NPRG013 JAVA — Homework

Submission deadline: 13.12.2013 12:00pm (noon)

Submission:

- Submit your solution using the **CodEx** system available at `https://codex.ms.mff.cuni.cz/codex-java/`.
- Use the class `CodEx.java`, which is in the provided ZIP archive. Fill in your name into the javadoc comment of this class. Insert all your code to this class (in the case you need more classes, create them as inner classes). Do not change the class name or package name (otherwise the **CodEx** system would not be able to evaluate your code)! In the ZIP archive, there is also the INPUT file containing several examples and the OUTPUT file containing correct results for these examples. If you execute your solution on the examples from the file INPUT (on a unix system e.g. `java CodEx < INPUT`), the results must be exactly the same as in the OUTPUT file (do not submit your solution otherwise).

Assignment: simple calculator with unlimited precision

Create a simple calculator, which allows computing with unlimited precision. The program reads data from the standard input and prints to the standard output. The calculator can add, subtract, multiply and divide. Priority of the operators is as usual and can be changed by parentheses. The input format of the numbers is usual with a decimal dot (integer numbers are of course also allowed). Numbers can be negative (i.e. unary minus can be also used).

Expressions are separated by a new line. The program always prints out the result of the expression. The result of the expression is also stored to the special variable **last**, which can be used in the subsequent expression (for the first expression, the **last** variable is set to zero). The elements on a single line are separated by an arbitrary number of white spaces (space, tab,. . .).

If the expression does not correspond to the above rules, the program prints out the text CHYBA. The **last** variable is then set to zero.

Initially, the program computes with the precision set to 20 digits after the decimal point. The precision can be changed via the **precision** command.

Examples:

$3 + 2$	⇒	5
$2 * (3 + 3)$	⇒	12
$(2 - 3 * (1 + 2))$	⇒	-7
$2 * 3$	⇒	6
last -3	⇒	3
$10 *$ last	⇒	30
$1.2 + 4.3$	⇒	5.5
$1/3$	⇒	0.33333333333333333333
$1000000000 * 1000000000$	⇒	1000000000000000000
precision 30		
$1/3$	⇒	0.333333333333333333333333333333
$3 * * 5$	⇒	CHYBA
$last * 3$	⇒	0