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:

- ATM.java
- FormulaCalculator.java

Specifications

Overview: You will write two programs in this project. The first will calculate the minimum number of paper currency denominations (twenties, tens, fives, and ones) in a specified number of dollars, and the second will calculate the result of formula for inputs x, y, and z.

- **ATM.java**

Requirements: A bank would like to have a program for its ATM department that allows the user to enter a value in whole dollars and then displays the combination of twenties, tens, fives, and ones such that each denomination is maximized in order by twenties, tens, fives, and ones. The input value should not exceed 500 dollars.

Design: The bank would like the output to look as shown in the examples below where 12345 is entered as the input for one run and 399 is entered for another run.

Line number	Program output
1 2 3	`Enter the amount: 12345` `Limit of $500 exceeded!`

Line number	Program output

1
2
3
4
5
6
7
8
	`Enter the amount: 399`
	`Bills by denomination:`
	`   $20: 19`
	`   $10: 1`
	`   $5: 1`
	`   $1: 4`
	`$399 = (19 * $20) + (1 * $10) + (1 * $5) + (4 * $1)`

Your program must follow the above format with respect to the output. Note that lines 3 through 6 for the input value of 399 begin with tab which is equivalent to three spaces in jGRASP (i.e., your output should use the **escape sequence for a tab**).

Code: In order to receive full credit for this assignment, you must read in a value and store it in a variable of type *int*, calculate the number of each denomination (twenties, tens, fives, and ones)

and store each value in a variable of type *int*. It is recommended as a practice that you do not modify any input values once they are stored. Your expressions should contain only *int* variables or *int* literals. Commas are not allowed in Java numeric literals (e.g., 1000 may be entered as 1000 or 1_000).

Test: You will be responsible for testing your program, and it is important to not rely only on the example above. The amount entered can be any valid *int* value. Consider testing with the input values for which it is easy calculate the number of twenties, tens, fives, and ones.

- **FormulaCalculator.java**

Requirements: A program is needed that inputs values for x, y, and z and calculates a result for the indicated formula. If the product of x, y, and z is zero, the result is zero.

Design: The result should be calculated as follows:

$$result = \frac{(2x - 7.4)\ (4y + 9.3)\ (6z - 11.2)}{xyz} \qquad \text{for xyz} \neq 0$$

Special case:

$$result = 0 \qquad \text{for xyz} = 0$$

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

Example #1

Line number	Program output
1 2 3 4 5 6	result = (2x - 7.4) (4y + 9.3) (6z - 11.2) / xyz Enter x: 1 Enter y: 1 Enter z: 1 result = 373.464

Example #2

Line number	Program output
1 2 3 4 5 6	result = (2x - 7.4) (4y + 9.3) (6z - 11.2) / xyz Enter x: 0 Enter y: 1 Enter z: 2 result = 0.0

Example #3

Line number	Program output
1	`result = (2x - 7.4) (4y + 9.3) (6z - 11.2) / xyz`
2	`   Enter x: 5.4`
3	`   Enter y: -20.25`
4	`   Enter z: 123.45`
5	`result = 13.173873451928744`
6	

Code: Your numeric variables should be of type double. Use an if-else statement to determine if the divisor in the formula is zero. Note that in Example #2, the value of x is zero so the divisor evaluates to zero.

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.

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, Web-CAT will not be able to compile and run its test files with your programs which means the submission will receive zero points for correctness**. I recommend that you create a jGRASP project and add the two files. Then you will be able to submit the project to Web-CAT from jGRASP. Activity 1 (pages 5 and 6) describes how to create a jGRASP project containing both of your files.