**CmpE 150 Introduction to Computing, Fall 2019**

**Assignment 2 – Due: 01/12/2019, 23.59**

---

You will write a Java program to evaluate arithmetic expressions using four lines of input which you will get from the user. One example input and output is provided below. Please read the assignment description completely before beginning to write any code.

**Example Input-1:**

```
int x=3;
int y      = 5 ;
double   z =   6    ;
(2+ 3*5   /  2    *x+  y*   (126 +z*2));
```

**Example Output-1:**

```
713.0
```

Please make sure you follow these rules in your implementation:

1. Your program should have at least two static methods in addition to your main method. Try to write your program as modular as possible (without overusing methods).
2. You are not allowed to use String methods other than:
    - replace
    - substring
    - length
    - contains
    - indexOf
    - lastIndexOf
    - equals
    - charAt

    You can obviously use the + operator for string concatenation.
3. You are not allowed to use any other class then String, Integer, Double, and Scanner. Note that this does not mean that you cannot use datatypes like int, double, boolean etc. You can use the datatypes you have learned in class.
4. You can and should make use of all the subjects you have learned in class such as while loops, conditional statements, booleans, etc.
5. You can use methods from the Integer and Double classes. For example:
    - Double.parseDouble(String arg0)
    - Double.toString(double d)

- Integer.parseInt(String arg0)
- Integer.toString(int i)

**Implementation Details:**

- "input which you will get from the user" means you need to use Scanner.
- The first three lines are going to be your variables. You will use their types and values as given in the input. Note that z in the **Example Input-1** is a double, so its value is 6.0.
- The fourth line will be the expression you are expected to evaluate and return the result of. Given expression can contain integers and/or doubles.
- Variable types in the input can change, but their types will be only int or double.
- Assume no erroneous input will be given.
- A few things to note on the input:
    1. All the input lines end with a semicolon.
    2. There can be any number of empty spaces anywhere after the variable type is given, up until the semicolon for the first three lines of input. (There are no empty spaces before the type keyword and there are no empty spaces after the semicolon.)
    3. There can be any number of empty spaces anywhere up until the semicolon for the fourth line of the input. (There are no empty spaces after the semicolon.)

    *You should format the inputs and the expression so that you can operate on them easier.*

- The order of precedence for the operations are parentheses first, then multiplication-division and then addition-subtraction. There won't be any other operations.
- For operations with the same precedence, the order of operations is from left to right.
- Parentheses can be nested and will require special effort to handle.

    *Think carefully on how to write your code such that it can solve for inputs with nested parentheses.*

- Operations performed need to consider the data types involved, for example, if you are doing integer division like 7/2, the answer should be 3; but if it is 7.0/2 the answer should be 3.5. This will also affect your result. If your answer is 65.0 (a double) but it was supposed to be 65 (an integer) this is a mistake.
- Assume that the operations will never cause an overflow/underflow situation, and no operation will result in a negative number.

**Submission:** You will submit a project report and your code over Moodle.

Project report should consist of five sections. These are:

1. Problem Description: In this section, you should describe the problem in your own words.

2. Problem Solution: In this section, you should specify the concepts (methods, loops, conditionals

etc.) that you use in your program. Explain each one (i.e. why you need it, what you accomplish by using it, so on.) broadly.

3. Implementation: This section will include your whole code with comments. You need to pay attention to indentation in order to improve readability.
   - Do not forget to explain each variable that you use (i.e. int count = 0; // count is the number of items).
   - Before each method, specify what the method does (i.e. /* This method … */)

4. Output of the program: A screenshot of your program input and output should be put in this section. One example run for each functionality of your implementation is enough.

5. Conclusion: You should evaluate your work here. State whether you have solved the problem correctly. If not, state what is missing, what could have been improved, and so on.

