Deliverables

Your completed project files should be submitted to Web-CAT by 11:59 p.m. on the due date in order to avoid a late penalty for the project. If you are unable to submit via Web-CAT, you should e-mail your project Java files in a zip file to your TA before the deadline.

Files to submit to Web-CAT:

- ExpressionResult.java
- TimeConverter.java

Specifications

Overview: You will write two programs in this project week. The first will find the result of a specified expression after reading input values for x, y, and z, and the second will calculate the number of days, hours, minutes, and seconds from a raw measurement of seconds.

- **ExpressionResult.java**

Requirements: A program is needed that inputs values of type double for x, y, and z and solves for the result of the indicated expression when *xyz* is not equal to zero. If *xyz* is equal to zero, then the result is undefined.

Design: The result should be calculated as follows (except for the special case):

$$result = \frac{(x - 1.1)\,(2y + 2.2)\,(4z + 4.4)}{xyz} \qquad \text{for } xyz \neq 0$$

Note: if *xyz* is 0, then *result* is undefined.

Five examples of program output for the indicated input values are shown below. Note that lines 2,3, and 4 for the input values begin with tab which is equivalent to three spaces in jGRASP (i.e., your program should use the \t **escape sequence for a tab**).

Example #1

Line #	Program output
	`----jGRASP exec: java ExpressionResult`
1	`result = (x - 1.1) (2y + 2.2) (4z + 4.4) / xyz`
2	`   x = 1.0`
3	`   y = 1.0`
4	`   z = 1.0`
5	`result = -3.528000000000003`
6	
	`----jGRASP: operation complete.`

Example #2

Line #	Program output
1 2 3 4 5 6	----jGRASP exec: java ExpressionResult result = (x - 1.1) (2y + 2.2) (4z + 4.4) / xyz x = 10.5 y = 11.6 z = 12.7 result = 8.520197044334974 ----jGRASP: operation complete.

Example #3

Line #	Program output
1 2 3 4 5 6	----jGRASP exec: java ExpressionResult result = (x - 1.1) (2y + 2.2) (4z + 4.4) / xyz x = 0 y = 1.7 z = 12.3 result is "undefined" ----jGRASP: operation complete.

Example #4 (note that if zero is entered for z, the result would be the same as below)

Line #	Program output
1 2 3 4 5 6	----jGRASP exec: java ExpressionResult result = (x - 1.1) (2y + 2.2) (4z + 4.4) / xyz x = 1.5 y = 0 z = 2.5 result is "undefined" ----jGRASP: operation complete.

Example #5

Line #	Program output
1 2 3 4 5 6	----jGRASP exec: java ExpressionResult result = (x - 1.1) (2y + 2.2) (4z + 4.4) / xyz x = -1.0 y = -10.0 z = -100.0 result = 14.787528000000002 ----jGRASP: operation complete.

Code: Your numeric variables should be of type double. Use an if-else statement to determine if the divisor *xyz* in the expressions is zero; i.e., `if (x * y * z == 0) {` . . .
Example #3 and #4 above show the input/output where *x* is zero and *y* is zero respectively and *result* is undefined. Note that if zero is entered for z, the result would also be undefined.

Test: You are responsible for testing your program, and it is important to not rely only on the examples above. Remember that the input values are doubles, so be sure to test both positive and negative values (with and without a decimal point) for x, y, and z. You should use a calculator or jGRASP interactions to check your answers.

- **TimeConverter.java**

Requirements: A digital timer manufacturer would like a program that accepts a raw time measurement in seconds (of type int) and then displays the time as a combination of days, hours, minutes, and seconds. When a negative raw time measurement is entered, an appropriate message is printed as shown in the first of the two examples below. This program should use integer division to find the number of days, then use the remainder to calculate the next smaller unit. That is, after finding as many days as possible using the division (/) operator, use the remainder (%) operator to find the number of remaining seconds, which is then used to calculate the number of hours. Continue this process using the division and remainder operators to find the numbers of minutes and seconds.

Design: The digital timer manufacturer would like the output to look as shown below when the test values -1234 is entered as the raw time for one run and 654321 is entered for another run.

Line #	Program output
1	----jGRASP exec: java TimeConverter
2	Enter the raw time measurement in seconds: -1234
	Measurement must be non-negative!
	----jGRASP: operation complete.

Line #	Program output
	----jGRASP exec: java TimeConverter
1	Enter the raw time measurement in seconds: 654321
2	
3	Measurement by combined days, hours, minutes, seconds:
4	days: 7
5	hours: 13
6	minutes: 45
7	seconds: 21
8	
9	654321 seconds = 7 days, 13 hours, 45 minutes, 21 seconds
10	
	----jGRASP: operation complete.

Your program must follow the above format with respect to the output. Note that lines 4 through 7 for the amount 654321 begin with tab (i.e., your output should use the **escape sequence for a tab**).

Code: In order to receive full credit for this assignment, you must calculate the number of days, hours, minutes, and seconds and store each of the values in separate variables. Create a Scanner object on System.in to read in the value for the raw time using the nextInt() method. It is recommended as a practice that you do not modify input values once they are read in and stored.

> ***Hint****: If you are not sure how to approach the solution for this program, think of making change for 43 cents with quarters, dimes, nickels, and pennies using the high value coins first. You would have 1 quarter, 1 dime, 1 nickel, and 3 pennies. To arrive at these values, you would first figure out how many quarters you have in 43 cents by dividing by 25, which would yield 1 quarter. Then you would use the reminder, from the division, or 18 cents, as the number of cents left of the original 43. You would divide the 18 cents by 10 to see how many dimes you had, which would yield 1 dime, and so on. You can apply this same approach when starting with the raw time in seconds then calculating the number of days, hours, minutes and seconds.*

Test: You will be responsible for testing your program, and it is important to not rely only on the example above. Assume that the amount entered can be any integer less than or equal to 2,147,483,647 (the maximum value for a 32 bit int) and greater than or equal to -2,147,483,648 (the minimum value for a 32 bit int).

Grading

Web-CAT Submission: <u>You must submit both "completed" programs to Web-CAT at the same time</u>. Prior to submitting, be sure that your programs are working correctly and that they have passed the Checkstyle audit. **If you do not submit both programs at once, the submission will receive zero points for correctness**. Activity 1 describes how to create a jGRASP project containing both of your files.