Рекурсия

Рекурсия – это очень удобная технология, но она сложная для понимания.

Рекурентные методы – это методы, которые вызывают внутри своего тела себя самих.

Например, рассмотрим задачу нахождения i-го числа Фибоначчи.

I-е число Фибоначчи является по определению суммой (i-1)-го и (i-2)-го чисел при i>2. Первые два числа Фибоначчи равны единицам.

```
int fib(int n){
    if (n == 1||n==2) return 1;
    else return fib(n-1)+fib(n-2);
}
```

Разберём более сложную задачу:

Напишите программу, с помощью рекуррентного метода, которая читает с клавиатуры целые числа, пока не будет введен 0 и выводит их в обратном порядке. Чисел гарантированно не больше 1000.

```
import java.util.Scanner;
public class Main {
    static Scanner sc = new Scanner(System.in);
    static void myMethod(int cnt,int [] arr) {
        int a = sc.nextInt();
        if (a == 0) {
            for (int i = cnt-1; i >= 0; i--) {
                System.out.print(arr[i]+" ");
            }
        } else {
            arr[cnt]=a;
            myMethod(cnt+1,arr);
        }
    }
    public static void main(String[] args) {
        int arr[] = new int[1000];
        int cnt = 0;
        myMethod(cnt,arr);
    }
}
```

В этой программе есть несколько важных моментов.

```
static Scanner sc = new Scanner(System.in);
```

Переменную sc мы выносим из тела main, чтобы она была доступна в других методах. Т.к. статичные методы могут работать только со статичными полями, то нам необходимо добавить в объявление сканера ключевое слово static.

```
static void myMethod(int cnt,int [] arr) {
```

Объявление метода выполнено с использованием ключевого слова static по тем же соображениям. В качестве аргументов этому методу мы передаём массив, в который будем записывать считанные значения и кол-во уже прочитанных значений.

Дальше логика работы метода проста: считываем новое число, если оно равно нулю, то просто выводим элементы полученного в качестве аргумента массива в обратном порядке, если нет, то добавляем новый элемент массива, увеличиваем кол-во занесённых элементов в массив и вызываем метод заново но уже с новыми параметрами.