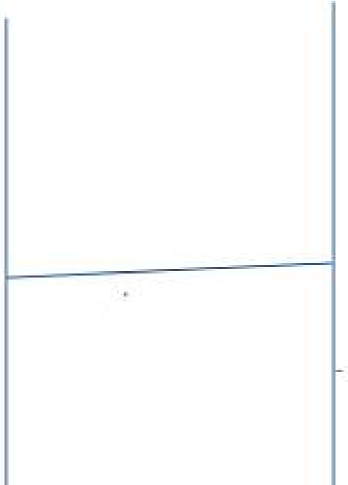
(1) return keyword

(2) when your fn code is completely executed



```
P.S.V. mai( Stri.[] ar a){
  int a = 10;
  int b' = 20;
  sum(a, b);
}
p. s. void sum( int a, int b){
  int val = a+b';
  Syso(val);
}
```

30



Q

n = 6!

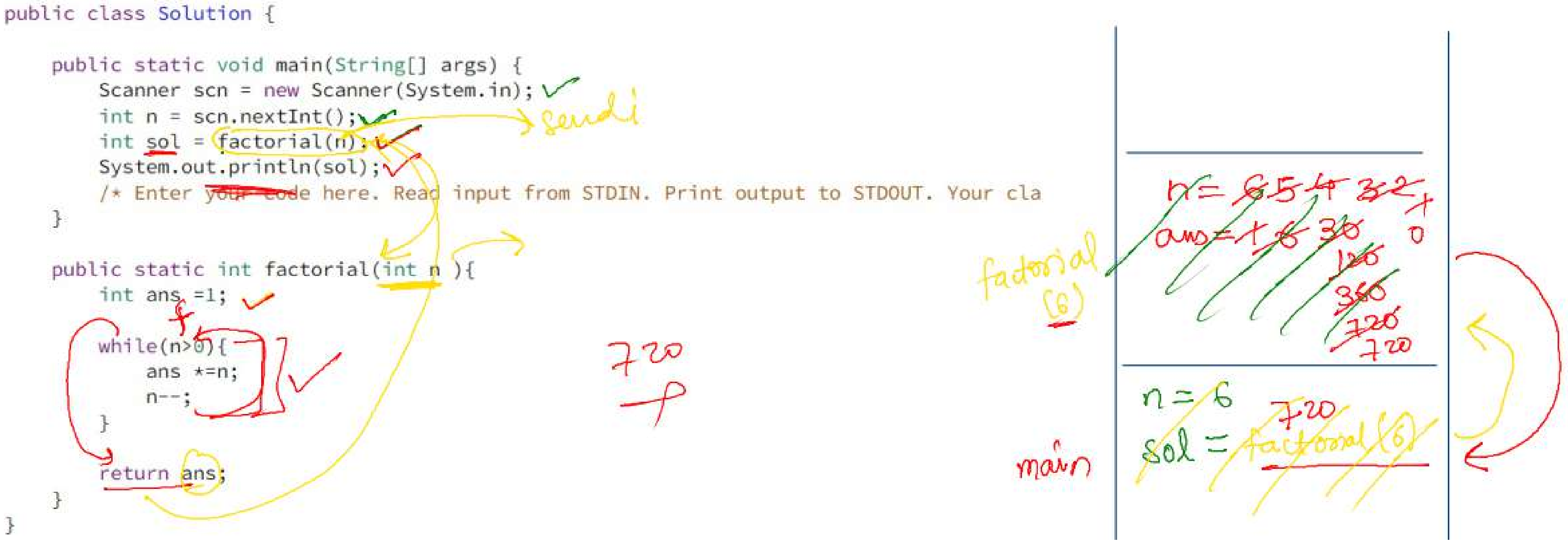
↳ 6 * 5 * 4 * 3 * 2 * 1

n! → n * n-1 * (n-2) * ----- 1;

