

Direct Recursion#

Direct recursion occurs when a method calls itself.

This results in a *one-step recursive call*: the method makes a recursive call inside its own body.

```
1 class ExampleClass {
2
3     private static void f() {
4
5         // some code...
6
7     f();
8
9     //some code...
10
11 }
12
13 public static void main(String args[] ) {
14
15     // Method called here
16
17 }
18
19 }
```

The code snippet below gives an example of a direct recursive method that **computes the square of a number**.

```
1 class Square {
2
3     // Recursive method to calculate square of a number
4     private static int square(int n) {
5
6         // Base case
7         if (n == 0) {
8             return 0;
9         }
10
11        // Recursive case
12        else {
13            return square(n-1) + (2 * n) - 1;
14        }
15    }
16
17    public static void main( String args[] ) {
18        int input = 6;
19        int output = square(input);
20        System.out.println("The square of the number " + input + " is: " + output);
21    }
22 }
```

Run Save Reset

We will now briefly discuss the two main parts of a recursive method, the base case and the recursive case, implemented in the code above.

The Base Case#

We have defined the base case on **line 5** where it states that when the variable **n** equals to 0, the method should terminate and start popping frames from the stack.

The Recursive Case#

Let’s take a look at the mathematical operation required to perform n^2 . We need to decrement the value of **n** in such a way that we can use it to call the same method but not change the mathematical formula. We get this:

$(n - 1)^2$ which opens up to be $(n^2 - 2n + 1)$.

From the formula above, we get that $(n - 1)^2 = (n^2 - 2n + 1)$. The laws of math say that we can rearrange this formula in a way such that we isolate the n^2 by bringing everything to one side. We get this as a result:

$n^2 = (n - 1)^2 + 2n - 1$

That is how we get our formula to iterate recursively to get the square of a number.

Indirect Recursion#

Indirect recursion (or mutual recursion) occurs when a method calls another method, eventually resulting in the original method being called again.

For example, if method **f()** calls another method **g()** , then **g()** calls another method **h()** and **h()** eventually calls the original method **f()** . This phenomenon results in **indirect recursion** (or mutual recursion).

```
class ExampleClass {

    private static void f() {
        // some code...
        g();
        //some code...
    }

    private static void g() {
        // some code...
        h();
        //some code...
    }

    private static void h() {
        // some code...
        f();
        //some code...
    }

    public static void main(String args[] ) {

        // Method called here

    }

}
```

The code snippet below gives an implementation of indirect recursion that **prints the first 20 integers**.

```
class ExampleClass {

    static int n = 0;
    public static void indirectRecursiveFunction1() {
        if (n <= 20) {
            System.out.print(n + " ");
            n++;
            indirectRecursiveFunction2();
        }
        else {
            return;
        }
    }

    public static void indirectRecursiveFunction2() {
        if (n <= 20) {
            System.out.print(n + " ");
            n++;
            indirectRecursiveFunction1();
        }
        else {
            return;
        }
    }

    public static void main( String args[] ) {
        indirectRecursiveFunction1();
    }

}
```

Run Save Reset

Now that you have understood the concept of direct and indirect recursion, let’s move on to the next lesson and find out when we should use recursion.