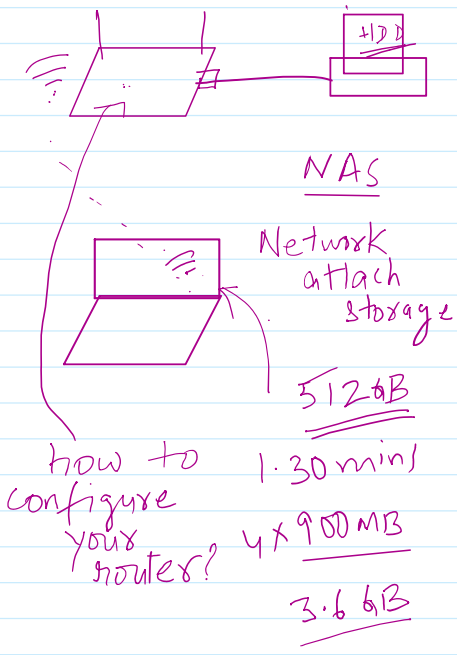
Recursion

28 October 2025 08:41

What is Recursion in Java?

Recursion is a programming technique where a method (function) calls itself in order to solve a problem. Instead of using loops (for, while) We sometimes solve problems using recursion. *Especially when the problem can be broken down into smaller sub problems of the same kind.*



Recursive methods has two parts:

1. **Base Case(Stopping Condition):** A condition that stops the recursion to prevent infinite calls.
2. **Recursive Case:** The part where the method calls itself with a smaller or simpler input.

Think like this: Solving a big problem by breaking it into smaller version of the same problem until you reach the simplest form (base case).

Simple example: Recursive function to print numbers from n to one.

```
package BasicRecursion;
public class PrintNumbersInDescending {
   public static void printDescending(int n){
      if(n == 1){ //Base case
         System.out.print(n);
      }
      else{
         System.out.print(n+", ");
         printDescending(n-1); //recursive call
      }
   }
}
```

Output:

```
jshell> import BasicRecursion.PrintNumbersInDescending;

jshell> PrintNumbersInDescending.printDescending(5);
5, 4, 3, 2, 1
```

```
package BasicRecursion;
/*
 * 5!=5*4*3*2*1 = 120
 */
public class Factorial {
   public static int getFactorial(int n){
      //n!= n * (n-1) * (n-2)*...*(n-k)  The condition is n > k
      // (n-k) = 1, then we stop or this is the base case.
      // n = 0, then 0! = 1 another base case
      //we need to think about base case or exit case
      if( n == 0 || n == 1){
         if(n==1){
            System.out.print(n);
         }
```

```
            return 1;
        }
        else{
            System.out.print(n+"*");
            return n * getFactorial(n-1);
        }
    }
    public static void main(String[] args) {
        System.out.print("\n 5! = ");
        System.out.print( " = " + getFactorial(5));
        System.out.print("\n 6! = ");
        System.out.println( " = "+getFactorial(6));
    }
}
```

Output:

```
5! = 5*4*3*2*1 = 120
6! = 6*5*4*3*2*1 = 720
```

For that, recursive methods must have a base case or they will lead to infinite recursion and eventually a stack overflow error.
Recursion can be elegant, but may use more memory than loops.

Why to use recursion?

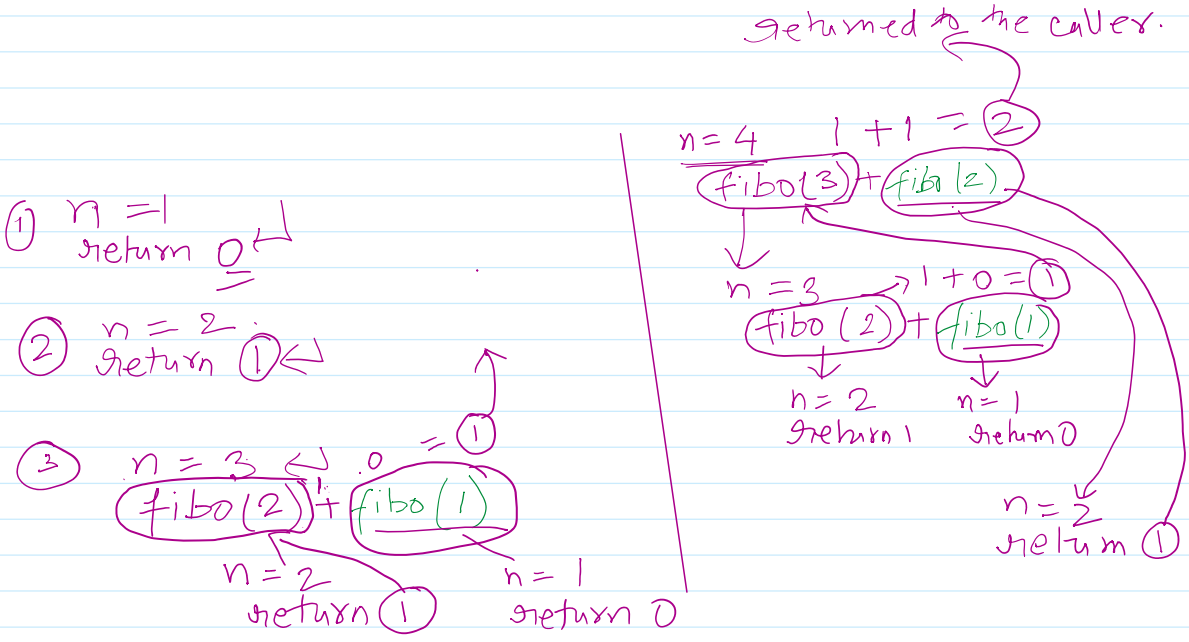
- **Problems with repetitive patterns (example factorial, Fibonacci).**
- **Divide and conquer problems. (examples are searching, sorting, etc.)**
- **Tree or graph traversals.**

