UNIVERSITÉ DU LUXEMBOURG

Programming 1

Aryobarzan Atashpendar (aryobarzan.atashpendar@uni.lu)

Lab 3 – Data Types, Decision Structures & Code Smells

Jujutsu Kaisen © *Gege Akutami, Shueisha*

1 Decision Structures

Exercise 1 – Positive attitude #If

Write a program that reads an integer from standard input and tells you if it's a positive number. The output (print to console) shall be 1 if it is positive, 0 if not.

1° Implement this program using if-then-else.

2° The requirements of the program have changed! Now we want to display 0 only if the number is zero, and -1 if the number is negative. Again, use if-then-else for your implementation.

Exercise 2 – Let it be different! #LogicalOperator #Boolean

In the previous exercise, you probably used something similar to "if (number<0)". In this example, "number<0" is a numeric comparison resulting in a boolean value (true or false). From past logics courses, you probably already know about logical operators ("and", "or", "not")! One such operator, which is perhaps less often used, is the "exclusive or" (xor) operator, denoted by the symbol ⊕ and defined by the following truth table:

a	b	a ⊕ b
false	false	false
false	true	true
true	false	true
true	true	false

1° The AND logical operator ("&&" in Java) can intuitively be interpreted as the outcome being true only if both a and b are true. On paper, try to write its truth table, then write a Java program which displays this truth table: do the results match your paper-based truth table?

   - You do not need to display an elaborate truth table in the console. Instead, declare two variables "a" and "b", whose values you update for each corresponding line of your truth table and whose result you print to the console when applying the AND operator to them.

2° Repeat the same process for the OR logical operator ("||" in Java), whose outcome is true as long as either a or b is true (or both).

   - You do not need to copy and paste your previous code, just replace the operator of your existing code.

3° Repeat the same process for the EQUAL logical operator ("==" in Java), whose outcome is true if a is equal to b.

4° Repeat the same process for the NOT EQUAL logical operator ("!=" in Java), whose outcome is true if a is not equal to b.

Exercise 3 – Comparison **#ConditionalOperator**

Write a program which reads two integers (a and b) from standard input and which produces an output comparing them. The output shall be 1 if a>b, 0 if a==b and -1 if a<b.

1° Implement this program using if-then-else.

2° The conditional operator ("?:") allows you to select a value depending on the evaluation of a boolean expression. Specifically, with "val = <condition> ? outcome1 : outcome2;", val will be assigned the value outcome1 if <condition> is true, and otherwise outcome2.

   - Modify your previous implementation to use the conditional operator ("?:") rather than if-then-else.

Exercise 4 – Rectangles

Write a program which 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.

2 Data Types

Exercise 5 – Divide and rule #Conversions

Consider the following block of code:

```java
float div;
// A
div = 1/2;
System.out.println("1/2 = " + div);
// B
div = 1f/2;
System.out.println("1f/2 = " + div);
// C
div = 1/2f;
System.out.println("1/2f = " + div);
// D
div = 1f*1/2;
System.out.println("1f*1/2 = " + div);
// E
div = 1/2*1f;
System.out.println("1/2*1f = " + div);
```

1° Try to predict the outcome of each System.out.println without executing the code and indicate your predictions as comments above each System.out.println statement. Do you think they will all have the same output?

2° Test this code by executing it: is it the result you expected? Can you explain why it's happening?

Exercise 6 – Touch the floor #Casting

In Java, it is possible to convert some types to another using casting. For instance, "int a = (int) floatVariable;" will store inside the int-type variable "a" a converted version of the value stored by the float-type variable "floatVariable".

1° Write a program which reads a float value from standard input and which outputs the same value after a (int) cast. Try different values: what do you observe?

2° Modify your program to print the decimal part of the number which was input.

Exercise 7 – Go with the (over-)flow

Numeric data types have certain lower and upper limits. Since device memory is finite, it is not possible to store infinitely large or precise numbers. (like π for example)

1° 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?

2° If you use `int` as a type, you can easily store values like −129 and +128. However, int has its limits too! Look up those limits and write a program demonstrating what happens when you exceed them. Is there a type which can store even larger whole numbers?

3° The decimal types `float` and `double` behave a bit differently in terms of limits. The differences manifest in the precision of the value, rather than the magnitude. Write a simple program which adds two double numbers - a very small one and a very big one - and print the result. Is the result what you expected?

3 Code Styles & Code Smells

Exercise 8 – oi oi oi
In the series *Jujutsu Kaisen*, there is a debate as to who is the strongest sorcerer: Ryomen Sukuna or Gojo Satoru? Consider the following piece of code:

```
String ryomenSukuna = "gojo satoru";
System.out.println(ryomenSukuna + ", " + ", are you the strongest sorcerer?");
System.out.println("I, " + ryomenSukuna + ", " + " am the strongest sorcerer!");
```

While the code is syntactically correct, there is a semantic smell in your code: the name of the variable ryomenSukuna is inappropriate, as it does not match its assigned value. This could later on cause some confusion in case another developer wants to extend your code! In addition, there are 3 occurrences of this variable, so renaming it manually could take some time, and you could potentially forget to rename some occurrence.

Therefore, consider one of the many refactoring options of Visual Studio Code, such as renaming variables. To do so, right-click on the variable's name and select "Rename symbol...". Then, enter the new variable name in the text box that appears and press Enter. All 3 occurrences of the variable should now read gojoSatoru instead of ryomenSukuna.

Exercise 9 – 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 condition B 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();
```

ℹ With respect to this topic, consider reading up on the "goto fail" bug regarding a piece of unreachable code in Apple's SSL/TLS implementation, which was discovered in early 2014.