Recursion in Java

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:**

1. returntype methodname(){
2. //code to be executed
3. methodname();//calling same method
4. }

Java Recursion Example 1: Infinite times

1. **public** **class** RecursionExample1 {
2. **static** **void** p(){
3. System.out.println("hello");
4. p();
5. }
7. **public** **static** **void** main(String[] args) {
8. p();
9. }
10. }

Output:

hello

hello

...

java.lang.StackOverflowError

Java Recursion Example 2: Finite times

1. **public** **class** RecursionExample2 {
2. **static** **int** count=0;
3. **static** **void** p(){
4. count++;
5. **if**(count<=5){
6. System.out.println("hello "+count);
7. p();
8. }
9. }
10. **public** **static** **void** main(String[] args) {
11. p();
12. }
13. }

Output:

hello 1

hello 2

hello 3

hello 4

hello 5

Java Recursion Example 3: Factorial Number

1. **public** **class** RecursionExample3 {
2. **static** **int** factorial(**int** n){
3. **if** (n == 1)
4. **return** 1;
5. **else**
6. **return**(n \* factorial(n-1));
7. }
9. **public** **static** **void** main(String[] args) {
10. System.out.println("Factorial of 5 is: "+factorial(5));
11. }
12. }

Output:

Factorial of 5 is: 120

**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

Java Recursion Example 4: Fibonacci Series

1. **public** **class** RecursionExample4 {
2. **static** **int** n1=0,n2=1,n3=0;
3. **static** **void** printFibo(**int** count){
4. **if**(count>0){
5. n3 = n1 + n2;
6. n1 = n2;
7. n2 = n3;
8. System.out.print(" "+n3);
9. printFibo(count-1);
10. }
11. }
13. **public** **static** **void** main(String[] args) {
14. **int** count=15;
15. System.out.print(n1+" "+n2);//printing 0 and 1
16. printFibo(count-2);//n-2 because 2 numbers are already printed
17. }
18. }

Output:

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