Data Structures and Algorithms(CSE2001)

MODULE 2

RECURSION

Recursion

What is a Recursion ?

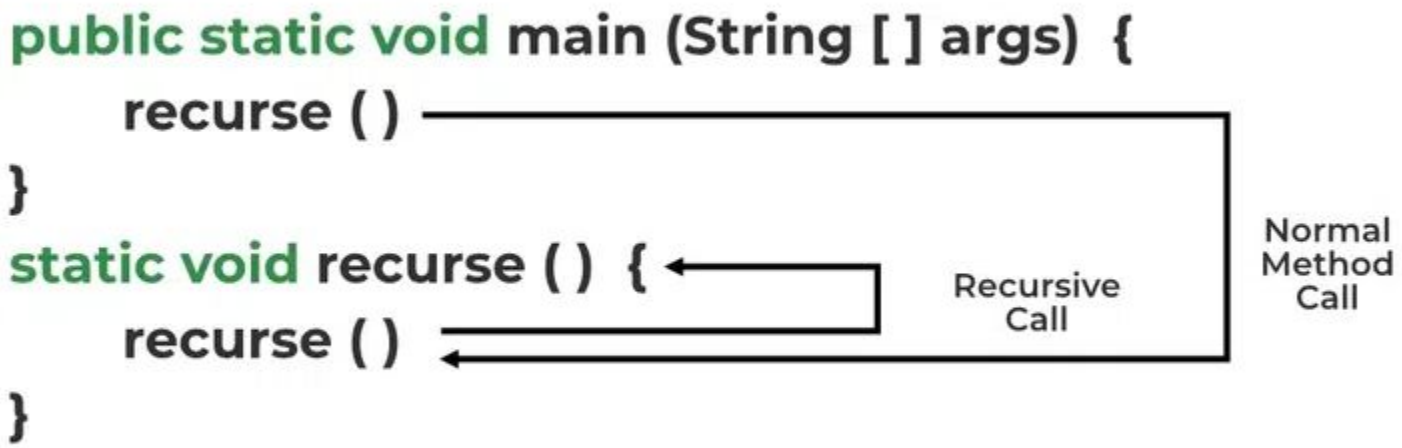
- Recursion in java is a process in which a method calls itself continuously. A method in java that calls itself is called recursive method.
- It makes the code compact but complex to understand.

Syntax:

```
returntype methodname(){
//code to be executed
methodname();//calling same method
}
```

Working of Recursion

- The idea is to represent a problem in terms of one or more smaller sub-problems and add base conditions that stop the recursion. For example, we compute factorial n if we know the factorial of (n-1). The base case for factorial would be n = 0. We return 1 when n = 0.



Recursion

How is memory allocated to different function calls in recursion?

- When any function is called from main(), the memory is allocated to it on the stack. A recursive function calls itself, the memory for the called function is allocated on top of memory allocated to the calling function and a different copy of local variables is created for each function call.
- When the base case is reached, the function returns its value to the function by whom it is called and memory is de-allocated and the process continues.

Examples

- **Java Recursion Example 1: Infinite times**

```
public class RecursionExample1 {
static void p(){
System.out.println("hello");
p();
}

public static void main(String[] args) {
p();
}
}
```

Output:

```
hello
hello
...
java.lang.StackOverflowError
```

Examples

❖ **Java Recursion Example 2: Finite times**

```
public class RecursionExample2 {
static int count=0;
static void p(){
count++;
if(count<=5){
System.out.println("hello "+count);
p();
}
}
public static void main(String[] args) {
p();
}
}
```

Output:

```
hello 1
hello 2
hello 3
hello 4
hello 5
```

Recursion using Stack

- **Java Recursion Example 3: Factorial Number**

```
public class RecursionExample3 {
static int factorial(int n){
        if (n == 1)
          return 1;
        else
          return(n * factorial(n-1));
    }
public static void main(String[] args) {
System.out.println("Factorial of 5 is: "+factorial(5));
}
}
```

Output:

