Programming 1

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

Lab 2 – Data Types, Decision Structures & Code Smells

2017–10–03

1 Data Types

In this part, you are not allowed to use any `if-then(-else)`, `switch/case` statements or `? :` constructions.

Exercise 1 – Tell me the truth!
The implication operator, denoted by $\rightarrow$, is defined by the following truth table:

a	b	a → b
false	false	true
false	true	true
true	false	false
true	true	true

Express the implication operator using only simple logical operators, i.e. *and* ($\wedge$, in Java `&&`), *or* ($\vee$, in Java `||`) and *not* ($\neg$, in Java `!`).

Write a program which implements your expression and displays its truth table. Verify that the displayed truth table and the truth table shown above are identical. Note that the priority of these logical operators, in decreasing order, is *not* > *and* > *or*, e.g.

$$a \wedge b \vee \neg c \iff (a \wedge b) \vee (\neg c)$$

$$a \wedge \neg b \vee c \iff (a \wedge (\neg b)) \vee c$$

$\iff$ means "equivalent to".

Exercise 2 – Comparison of two positive numbers
Write a program that reads two **positive** numbers from standard input (you may need the `Math.abs` method to adjust them being positive). The program indicates whether the first number is bigger than the second number. If it is, the program outputs 1, otherwise 0. Distinguish between the following cases:

1° The two numbers are integers. Use the integer division to do the comparison.

2° At least one of the two numbers is a real. Use the floating point division to do the comparison.

The methods `Math.min` and `Math.floor` might be useful.

Exercise 3 – Go with the (over-)flow
Using the `byte` data type, whose domain is $[-128; 127]$, show that an overflow occurs once the limits are reached. Consider both the top and the bottom limit of the domain. What do you observe, and how would you explain this?

Programming 1

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

Lab 2 – Data Types, Decision Structures & Code Smells

2 Decision Structures

Exercise 4 – United States of Aggregation
Write a program that reads a temperature (in Celsius) from standard input and displays whether water would be solid (ice), liquid or gaseous (steam) at that temperature.

Exercise 5 – Multiplication
Write a program that reads two integers from standard input and computes the sign of the product without actually computing the product. The output shall be 1 if the product is a positive number and -1 if it is negative. Instead of using an `if-then-else` statement, try to write this program using the `? :` construct.

Conditional operator `? :`
The conditional operator `? :` is a shorthand ternary operator (three operands) that can replace short `if-then-else` statements. For instance, the following code

```
int n;
if(<someCondition>) {
  n = 42;
} else {
  n = 21;
}
```

could be written as

```
int n = <someCondition> ? 42 : 21;
```

which makes your code a lot more concise.

Exercise 6 – Sorting
Write a program that reads three integers from standard input and displays them in ascending order.

1° First, provide a solution with nested `if`-statements.
2° Now, provide another solution without nesting.

Exercise 7 – Wait a second ...

1° Write a program that reads a time consisting of hours, minutes and seconds from standard input. The user may also specify the clock format (12-hour clock with AM/PM or 24-hour clock). Be sure to provide sanity checks for the indicated values (e.g. minutes are comprised between 0 and 59).
2° Display the indicated time in the format `hh:mm:ss`. Make sure to use leading zeros for numbers less than 10.
3° Advance the time by one second. For instance, if the input time is `09:59:59`, the new time will be `10:00:00`.
4° Display the new time.

Programming 1

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

Lab 2 – Data Types, Decision Structures & Code Smells

Exercise 8 – Rectangles

Write a program that determines whether a point $P(x_P; y_P)$ is inside a rectangle. The program reads the coordinates of P from standard input. It also reads the coordinates of the rectangle from standard input. A rectangle can be defined by a quadruple of coordinates $(x_{min}, y_{min}, x_{max}, y_{max})$ (we only consider rectangles whose sides are parallel to the axes).

ℹ Such a check may be used in game development for collision detection between sprites.

Exercise 9 – Crowdfunding Rewards

On crowdfunding platforms like Kickstarter or Indiegogo, it is common to offer a set of rewards to people who back a project by pledging a certain amount of money.

In this exercise, the user shall be able to enter the amount of money he wants to pledge. The available amounts are 10 €, 20 €, 50 €, 100 €, 200 €and 500 €.

The following rewards would be given the supporters of a project:

10 €	High five with TAs
20 €	Your name in ASCII art
50 €	Public display of your donation on Moodle
100 €	– USB Stick of 8GB – Public display of your donation on Moodle
200 €	– Uni.lu Smartphone cover – USB Stick of 8GB – Public display of your donation on Moodle
500 €	– Autograph of your professors – Uni.lu Smartphone cover – USB Stick of 8GB – Public display of your donation on Moodle

Using a `switch` statement, display these rewards. Try to minimize the number of `System.out.println` calls for each donation amount. If an amount not represented in the above table is entered, display some error message.

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

3 Code Styles & Code Smells

Exercise 10 – Smelly cat, smelly cat, ...
Remember Phoebe from the sitcom *Friends*? Her song *Smelly Cat* serves a perfect example for code smells, too!

Consider the following piece of code:

```
    String smellyDog = "smelly cat";
    System.out.println(smellyDog + ", " + smellyDog + ", what are they feeding you?");
    System.out.println(smellyDog + ", " + smellyDog + ", it's not your fault!");
```

While the code looks pretty great and even the output of the program is correct, there is a semantic smell in your code, as the name of the variable `smellyDog` is incorrect. This could later on cause some confusions, if the program gets further developed. In addition, there are 5 occurrences of this variable, so renaming it could go wrong if you forget an occurrence.

Therefore, consider one of the many *refactoring* options of Eclipse, namely renaming the variable. To do so, right-click on the variable's name and select `Refactor` → `Rename` .... Now, all the occurrences of the variable will appear with a border around, and while typing, all the occurrences are changed at once. Once you're done, press `Enter`. All 5 occurrences of the variable should now read `smellyCat` instead of `smellyDog`.

Exercise 11 – `goto fail`
You have seen in the lecture that the syntax of `if`-statements in Java allows to omit the curly braces in case of a single instruction to be executed.

What do you think about the following pseudo-code? Will `conditionB` even be evaluated? Why (not)? What security issues may this potentially cause? What consequences may this have for the designers of future programming languages?

```
if(<conditionA>)
  stopProgram();
  stopProgram();
if(<conditionB>)
  stopProgram();
```

↯ With respect to this topic, please consider reading on the *goto fail bug*, a piece of unreachable code in Apple's SSL/TLS implementation, discovered in early 2014.

Exercise 12 – Save Actions
Please refer to the document on *Save Actions* on Moodle to configure your Eclipse IDE accordingly.