6. Your .java file should be named with your initials and your student number together (e.g., OS2013800027). If you have Turkish characters as your initials, please change them to non-Turkish. (Example: ÖS2013800027 should be OS2013800027) You will submit these over Moodle as a single zip file where the file name is your student number. Your zip file should consist of your .java file and your report in .doc or .pdf format. Do not use any Turkish characters in your code, class/variable names, or .java file names.

**Partial Submission:** If you cannot complete the assignment, you should still submit your code as well as your report. Try to implement as much as you can. In your report, explain which parts you were able to complete and which parts you were not. Partial credit will be given depending on your output. If you can produce the correct output for multiple tests but not all, you will still get partial credit.

A: If you can perform summation and subtraction, for only integers in expression: +20

B: If you can perform multiplication and division for only integers in expression: +20

C: A and B and if you can solve statements with combined operators (for example both addition

and multiplication may appear in the expression), for only integers in expression: +20

D: A or B or C and if you can solve statements with non-nested parentheses, for only integers in expression: +10

E: D and if you can solve statements with nested parentheses, for only integers in expression: +10

F: A or B or C and if you respect the integer/double operations (such as integer division) as well, starting from this case you are required to handle integers and/or doubles in the expression: +10

G: E and F and if you can solve statements with the input variables (double-int-double, int-int-double, etc.): +10

**Example Case-1:** Let's say in your implementation you do not use the input variables (first three lines) at all, but you take the fourth line and can successfully perform summation, subtraction, multiplication and division including their combinations for integers only, and you can handle non-nested parentheses. If you pass all relevant test cases, you will get 70 from the project. (case D with case C)

**Example Case-2:** Let's say in your implementation you use the input variables, and can successfully perform summation, subtraction, multiplication and division operations but can't combine them together respecting the operator precedence, but you can perform the operations respecting integer/double constraints and you can even handle nested parentheses and your implementation uses the input variables as well. If you pass all relevant test cases, you will get 80 from the project. (case G with A and B but not C)

**Late Submission:** Late submission is possible with penalty = -10*(daysLate)^2.

**Evaluation Procedure:** We will evaluate your code based on the program output for many test cases.

You can create test cases on your own and compare the outputs.

**Notes:**

*You should obey the rules otherwise depending on the severity you might lose **all the points.***

*Start early.*

*Good luck && have fun.*

**Example Input-2: case E with C**

```
double a = 7;
int b=8;
double c = 79.2 ;
      (2+ 4* 3   *(8-2)/ 3);
```

**Example Output-2:**

```
26
```

**Example Input-3: case A**

```
int d  = 1;
int e =2;
double f = 3.4;
  2 +74 +    396     -213;
```

**Example Output-3:**

```
259
```

**Example Input-4: case E with A**

```
int g = 4;
double h= 13.76;
int i = 90;
9 + ( 78    -  (3  + 44)) - 4;
```

**Example Output-4:**

```
36
```

**Example Input-5: case B**

```
double j = 58;
int k = 2;
double L = 1.7;
2 *4 *13   / 3      ;
```

**Example Output-5:**

```
34
```

**Example Input-6: case C**

```
int m = 13;
int n = 27;
int o= 39;
3 + 9 *39 - 27/3 *13;
```

**Example Output-6:**

```
237
```

**Example Input-7: case F with C but without D or E**

```
int p = 6;
int r = 4;
double s = 3.1;
12.8*9-76/3;
```

**Example Output-7:**

```
90.2
```

**Example Input-8: case F with B and D**

```
Int t = 1;
int u = 409;
double v = 56;
  2*56.0 * (409/7)*3   ;
```

**Example Output-8:**

```
19488.0
```

**Example Input-9: case G**

```
double x = 2.4;
double y = 18.2;
double z = 4;
(   3 + (5-x)*(y-4)-z+ 2*9);
```

**Example Output-9:**

```
53.92
```

**Example Input-10: case G**

```
double w = 4.3;
double q = 39.66;
int aa = 71;
aa/4-w+ (q*w-1)*w+(((aa-q)));
```

**Example Output-10:**

```
773.0534
```