Assignment 1

Instructor: Musard Balliu

September 4, 2016

Introduction

In this assignment, you will program a mini-calculator for 2 digit numbers (numbers between 1 and 99) using the schoolbook algorithm for addition, subtraction, multiplication and division.

In the elementary school, we learn addition $+$, subtraction $-$, multiplication $\times$ and division / of natural numbers for any digits. In this assignment, we will implement such operations by only for natural numbers between 1 and 99. The following examples provide a quick reminder of these operations.

Examples: **Addition** $77 + 38$, $25 + 6$, $4 + 99$, $6 + 3$

$$\begin{array}{rrrl} & & {\scriptstyle 1} & \longleftarrow \text{carry} \\ + & & 7 & 7 \\ & & 3 & 8 \\ \hline & 1 & 1 & 5 \end{array} \qquad \begin{array}{rrr} & {\scriptstyle 1} & \\ + & 2 & 5 \\ & 0 & 6 \\ \hline & 3 & 1 \end{array} \qquad \begin{array}{rrrr} & & {\scriptstyle 1} & \\ + & & 0 & 4 \\ & & 9 & 9 \\ \hline & 1 & 0 & 3 \end{array} \qquad \begin{array}{rr} & 6 \\ + & 3 \\ \hline & 9 \end{array}$$

We can do addition by writing one number below the other and then add one column at a time. If the sum of the last digits is greater than 9, as in the first example above ($7 + 8 = 15$), we must remember, namely *carry*, a 1 to the top of the next column and use it in the final addition.

Examples: **Subtraction** $27 - 9$, $11 - 66$, $6 - 95$, $7 - 8$

$$\begin{array}{rrr} - & 2 & {}_{1}7 \\ & {}_{1}0 & 9 \\ \hline & 1 & 8 \end{array} \qquad \begin{array}{rrr} - & 6 & 6 \\ & 1 & 1 \\ \hline & 5 & 5 \end{array} \qquad \begin{array}{rrr} - & 9 & {}_{1}5 \\ & {}_{1}0 & 6 \\ \hline & 8 & 9 \end{array} \qquad \begin{array}{rr} - & 8 \\ & 7 \\ \hline & 1 \end{array}$$

We can do subtraction between numbers with more than one digit by writing down the larger number first and the smaller number directly below it, making sure to line up the columns. Then, we subtract one column at a time as in the above examples. If a column has a smaller number on the top, we make that number larger by regrouping, namely *borrowing*, a 10 from the previous column ($10+7=17$), and subtracting 1 from that column ($2-1=1$). Finally, we need to be careful about the sign of the result, whenever the order of the numbers has been changed due to the rule above. In particular, the result in the last three examples above should be -55, -89 and -1, respectively.

Note that operands can in general be any numbers between 1 and 99, and the result can potentially have more than 2 digits (as in $4+99$) or be a negative number (as in $7-8$). You can take a look at the following online resources for an overview of addition and subtraction algorithms.

Similarly, you will implement multiplication and division algorithms for numbers between 1 and 99.

Examples: **Multiplication** 25×15, 5×25, 10×10, 95×72

$$\begin{array}{ccc} & 2 & 5 \\ \times & 1 & 5 \\ \hline 1 & 2 & 5 \\ 2 & 5 & \cdot \\ \hline 3 & 7 & 5 \end{array} \qquad \begin{array}{ccc} & & 5 \\ \times & 2 & 5 \\ \hline & 2 & 5 \\ 1 & 0 & \cdot \\ \hline 1 & 2 & 5 \end{array} \qquad \begin{array}{ccc} & 1 & 0 \\ \times & 1 & 0 \\ \hline & 0 & 0 \\ 1 & 0 & \cdot \\ \hline 1 & 0 & 0 \end{array} \qquad \begin{array}{cccc} & & 9 & 5 \\ & \times & 7 & 2 \\ \hline & 1 & 9 & 0 \\ 6 & 6 & 5 & \cdot \\ \hline 6 & 8 & 4 & 0 \end{array}$$

We can do multiplication by first multiplying the last digit of the second number (5) by the first number (25), and then multiplying the second digit of the second number (1) again by the first number (25), and finally adding the partial results ($125+250=375$); the operator $+$ is not shown in the figures. We just have to be careful about lining up the columns, whenever performing the second multiplication (in the example above, we do it by putting a dot . in the last column). We interpret the dot . as number 0, when performing the final addition. As for addition, multiplication is done digit-by-digit and *carrying* a digit to the next column whenever the result is greater than 9, as for example in $5*5=25$ where we have to remember a 2 (not shown in the figure).

