CS1 — Practical Session 4: Loops

Kurt Driessens

September 19, 2018

In this fourth practical session you will get some personal experience with loops in Java.

By the way: I am assuming that you finished the exercises of last session at home if you couldn't finish them during the lab. If not, see to them first, but do it at home next time.

Exercise 1

Write a program that first asks the user for a word and then asks for a number, and prints out the word as many times as indicated by the number.

Exercise 2

Read in an unspecified number of integers, typed in on 1 line, separated by spaces and terminated with a letter. When the line ends, write out the maximum and minimum of the numbers.

Exercise 3

Write a Java program that prints a pyramid like this:

```
    []
  [] [] []
[] [] [] [] []
```

Let your program ask the user for the height of the pyramid. (Hint: you will need multiple nested loops for this exercise.)

If you are having difficulties, try having your program print a staircase first:

```
[]
[] []
[] [] []
```

then adapt it to print the pyramid.

Assignment!

Write a method that for any positive whole number determines whether the number is prime or not (this means the method will return a boolean). Then use the method to count the number of primes between 0 and 10000.

As usual, upload your `.java` file on EleUM under the lab-session 4 assignment.

Bonus — so you don't get ... right, you know by now.

It is known in mathematics that if you take any natural number n (i.e. 1, 3, 27), apply the following rules:

1. if n is even: divide n by 2
2. if n is uneven: multiply by 3 and add 1

and continue to do this with the new number you obtain, you will at some point always reach the number 1.

Implement a Java Program that simulates this process, i.e. that reads in a integer number and then uses the rules listed above to build a sequence of numbers until it reaches the value 1. Your program should print out all the numbers that make up the sequence together with, at the end, the number of steps it took to reach the value 1.