Exercise 1 Boolean types

Boolean expression can take two values: *true*, *false*. The operations on booleans in Java are as follows:

- values for `boolean` variables are `true` or `false`
- testing equality: `==`
- testing difference: `!=`
- testing order: `<, >, <=, >=`
- negation: `!`
- conjunction (and logic): `&&`
- disjunction (or logic): `||`

Example:

```
int x;
x = input.nextInt ();
boolean b = x > 0 && x < 10;
```

Note: the evaluation of boolean expression is made **from left to right** and **stops as soon as the value of the expression is known**. For example in `A && B` if `A` is `false` the expression `B` is not evaluated anymore since the value of the expression is known.

Similarly, in the expression `A || B` if `A` is true then the expression is true therefore `B` is not evaluated.

Exercise 2 The if structure

To define constants in Java, we declare them using the attribute `final`. For example:

```
if (boolean expression) {
    // instruction block A
} else {
    // instruction block B
}
```

If during the execution of the program the evaluation of the boolean expression is `true` then the block A is executed. Otherwise the block B is executed. If B is void then the `else` part can be ommited.

If the block of instruction is reduced to a single expression then the brackets can be removed. Nevertheless, using brackets all the time is a good habit to avoid unnecessary errors.

Exemplu:

```
int x, y;
// ...
if (x > y)
    System.out.println ("x e mai mare");
else
    System.out.println ("y e mai mare");
```

Exercise 3 The switch structure

The structure `switch` allows to have different instructions following the possible values of an expression. The syntax is as follows:

```
switch ( expression ) {
    case constante1:
        // ...
        break;
    case constante2:
        // ...
        break;
    // ...
    default:
        // ...
        break;
}
```

The expression is evaluated then the execution of the program continues starting from the `case` label whose constant equals the value of the expression. The execution continues up to the first `break` instruction. If no `case` label corresponds to the value of the expression then the `default` instructions are executed.

If `break` instructions are omitted then all instructions for the following cases are executed. Test the following code:

```
int x;
// ...
switch (x) {
    case 0:
        System.out. print ("A");
        break;
    case 1:
        System.out. print ("B");
    case 2:
        System.out. print ("C");
        break;
    default:
        System.out. print ("D");
        break;
}
```

Exercise 4 The ternary operator

It is a conditional operator allowing to construct expressions for which the value will be determined by the evaluation of a boolean expression. The syntax is as follows:

`(boolean expression)? expression A : expression B`

The value of this expression is the one in `A` if the boolean expression is `true`, otherwise it is given by `B`.

Consider the following example and explain it:

```
int x, y, z;
// ...
z = (x > y ? x : y );
```

This is equivalent to

```
int x, y, z;
// ...
if (x > y)
    z = x;
else
    z = y;
```

Exercise 5 Using an if statement in practice

(A) Write a program which asks the user for its name, its year of birth and its gender. Then write the message:

"Hello (NAME), you are (x) years old.

Replace the NAME with the given one and make sure the accords are correct.

The current year can be found with:

`Calendar. getInstance (). get(Calendar.YEAR)`

(B) Write a program which asks the user for a name and which outputs if the year has 365 or 366 days. Test your program for various inputs.

(C) Write a program which solves the second degree equation

$$ax^2 + bx + c = 0.$$

The inputs are a, b, c. Recall that:

- We need to compute $\Delta = b^2 - 4ac$.
- We have three cases:
  - $\Delta = 0$: a single real root: $x = -b/(2a)$.
  - $\Delta > 0$: two real roots
$$x_{1,2} = \frac{-b \pm \sqrt{\Delta}}{2a}.$$
  - $\Delta < 0$: two complex roots
$$x_{1,2} = \frac{-b \pm \sqrt{-\Delta}i}{2a}.$$

Complete the program to take into account the case $a = 0$.

Exercise 6 Simple Calculator

Write a program which reads an arithmetic expression of the type *operand operation operand* and which outputs the result. The operator could be either one of the basic operations: $+$, $-$, $\times$, $/$.

Indications: The operator is represented through a simple character. It can be read in a variable of type `char` in Java isong `input.next().charAt(0)`.

Make two variants, one using `if ... else` and one using `switch`.

Possible extensions: add additional operations.