Note that you are allowed to use the Java addition operator $+$ to add the partial results above, in the example $125+250=375$. You can take a look at the following online resource for an animated overview of the multiplication algorithm.

Examples: **Division** $78/11$, $99/3$, $13/77$, $99/4$

Division is slightly more advanced. You can take a look at the following online resource for an animated overview of the division algorithm.

Consider the first example above, $78/11$. We call the number to be divided into, namely 78, the *dividend*, and the number which divides the other number, namely 11, the *divisor*. The result of integer division is called *quotient* (7 in the example), and the remainder ($78\%11 = 1$) is unsurprisingly called *remainder* (1 is the example below). The goal of integer division is to calculate the quotient and the remainder for a given dividend and a given divisor, as shown in the first three examples below.

```
 7 8 | 1 1
-7 7 | 7
   1 |
```

```
 9 9 | 3
-9   | 3 3
 0 9 |
-  9 |
   0 |
```

```
 1 3 | 7 7
-  0 | 0
 1 3 |
```

```
 9 9     | 4
-8       | 2 4.7 5
 1 9     |
-1 6     |
   3 0   |
-  2 8   |
     2 0 |
-    2 0 |
       0 |
```

The last example above shows the schoolbook calculation of normal division, where also the decimals (numbers after the dot .) are calculated.

Main Task

You will program a mini-calculator in JAVA, which asks the user to choose an operator between $+, -, \times, /$, and provide two integers in the range between 1 and 99. Subsequently, the calculator makes use of the implementations of the above algorithms to compute the final result and print it to the console. The program should check whether or not the user's input is correct, namely that the operator is one of $+, -, \times, /$, and the input numbers are between 1 and 99. If the input is not correct, the program should ask the user to provide an input again, without terminating the program. Finally, if the user inputs a q, the program should terminate.

The program should present the following interface to the user.

```
+***********************************************+
*                                               *
*        Welcome to the DIT948 Calculator       *
*                                               *
+***********************************************+

Please enter the operator to use ('+', '-', 'x', '/')
or press 'q' to quit:  +
Please enter the first value: 77
Please enter the second value: 38
The sum is 115

Please enter the operator to use ('+', '-', 'x', '/')
or press 'q' to quit: *

Please enter the first value: 25
Please enter the second value: 15
The product is 375

Please enter the operator to use ('+', '-', 'x', '/') or
press 'q' to quit: q

Goodbye!
```

Remarks and Notation

1. It goes without saying that you *must* use the schoolbook algorithms described above to implement each operation. If you choose to only implement addition and subtraction (see **Grading** below), you can use the multiplication and division operators of Java to complete the task. Alternative solutions will not be accepted.
2. Integer division operator / and modulo (remainder) operator $\%$ in Java can be useful to implement the task.

Grading

- To get a G it is necessary that the program works for at least for addition and subtraction.
- To get a VG it necessary that the program works for multiplication and (integer) division.
- Too easy? You can implement normal division (as illustrated in the fourth example above) and calculate decimals up to a given threshold. This extra

task will not count to the final grade, however, it will definitely impress the instructor.

Note that these requirements are necessary, but not sufficient, to get the respective grades (see **Administrative** matters).

Administrative matters

Strive for readable code with appropriate comments! While the ultimate test of a program is that it does what it is supposed to do, **we should be able to read the program and understand it**.

The assignment is to be completed in groups of two students. (Groups of different size should first be agreed with the course instructor.)

Solutions must be uploaded to the GUL system **by 23:59 on September 14**.

Please note: **The submission must be made via GUL. It's no good sending it to me via email, either before or after the deadline!**

Only Java source files should be uploaded, no class files. Your Java files should be put in a zip-archive with the following name:

```
assign1_author1_author2.zip
```

where author1, author2 are the surnames (family names) of the group members. You must also supply a README-file (README.txt) containing the names of the authors and their social security numbers, together with a brief statement about each author's contribution. For example, "author1 has been responsible for user input and author2 for the program logic". A statement such as "All authors have contributed equally to all aspects of the program" is not acceptable.

Note that **each author must submit individually via the GUL** (even though it is the same program!). Only students who submit via the GUL can be graded. All comments etc. will be posted on the course web page.

Once the deadline has passed, each group **must explain solutions to the TAs during the first Thursday supervision session**. Exceptions are to be agreed with the instructor and the TAs. Group members are expected to know and explain all parts of the code despite their statement of contribution.