Recursions + JAVA

- from Apna College

+ Prerequisites +

1 - Iteration

2 - functions

Basic concept of Recursion - from math

Say $f(x) = x^2$ - given

$f(f(x)) = f(x^2)$ -①

∴ Recursion + A function that calls itself

for above function if $\boxed{x=2}$

$f(x) = 2^2 = 4$

& $f(fx) = f(4) = 4^2 = 16$

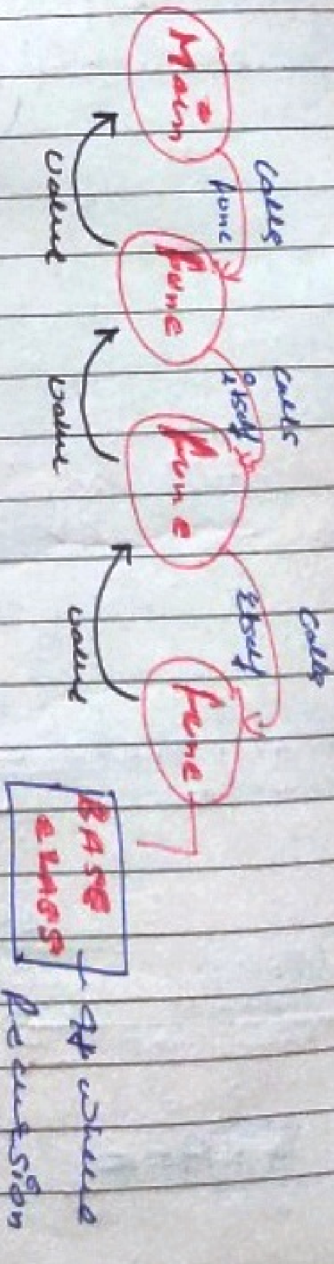
* outer function takes only one step and depends all on the inner function for all the further calculations

{working}

① outer func - takes 1 step

② ~~itera~~ and assume : Inner func will do all the work

Diagrams + of working



How to Approach the Recursion problems

1. Given -

2. what you want to do

{ forming or making the Recursion condition } Zero work

3. Base Condition

Condition which ends the Recursion

[Recursion]

- print numbers from 5 to 1 - using Recursion

Code

```
// writing a method

public static void printNum
                  (int n) {
  if(n==0) {return;}
else{Syso(n);
   printNum(n-1);}

}
```

Parameter

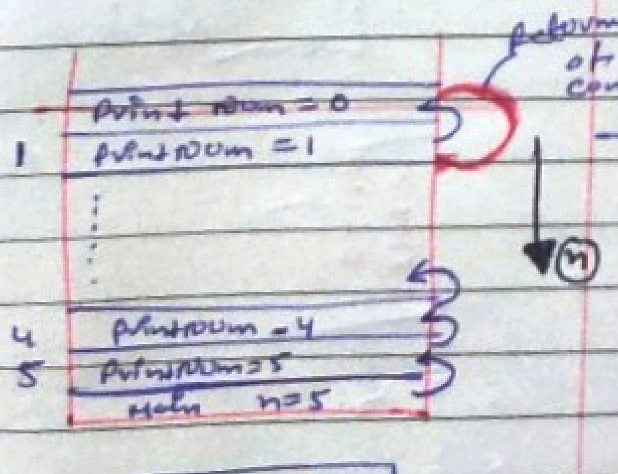
i

BASE CASE

Recursion Condition

IMP. POINT

WHAT HAPPENS IN MEMORY → In Recursion, because each print is called again it will ~~make~~ a new occupie memory space.



→ (In our case n is created at each level - which has occupied a new memory location)

STACK

- NO BASE CLASS
  ||
  STACKOVERFLOW

whole space (Memory) will get filled

Question - Print the sum of n natural numbers.

CODE - if |n = 10|

```
public static void sumnum(int n)
{   if(n==0)
    { Return; }

    int sum = 0;
    sum = sum + n
    ~~print~~ sumnum (n-1);
}

p. s. v main (String[], arg)
{
   Scanner input = new Scanner(sys.in)
   syso ("Enter nth term")
```

WRONG

```
Syso ( sumnum(n-1));
}|
}
```

- Correct approach starts with Right questions :-

CODE - if n=5

```
Public static void sumnum
            (int n, sum)
{
    if (n==0)
    {
     sysoPrint ( sum)
       return;
    }
    sum = sum+n;
    sumnum ( n-1, sum);
}

p. s. v main (String[] arg)
{   sumnum (5,0)

}
```

output
= 15 = for n=5

FACTORIAL of A NUMBER

facto of a number eg

$= n * (n-1)! --- * 1$

i.e. $2! = \boxed{2*1}$

$3! = \boxed{3*2*1}$

$\left\{ \begin{array}{l} 1! = 1 \\ 0! = 1 \end{array} \right\}$

- write problem on paper before writing the code

Code

```
static int facto(int n) -we know in this prob
{
    if (n==0 || n==1)
    {
        return 1;
    }
    n = n * facto(n-1)
    return n;
}
p.s.v main (String [] arg)
{
    sout print(facto(5));
}
}
```

n = ~~5*4 = 20~~ 20*3 = 60

~~5*3 = 15~~

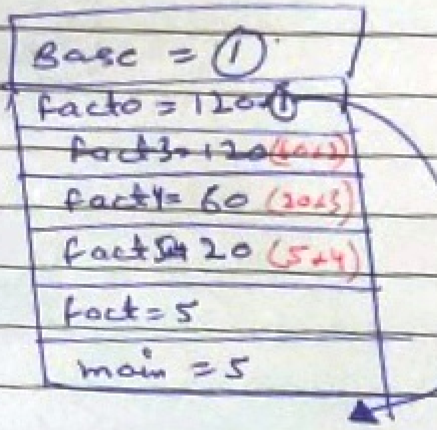
120

~~5*2 = 10~~

~~5*1 = 5~~

$\boxed{120*1}$ 1

Stack



Base = 1

facto = 120*1

fact3 = 120 (4*3)

fact4 = 60 (20*3)

fact5 = 20 (5*4)

fact = 5

main = 5

FIBONACCI SERIES

0, 1, 1, 2, 3, 5 ....

a b c

c = a+b

Lets break this problem

- first we will predefine

a = 0 ← 1st term

b = 1 ↑ 2nd term

$\overset{0}{a}, \overset{1}{b}, 1$ — a, a

(2) Step 2 — third term

a^4

c = a+b } third term

(3) Step 3 ~~a = a+b~~ = 1

~~b = b+a~~ = 1

now further will be printed

a	b	c
↓	↓	↓
0	1	1

we'll replace

a → b

b → c

→ And we'll put this condition in the **Recursion function**

Logic for x^n

it can be $x \cdot (x^{n-1})$

i.e $x^{n-1+1} = x^n$

(1) we calculate (n-1)

(2) we multiply the base i.e a with (b-1) → using recursion

(2) a · a^3 (3)

Question → print x^n — using recursion and also draw the stack diagram of memory usage

Sol Code

```
public static int Xpower
  (int a, int b)

{    if (b==0)
       { return 1; }
     if (a==0)
       { return 0 }


     int Xpown1 = Xpower(a,b-1)
     int Xpown = a*Xpown1;
     return Xpown;
}

p.s.v.m (String [], arg)

{  Scanner input = new Scanner
                     (System.in)
   Sout ("Enter a base")
   int base = . .
   Sout ("Enter a power")
   int power =
   Sout ( Xpower (base, power)
   }
```