Learning goals

Before the next day, you should have achieved the following learning goals:

- Be able to write recursive code for simple problems

- Understand what a recursive method looks like, and how it differs from an iterative method.

Remember that **exercises marked with a star (*) are harder and take longer to solve**, so you should not try them until you have solved all the non-star ones.

1 Recursive code, line by line

a)

What number would the following method print on screen if you called `printNumbers(6)`?

```java
void printNumbers(int n) {
    if (n <= 0) {
        return;
    }
    printLine(n);
    printNumbers(n-2);
    printNumbers(n-3);
    printLine(n);
}
```

Do this exercise on paper. Then add this method to a Java Decaf program and check your answer.

b)

Do you see anything wrong with the following code? How would you fix it?

```java
String buggyMethod(int n) {
    if (n == 0) {
        return "";
    }
    return n + " " + buggyMethod(n - 2);
}
```

2 Classics

2.1 a) Factorial

Write a small program with a method that calculates the factorial of an integer number as seen in the notes.
Is it easy to do this both iteratively and recursively? Try both ways and see which is more natural for you. If one takes too long, try the other way.

2.2 b) Fibonacci

Write a small program with a method that calculates the nth element of the Fibonacci sequence as seen in the notes.
Is it easy to do this both iteratively and recursively? Try both ways and see which is more natural for you. If one takes too long, try the other way.
When doing it recursively, do it with and without memoization. Compare the time that is needed in each case to find F(40) or F(45).

2.3   c) Hanoi towers

A legend says that, somewhere in the East, near Hanoi, there is a temple. In the temple, there are three posts. In the posts, there are 64 discs of 64 different sizes. When the world was created, all the discs were in the first post; when all the discs are moved to the last post, the world will end.

The monks in the temple move the discs between posts, but they must obey two simple rules. First, only one disc can be moved at a time, from one post to another post (it cannot be left anywhere else). Last, but not least, a disc can only be placed on top of a bigger disc, never on top of a smaller disc. The smallest disc can be placed on any post (all other discs are bigger); the biggest disc can only be placed on an empty post.

Create a method that calculates the number of moves necessary to move a set of n Hanoi disks from the initial post to the last post. **Hint:** if you want to play monk yourself in order to better understand the problem, ask the lecturer for a "Hanoi envelope".

3   Paper sizes (*)

A Din-A0 sheet of paper is 841mm × 1189mm (its surface is one square meter). Successive measurements of paper are defined recursively as "half" or "double" the preceding size. Therefore, a Din-A1 sheet of paper is half of Din-A0, or 594mm × 841mm; while a Din-A00 is double of Din-A0, or 1189mm× 1682mm.

Create a method that takes a String parameter representing a size (e.g. "A4", "A00000") and prints on screen the size of the corresponding sheet of paper. For simplicity, you can ignore rounding errors when calculating smaller sizes.

Hint: note that two consecutive sizes always share one side and only the other one is multiplied or divided by two, e.g. the *short* side of Din-A2 is as long as the *long* side of Din-A3, while the long side of Din-A2 is twice as long as the short side of Din-A3.

4   Palindrome

Create a program with a (recursive) method that takes a String paremeter and returns true if the String is a palindrome and false otherwise. Compare this recursive version with the iterative version you wrote in past weeks. Which one seems clearer to you?

5   Power

Create a class with a method `pow` that takes two integers (`base` and `exponent`) and calculates the power, e.g. `pow(2,3)` calculates 2^3.

Is it easy to do this both iteratively and recursively?

6   Greatest common divisor

The greatest common divisor (GCD) of two natural numbers is the largest natural number that can exactly divide them both. For example, the GCD of 12 and 15 is 3, the GCD of 12 and 6 is 6 (because both divisons 12/6 and 6/6 are exact), and the greatest common divisor of 14 and 27 is 1.

Finding the GCD of two numbers has a lot of applications in cryptography and information security. We can find the GCD of two numbers by finding all of them and them matching the lists for both numbers, but there is a fastest way. The GCD has the property that

$$\texttt{if } p > q, \texttt{ then } gcd(p, q) = gcd(q, p \bmod q)$$

where *mod* represents the modulo operator (i.e. %). Therefore, given two numbers p and q, where $p > q$, if q is a divisor of p, then q is the GCD of both number; otherwise, we can apply the property above to find a pair of smaller numbers with the same GCD.

Write a program that finds the GCD of two natural numbers recursively using this property.

(Hint: your base case is that if q is a divisor of p, then q is the GCD of p and q.)

7 Hanoi Redux (**)

The legend of the Towers of Hanoi has inspired countless variations as mind games. Here is one of them: if we have three posts and 64 discs, where all the even-numbered discs are on the leftmost post while the odd-numbered discs are on the rightmost post, how many moves do we need to exchange the discs following the Hanoi rules for disc movement?

Write a method that calculates the number of moves necessary to exchange n discs.

8 How big is your stack? (*)

Write a method that calculates the maximum size of the stack (measured in method calls) by forcing a `StackOverflowError`. Use different version of your method, each of them with a different number and different types of local variables, and see how the number of maximum function calls changes depending on that.