Deliverables

Your project files should be submitted to Web-CAT by the due date and time specified. In order to avoid a late penalty for the project, you must submit your completed code files to Web-CAT by 11:59 p.m. on the due date. 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:

- ExpressionSolver.java
- PetroleumUnits.java

Specifications

Overview: You will write two programs this week. The first will find the result of a specified expression after reading input values for x, y and z, and the other will determine the number of barrels, gallons, quarts, and ounces for an input value of a raw amount in ounces.

- **ExpressionSolver.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 should be reported as undefined rather than infinity.

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

$$result = \frac{(9x + 6.25)\,(6y - 4.5)\,(3z + 2.75)}{xyz} \qquad \text{for } xyz \neq 0$$

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

Three 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 ExpressionSolver`
1	`result = (9x + 6.25) (6y - 4.5) (3z + 2.75) / xyz`
2	`x = 1.0`
3	`y = 2.0`
4	`z = 3.0`
5	`result = 223.984375`
	`----jGRASP: operation complete.`

Example #2

Line #	Program output
1 2 3 4 5	----jGRASP exec: java ExpressionSolver result = (9x + 6.25) (6y - 4.5) (3z + 2.75) / xyz x = 0.0 y = 23.45 z = 67.8 result is "undefined" ----jGRASP: operation complete.

Example #3

Line #	Program output
1 2 3 4 5	----jGRASP exec: java ExpressionSolver result = (9x + 6.25) (6y - 4.5) (3z + 2.75) / xyz x = 12.3 y = 0.0 z = 45.6 result is "undefined" ----jGRASP: operation complete.

Example #4

Line #	Program output
1 2 3 4 5	----jGRASP exec: java ExpressionSolver result = (9x + 6.25) (6y - 4.5) (3z + 2.75) / xyz x = 12.3 y = 45.6 z = 0.0 result is "undefined" ----jGRASP: operation complete.

Example #5

Line #	Program output
1 2 3 4 5	----jGRASP exec: java ExpressionSolver result = (9x + 6.25) (6y - 4.5) (3z + 2.75) / xyz x = 1.0 y = 0.5 z = -1.0 result = -11.4375 ----jGRASP: operation complete.

Code: Your numeric variables should be of type double. Use an if-else statement to determine if the divisor in the expressions is zero. Note that in the example output above, one of the variables is zero in Example #2, #3, and #4, which means the divisor, *xyz*, is zero, and thus, *result* is undefined in each case. *Hint: your if statement should check to see if (x * y * z == 0).*

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.

- **PetroleumUnits.java**

Requirements: A petroleum company would like a program that allows the user to enter an amount of petroleum in ounces, which must not exceed 1 billion, and then displays the number of barrels, gallons, quarts, and ounces. The number of each of these should be maximized from largest to smallest as indicated in Examples 2 and 3 below. Your program should use the following conversion values in the computation:

1 barrel = 42 gallons (or 5376 ounces), 1 gallon = 128 ounces; 1 quart = 32 ounces.

Design: The petroleum company would like for the program's I/O to be as shown in the three example runs below where (1) 1234567890 is entered, (2) 654321 is entered, and (3) 1234567 is entered.

Example 1

Line #	Program output
1 2	`Enter amount of petroleum in ounces: 1234567890` `Amount must not exceed 1,000,000,000.`

Example 2

Line #	Program output

1
2
3
4
5
6
7
	`Enter amount of petroleum in ounces: 654321`
	`Petroleum amount in units:`
	`  Barrels: 121`
	`  Gallons: 29`
	`  Quarts: 3`
	`  Ounces: 17`
	`654321 oz = (121 bl * 5376 oz) + (29 gal * 128 oz) + (3 qt * 32 oz) + (17 oz)`

Example 3

Line#	Program output

1
2
3
4
5
6
7
	`Enter amount of petroleum in ounces: 1234567`
	`Petroleum amount in units:`
	`  Barrels: 229`
	`  Gallons: 27`
	`  Quarts: 0`
	`  Ounces: 7`
	`1234567 oz = (229 bl * 5376 oz) + (27 gal * 128 oz) + (0 qt * 32 oz) + (7 oz)`

Your program must follow the above format with respect to the output. Note that lines 3 through 6 for Examples 2 and 3 begin with tab (i.e., you must use the **escape sequence for a tab**).

Code: Your numeric variables should be of type *int*. A simple if-else statement can be used to check that the amount does not exceed one billion, where the true block prints the error message, and the false block prints the normal output (or vice-versa). Also, the return statement (`return;`) can be used in an if statement to return from main to immediately end the program after the error message. In order to receive full credit for this assignment, you must calculate the number of barrels, gallons, quarts, and ounces and store them in separate variables. It is recommended as a practice that you do not modify input values once they are stored.

Test: You will be responsible for testing your program, and it is important to not rely only on the example above. For example, test with the following amounts: 16, 32, 128, and 5376. The last amount is the number ounces in a barrel (42 * 128).

Grading

Web-CAT Submission: You must submit both "completed" programs to Web-CAT at the same time. Prior to submitting, be sure that your programs are working correctly and that they have passed Checkstyle. **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, which is recommended.