n=7 → 7×6×5×4×3×2*1

n=8 → 8×7×6×5×4×3×2*1

n = 3 = 3×2*1



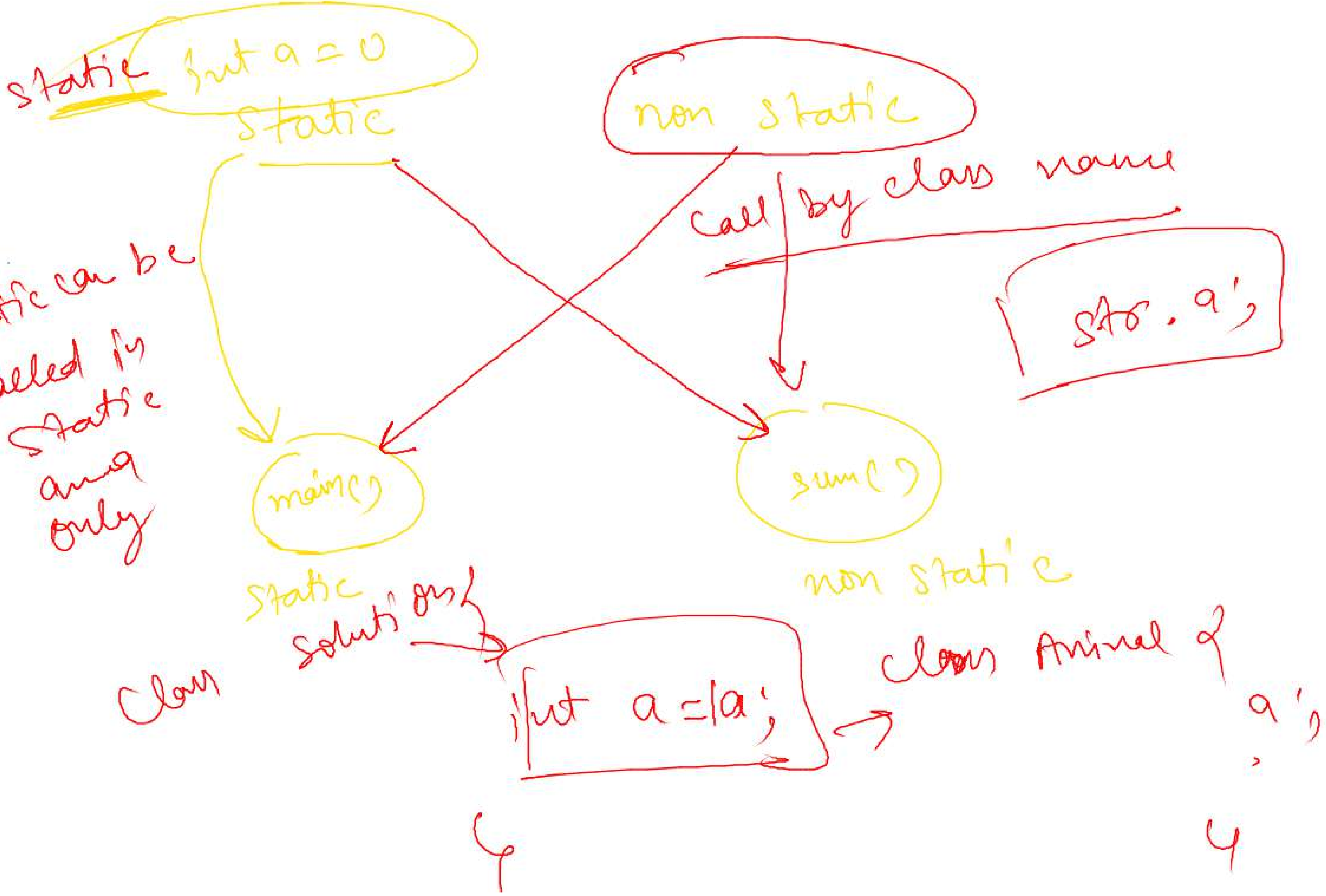
```
public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int sol = factorial(n);
        System.out.println(sol);
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your cla
    }

    public static int factorial(int n ){
        int ans =1;

        while(n>0){
            ans *=n;
            n--;
        }

        return ans;
    }
}
```



```
static int a = 0
Class solutions{
int a=10;
}
```

```
class Animal {
a;
}
```