Factorial of 5 is: 120

Recursion using Stack

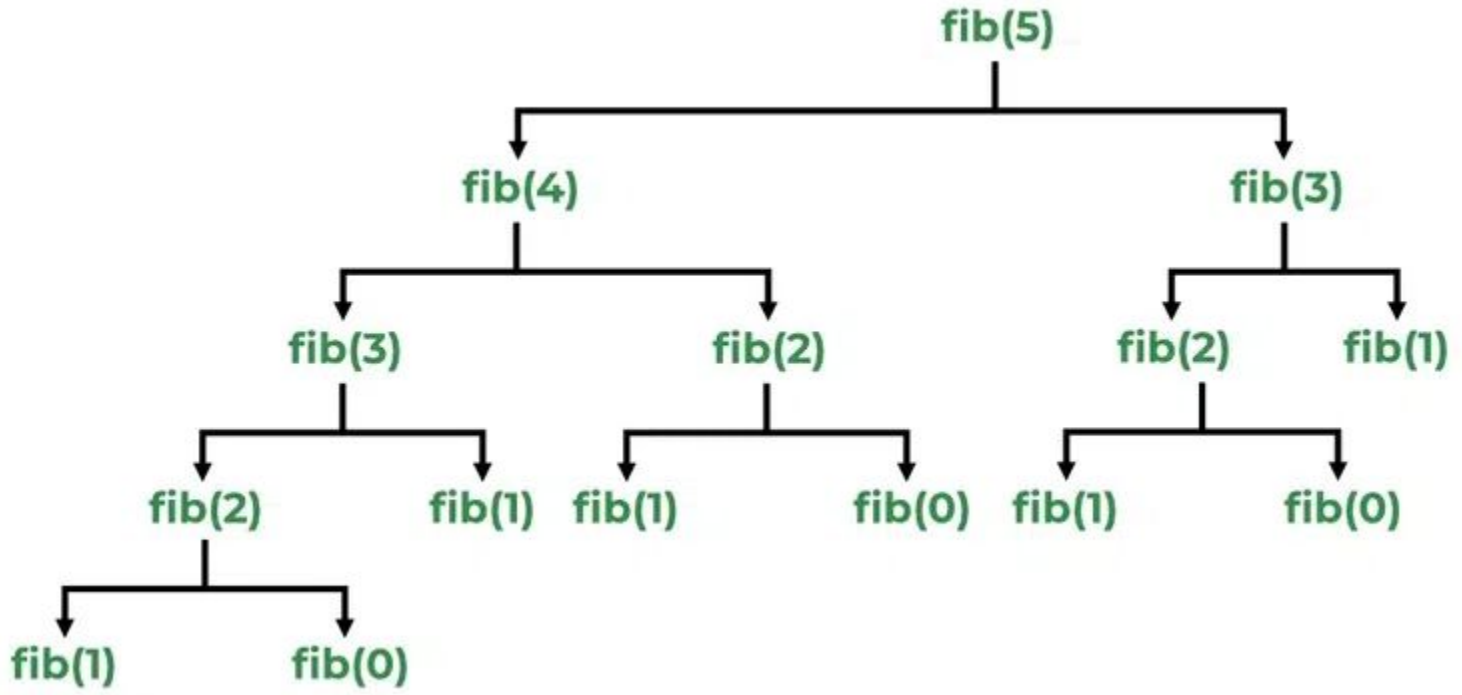
- **Working of above program:**

factorial(5)
  factorial(4)
    factorial(3)
      factorial(2)
        factorial(1)
          return 1
        return 2*1 = 2
      return 3*2 = 6
    return 4*6 = 24
  return 5*24 = 120

Recursion using Stack

❖ Java Recursion Example 4: Fibonacci Series

- Fibonacci Numbers are the numbers is the integer sequence where Fib(N) = Fib(N-2) + Fib(N-1). Below is the example to find 3,4,5.
- Fib(3) = Fib(2) + Fib(1) = Fib(1) + 0 + 1 = 1+1 = 2
- Fib(4) = Fib(3) + Fib(2) = 2+1 = 3
- Fib(5) = Fib(4) + Fib(3) = 3 + 2 = 5



Recursion using Stack

- **Java Recursion Example 4: Fibonacci Series**

```
public class RecursionExample4 {
    static int n1=0,n2=1,n3=0;
     static void printFibo(int count){
        if(count>0){
            n3 = n1 + n2;
            n1 = n2;
            n2 = n3;
            System.out.print(" "+n3);
            printFibo(count-1);
        }
    }
```

Recursion using Stack

❖**Java Recursion Example 4: Fibonacci Series (Contd.,)**

```
public static void main(String[] args) {
   int count=15;
    System.out.print(n1+" "+n2);//printing 0 and 1
    printFibo(count-2);//n-2 because 2 numbers are already printed
}
}
```

Output:

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

❖ Advantages of Recursive Programming

The advantages of recursive programs are as follows:

- Recursion provides a clean and simple way to write code.
- Some problems are inherently recursive like tree traversals, Tower of Hanoi, etc. For such problems, it is preferred to write recursive code.

Disadvantages of Recursive Programming

The disadvantages of recursive programs is as follows:

- The recursive program has greater space requirements than the iterative program as all functions will remain in the stack until the base case is reached.
- It also has greater time requirements because of function calls and returns overhead.