Programming 1

Christian Grévisse (christian.grevisse@uni.lu)
Gilles Neyens (gilles.neyens@uni.lu)

Lab 3 – Iterative Structures

Exercise 1 – Chalkboard Gag

At the beginning of many episodes of *The Simpsons*, a chalkboard gag shows Bart writing some funny phrase on the chalkboard a number of times, as in figure 1. Write a program that lets Miss Krabappel specify the number of times Bart has to write, e.g., the sentence "I am not allergic to long division" (or some other phrase of your choice). This number is read from the console and the sentence is printed to the console as many times.

Figure 1 – Copyright: 20th Century Fox

Exercise 2 – Sum

Write a program that reads from the console a positive integer n and that calculates the sum of all integers from 1 to n. As long as the user has not entered a *positive* integer, keep on asking. Calculate it in three different ways, using a `for` loop, a `while` loop and a `do-while` loop. Finally, compare the different results with the result of the formula

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$$

Exercise 3 – Average

Write a program that calculates the average of a sequence of integer numbers read from the console. If the user enters 0, the sequence will stop and the average will be shown in the console.

Exercise 4 – Translate digits to words

Write a program that reads an integer from the console and prints each digit of the number. For instance, if the number 314 is read, then the console shows `three one four`.

Hints:

- Use integer division and the modulo operator (%) to go over the different digits of the number.
- Use a `switch`-statement to map the digits to their textual representation.
- If the read number is negative, put `minus` at the beginning.

Exercise 5 – Decomposition of a 1 € coin

Write a program that calculates all possibilities of decomposing a 1 € coin into 20, 10 and 5 cent coins. Print those possibilities and their total number.

Programming 1

Christian Grévisse (christian.grevisse@uni.lu)
Gilles Neyens (gilles.neyens@uni.lu)

Lab 3 – Iterative Structures

Exercise 6 – Guess a number

Write a program that randomly chooses a number between 1 and 100 and lets the user guess which number was chosen. With each guess the program shall tell the user whether the guessed number was bigger, equal to or smaller than the searched number until the right number is guessed. Finally the program shall print out how many tries the user needed to guess the right number.

Pseudo-random numbers in Java

In order to generate pseudo-random numbers in Java, you may use the `Random` class from the `java.util` package (`import` at line 1). When instantiating a random number generator, the *seed* is *the initial value of the internal state of the pseudorandom number generator*[a]. A non-constant seed such as the current system time ensures a different sequence of pseudo-random numbers at each run (line 4). The `nextInt(int bound)` method (line 5) returns a number between 0 (inclusive) and the specified bound (exclusive).

```
1 import java.util.Random;
2 public class RandomExample {
3   public static void main(String[] args) {
4     Random generator = new Random(System.currentTimeMillis());
5     int randomNumber = generator.nextInt(42);
6   }
7 }
```

[a] https://docs.oracle.com/javase/8/docs/api/java/util/Random.html

Exercise 7 – Fibonacci sequence

Write a program that shows the Fibonacci sequence $\mathcal{F}$ until the n^{th} term $\mathcal{F}_n$. Your solution must be based on loops, not on recursion. Note that the Fibonacci sequence is defined as:

$$\begin{cases} \mathcal{F}_1 & = 1 \\ \mathcal{F}_2 & = 1 \\ \mathcal{F}_n & = \mathcal{F}_{n-1} + \mathcal{F}_{n-2} \quad n > 2 \end{cases}$$

Exercise 8 – Refactoring: Extracting variables

As seen in the lecture on Friday, there are several reasons why extracting variables is a useful refactoring technique wrt. code maintainability. In Eclipse, you can extract a variable from an expression by selecting it, right-clicking and selecting `Refactor` → `Extract Local Variable` .... Try this by extracting variables for each of the 3 expressions in the `if`-statement of the following code snippet.

```
1     String platform = "macOS Sierra";
2     String browser = "Safari";
3     double zoomLevel = 1.5;
4     if (platform.toUpperCase().indexOf("MAC") > -1 && browser.equals("Safari") && zoomLevel >=
          1.5) {
5       // do something ...
6     }
```