TP2: Java language, type of expressions, constants, mathematical libraries

Finish TP1 if it is not already the case!

When evaluating expressions, the type of the result is determined by the type of operands. In general, an operation between operands of the same type (integer, float) will give a result of the same type. If the operands are of different type, the result will be of the "more general type". For example, an operation between an `int` and a `float` will give a `float`.

It is important to understand this rules, as we have already seen in the exercise regarding the conversion Fahrenheit-Celsius in TP1.

Exercise 1 Type of expressions in computations

Retake exercise 4 from previous week and explain the difference between

```
double fahr;
double celsius;
...
celsius = 5/9 * (fahr- 32);
```

and

```
double fahr;
double celsius;
...
celsius = 5 * (fahr- 32) / 9;
```

Explanations. In the two variants the variables are reals and the numbers are integers. Only the second variant gives the good result, while the first one gives 0. This is explained as follows:

`celsius = 5/9 * (fahr - 32)`

- The first operation is `(fahr - 32)`, the result is a real number.
- Then the division is computed, obtaining the division of 5 by 9 **as an integer**, which is zero.
- Finally, zero is multiplied with the result of the first operation. The result is a real (integer times float) and is equal to 0.

`celsius = 5 * (fahr - 32)/ 9`

- The first operation is `(fahr - 32)`, the result is a real number.
- Then the multiplication is computed, obtaining a real number: multiplication of the result obtained in the paranthesis with 5.
- Finally, the result of the multiplication, which is a float, is divided by 9. The first operand is real, therefore a **real division** is performed, giving an approximation of the correct result.

Therefore, we must make sure that the computations are made with the correct type of variables. This can be achieved in two ways:

Implicitely: using constants which are of the correct type

Explicitly: using a **cast** operation, which changes the type.

Examples: to obtain an implicit correct type we can use:
`celsius = 5.0/9 * (fahr - 32)` or `celsius = 5/9.0 * (fahr - 32)`
A casting operation can be used in the form:
`celsius = (double)5/9 * (fahr - 32);`
Nevertheless, we prefere using implicit typing in the form:
`celsius = 5.0/9.0 * ( fahr - 32.0)`

Remarks on casting. The casting is *not recommended*. We try to avoid using it except for situation when it is not possible to do otherwise. The inconvenients of this operations are as follows:

- reading the expressions becomes more difficult
- The maintenance of programs becomes more complexe (if we want to change the types of some expressions, for example)
- the programmer does not recieve the help of the compiler to detect some programming errors

Exercise 2 Constants

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

```
final double G = 9.81;
final int YEAR = 1984;
```

Exercise 3 Mathematical functions

The class `Math` defines a number of mathematical functions, which are generally applied to floating point numbers.

- trigonometric functions: `Math.sin`, `Math.cos`, `Math.tan`, `Math.atan`, etc.
- exponential and logarithm functions: `Math.exp`, `Math.log`, `Math.log10`, etc.
- Power function: `Math.pow`, `Math.sqrt`;
- Rounding functions: `Math.abs`, `Math.ceil`, `Math.floor`;

Test these mathematical functions. A more complete list and documentation can be found at: `http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html`.

Exercise 4 Solve some problems

a) Compute number of radiators. Write a program computing the number of radiators needed to heat a room. We know that a radiator can efficiently heat a volume of $8m^3$. The user will give the length, width and height of the room in meters.

b) Number of cereal boxes. We know that a ton equals 35273.96 ounces. Write a program which reads the weight in ounces of a breakfast cereal box and outputs the weight of the box in the metric system. Also compute the number of cereal packs that can be filled with a ton of cereals.

Exercise 5 Machine epsilon

When performing the addition or subtraction of two floating point numbers, they are shifted such that they have the same exponent part, then the operation is performed on the significant digits of the fractional numbers involved. See `https://en.wikipedia.org/wiki/Floating-point_arithmetic` for more details.

Some of the information contained in the smallest number is lost, leading to the so called *round-off* error.

When the ratio of two floating point numbers is too small, all information in the smallest number is lost. The smallest ratio between two floating point numbers for which addition leads to a modification is called **machine epsilon**. The machine epsilon can be found therefore as *the smallest floating point number* ε *such that* $(1 + \varepsilon) \neq 1$.

a) Use a while loop to find the machine epsilon for the `float` and `double` types.

- Initialize $\varepsilon = 1$
- While $(1 + \varepsilon) \neq 1$ replace ε by $\varepsilon/2$.

b) Compare the numerical results with the theoretical value: the machine ε is equal to 2^{-b} where b is the number of bits in the mantissa.