Programming in Java (23/24)

– Exercises Day 1 –

Learning goals

Before the next day, you should achieve the following learning goals:

- Be able to read and write simple Java programs with user input and text output that use integers, strings, conditionals, and loops.
- Compile and run Java programs on a computer.

Star exercises (labelled with (*) or (**)) are more difficult. **Do not try star-exercises unless the other ones are clear to you**.

1 Ende homage

Is there anything wrong with the following piece of code (hint: yes)? What does it do?

```java
        int i = 10;
        while (i < 5) {
            i = i + 1;
            System.out.println(i);
        }
```

2 Yet another loop

What does the following piece of code do?

```java
        java.util.Scanner scan = new java.util.Scanner(System.in);
        int i = scan.nextInt();
        while (i < 10) {
            i = i + 1;
            int j = scan.nextInt();
            if (j == 0) {
                break;
            } else if (j != 1) {
                System.out.println(j);
            }
        }
        System.out.println("finished");
```

Hint: the reserved word **break** exits the current loop.

3 Weird `if`

What does the following piece of code do? Why might this be the case?

```java
        int x = 1;
        if (x == 2); {
            System.out.println("x is " + x);
        }
```

4 Prime numbers

Write a program that asks for a number from the user, then says whether the number is prime or not. Remember that a prime number is a positive number (i.e., it is greater than 0) that is divisible only by 1 and itself. You can use the modulo operator (if `a % b` is zero, then a is divisible by b).

5 Multiplication

Write a program that requests two numbers from the user and then outputs their product. You cannot use the "*" operator.

Can you make your program work correctly also if one or both numbers are negative?

6 Division

Write a program that requests two positive numbers from the user and then outputs the quotient and the remainder, e.g., if the user enters 7 and 3, your program should output something like "7 divided by 3 is 2, remainder 1". You cannot use the "/" or "%" operators.

7 Naive sorting

Write a program that reads three numbers and prints them in order, from lowest to highest.

8 Maximising

Write a program that reads a (arbitrarily long) sequence of positive numbers. The sequence is ended when the users enters "-1". At that point, the program must output the highest number in the sequence.

9 Going up!

Read an arbitrarily long sequence of positive numbers from the user, until -1 is entered. At that point, print "Yes" if the numbers were consecutive and increasing and "No" otherwise. For example, the sequences "1,2,3,4,-1" and "5,6,7,8,9,10,11,-1" should lead to the output "Yes", but "2,3,5,6,7,-1", "10,9,8,7,-1", and "1,1,2,3,4,5,-1" should lead to the output "No".

10 You said high, I said low...

Modify your program from the previous exercise so that it outputs "Yes" when the numbers are consecutive, regardless of whether they go up or down. For example, both "2,3,4,5,6,-1" and "10,9,8,7,-1" should now result in "Yes".

11 Number pyramids

a)

Write a program that outputs a number pyramid like the one below, going on forever (until you press Ctrl-C to stop the program).

```
1
22
333
4444
55555
666666
7777777
...
```

(The formatting of the pyramid will mess up after a few numbers but that is OK for this exercise. If that bothers you, move on to the second and harder part of this exercise).

b) (*)

Write a program that reads a number between 1 and 25, and then outputs a number pyramid like the one below (the example is for number 8) with that number of levels. Notice that you must write the right number of spaces at each level so that the pyramid is properly aligned to the right.

```
              1
            2 2
          3 3 3
        4 4 4 4
      5 5 5 5 5
    6 6 6 6 6 6
  7 7 7 7 7 7 7
8 8 8 8 8 8 8 8
```

12 All the primes up to 1,000 (*)

Write a program that prints on screen all prime numbers up to 1,000.

13 Up to 1,000 primes (*)

Modify the program that you wrote for the former exercise so that it writes on screen the first 1,000 primes.

14 Guess my number (*)

Write a program that thinks of a random number between 0 and 1000, and then lets the user try to guess it. For every guess, the computer says whether the guess is correct, or too low, or too high. When the user finds the number, the computer will tell how many guesses were needed. The output could be similar to the following example:

```
Try to guess my number!
Tell me a number: 2
No! My number is higher.
Tell me a number: 800
No! My number is lower.
Tell me a number: 500
No! My number is lower.
Tell me a number: 350
No! My number is higher.
Tell me a number: 376
CORRECT!
You needed 5 guesses.
```

Hint: to get a random number between 0 and 1000, use the following line:

```java
        int numberToGuess = (int) Math.round(1000 * Math.random());
```

15 Rock, Paper, Scissors (*)

Write a program that reads 2 characters from the keyboard. The characters are either PP, PR, PS, RP, RR, RS, SP, SR, or SS. They correspond to the selections made by 2 players playing the game of rock-paper-scissors. Recall that rock beats scissors, paper beats rock, scissors beats paper.

Make the program accept inputs until one player's score is more than 3 points ahead of the other.

Hint: remember that you can use `.substring()` to get the elements of a String.

16 Optimus Prime (**)

Write a program that reads an integer number from the user, and then outputs the closest prime number. If there are two prime numbers at the same distance, it should output both. For instance, if the user enters 5116, the output should be 5113 *and* 5119.

17 π (**)

Pi (π), the ratio of a circle's circumference to its diameter, can be computed by adding the following terms:

$$\pi = 4\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \cdots$$

Create a program that asks the user for a number *n* and then calculates the partial addition of *n* terms of this infinite sum. How many terms do you need to get the first three digits right (3.14)? How many for the first 10 digits (3.14159265358...)?

Hint: to represent fractions in the computer, you can use *floating-point numbers*. A suitable data type for floating point numbers in Java is called **`double`** (for "double-precision floating-point numbers"). You can use the standard arithmetic operators "+", "-", "*", "/", ">", ... to work with values of type **`double`**. Doing maths with floating-point numbers will involve small rounding errors, but can be done very efficiently in a modern computer.