Q1. WAP to print n Fibonacci numbers using recursion. The series is created by adding two previous term to get the current term.
Input: 5
Output: 0, 1, 1, 2, 3

Input: 6
Output: 0, 1, 1, 2, 3, 5

Input: 7
Output: 0, 1, 1, 2, 3, 5, 8

Input: 8
Output: 0, 1, 1, 2, 3, 5, 8, 13




```
int fiboTerm(int n){
    if(n == 1){
        return 0;
    }
    else if(n == 2){
        return 1;
    }
    else{
        return fibo(n-1) + fibo(n-2);
    }
```

```java
package BasicRecursion;
public class FibonacciSeries {
   public static int fibo(int n){
      if(n == 1){ //base case
         return 0;
      }
      else if(n == 2){//base case
         return 1;
      }
      else{ //recursive case
         return fibo(n-1) + fibo(n-2);
      }
   }
   public static void printFiboTerms(int howmany, int start){
      if(start<howmany){          //base case
      System.out.print(fibo(start++)+", ");
      printFiboTerms(howmany, start);//recursion
      }
      else{
      System.out.print(fibo(start));
      }
   }
}
```

due statements are present, so memory is involved.

updated value of start

there is no due statement, so this code is equivalent to any looping code.

Output:

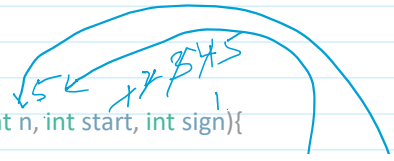
```
jshell> FibonacciSeries.printFiboTerms(10, 1);
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
jshell> import BasicRecursion.FibonacciSeries;

jshell> FibonacciSeries.printFiboTerms(20, 1);
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181
```

```java
package BasicRecursion;
public class Q1e {
   public static void printSeries(int n, int start, int v){
      if(start < n ){
         v = 10*v + start;
         System.out.print(v+", ");
         printSeries(n, start+1, v);
      }
      else{
         v = 10*v + start;
         System.out.print(v);
      }
   }
}
```

Output:

```
jshell> Q1e.printSeries(5,1, 0);
1, 12, 123, 1234, 12345
```

```java
package BasicRecursion;
import java.util.Scanner;
public class Q1a {
   public void printSeries(int n, int start, int sign){
      if(start < n){
```



output:
1, −3, 5, −7, 9.

```
import java.util.Scanner;
public class Q1a {
   public void printSeries(int n, int start, int sign){
      if(start < n){
         System.out.print((start*2-1)*sign + ", ");
         printSeries(n, start+1, sign*(-1));
      }
      else{
         System.out.println((start*2-1)*sign);
      }
   }
   public static void main(String[] args) {
      int n;
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter number of terms: ");
      n = sc.nextInt();
      new Q1a().printSeries(n, 1, 1);
      sc.close();
   }
}
```

output:
1, −3, 5, −7, 9

0, 1, 2, 3, 6

a = 0 b = 1 c = 2 3

v = a+b+c = 1, 2, 3, 6

a = b

b = c

c = v

a = 0, b = 0, c = 1

0, 1, 2, 3, 6

a = 0 0 1 b = 0 1 2 c = 1 2 3 v = 1 2 3 6

if(base condition){

print v + ", "

a = b

b = c

c = v

v = a+b+c

a = 0 0 1

b = 0 1 2

c = 1 2 3

v = 1 2 3 6

}

else{

print the last value ..

}

0, 1, 2, 3, 6

0, 1, 2,

```
package BasicRecursion;
import java.util.Scanner;
public class Q1c {
```

```
    public void printSeries(int n, int a, int b, int c, int v){
        if(n > 1){
            System.out.print(v+", ");
            a = b;
            b = c;
            c = v;
            v = a + b + c;
            printSeries(n-1, a, b, c, v);
        }
        else{
            //print the last value
            System.out.print(v);
        }
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter number of terms: ");
        int n = sc.nextInt();
        new Q1c().printSeries(n, 0, 0, 1, 0);
    }
}
```