Chapter 10

Recursive problem solving – Method recursion

Some data manipulation tasks can, by their nature, best be performed as performing some small manipulation task on the result of first performing the same task on simpler data. This is repeated until such a simple task results that it can be performed directly. The small manipulation tasks created in this process are then performed, in the order opposite of the one in which they were created.

This is the recursive (recurring to itself) execution of a task: a recursive algorithm. A task is called recursive by nature if a recursive algorithm is the easiest kind of algorithm to design for it.

10.1 Aim

We want to perform a task recursively (the simplification usually being in terms of the data).

10.2 Means

We write a recursive method for the task. The method body firstly contains code that describes the recursive part of the task, followed by a call to the method itself, with simpler data. This part is executed when the simplest case is not reached yet. Secondly, the method body contains code that describes how to directly perform the task for the simplest case. This code is executed when the method is called with data corresponding to the simplest case.

The data that is relevant for the recursive execution of the task, and thus has to become simpler at each recursive call, is usually provided through parameters. To enable use of the results of the calls, in building the total result, the results of the calls are usually provided through the return.

NB A recursively performed task can always also be performed using a, different, iterative algorithm. Translation often is not easy. Usually an iterative algorithm is more efficient, especially in terms of memory use, than a recursive one.

In the example below, a recursive method is used to compute the factorial of an integer number.

```java
import java.util.*;
class ComputeFactorial{
    Scanner sc = new Scanner(System.in);
    long factorial(int n){
```
if (n==0)
    return 1;
else
    return n*factorial(n-1);
}

public void demo(){
    int n;

    System.out.println("Provide a number to compute the factorial for");
    n = sc.nextInt();
    System.out.println("Factorial of " + n + " is " + factorial(n));
}

// main instantiates and uses the object
public static void main(String[] args) {
    ComputeFactorial computeFactorial = new ComputeFactorial();
    computeFactorial.demo();
}

10.3 Execution model

Each recursive call gives rise to a new instantiation of the method, until the simplest case is reached. When, starting with the simplest case, the execution of some method is completed, the result is returned to the previous method and the returning method disappears.