Project: Variables and Expressions

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:

- ThreeVariableExpression.java
- ATM.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, z and the other will determine the number bills to be dispensed from an ATM by denomination ($20, $10, $5, and $1 bills).

- **ThreeVariableExpression.java**

  Requirements: A program is needed that inputs values of type double for x and y 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):

$$result = \frac{(3.5x - 7.75)\ (2.85y + 6.0)\ (1.5z - 3.1)}{xyz} \qquad \text{for } xyz \neq\ 0$$

  If *xyz* is 0, then *result* should be reported as `"undefined"` as shown in Examples #2, #3, and #4 below.

  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 ThreeVariableExpression
1        result = (3.5x - 7.75) (2.85y + 6.0) (1.5z - 3.1) / xyz
2           x = 1.0
3           y = 1.0
4           z = 1.0
5        result = 60.18
6
          ----jGRASP: operation complete.
```

Example #2

Line # Program output

```
----jGRASP exec: java ThreeVariableExpression
result = (3.5x – 7.75) (2.85y + 6.0) (1.5z – 3.1) / xyz
   x = 0
   y = 21.5
   z = 88
result is "undefined"

 ----jGRASP: operation complete.
```

Example #3

Line # Program output

```
----jGRASP exec: java ThreeVariableExpression
result = (3.5x – 7.75) (2.85y + 6.0) (1.5z – 3.1) / xyz
   x = 10
   y = 0
   z = 5
result is "undefined"

 ----jGRASP: operation complete.
```

Example #4

Line # Program output

```
----jGRASP exec: java ThreeVariableExpression
result = (3.5x – 7.75) (2.85y + 6.0) (1.5z – 3.1) / xyz
   x = 10.0
   y = 5.0
   z = 0.0
result is "undefined"

 ----jGRASP: operation complete.
```

Example #5

Line # Program output

```
----jGRASP exec: java ThreeVariableExpression
result = (3.5x – 7.75) (2.85y + 6.0) (1.5z – 3.1) / xyz
   x = -1.0
   y = -2.0
   z = -3.0
result = -4.274999999999998

 ----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.

- **ATM.java**

Requirements: An ATM manufacturer would like a program that allows the user to enter the amount of cash in whole dollars and then displays the number bills by denomination ($20, $10, $5, and $1 bills) be dispensed if the limit of $500 is not exceeded. Your program should calculate the maximum number of twenties in the amount entered, followed by the maximum number to tens in what's left after the value of the twenties has been removed, and so on. For example, if 36 is entered as the amount, you would have one of each denomination as shown in Example #2 below.

Design: The ATM manufacturer would like the output to look as shown below when 501 is entered as the amount, when 36 is entered, and when 432 is entered three separate runs.

Example #1

```
Line number   Program output
              ----jGRASP exec: java ATM
1             Enter the amount: 501
2             Limit of $500 exceeded!
3
               ----jGRASP: operation complete.
```

Example #2

```
Line number   Program output
              ----jGRASP exec: java ATM
1             Enter the amount: 36
2             Bills by denomination:
3                $20: 1
4                $10: 1
5                $5: 1
6                $1: 1
7             $36 = (1 * $20) + (1 * $10) + (1 * $5) + (1 * $1)
8
               ----jGRASP: operation complete.
```

Example #3

```
Line number   Program output
              ----jGRASP exec: java ATM
1             Enter the amount: 432
2             Bills by denomination:
3                $20: 21
4                $10: 1
5                $5: 0
6                $1: 2
7             $432 = (21 * $20) + (1 * $10) + (0 * $5) + (2 * $1)
8
```

	`----jGRASP: operation complete.`

Your program must follow the above format with respect to the output. Note that lines 3 through 6 for the amount 432 begin with tab, which is set to three spaces in jGRASP (i.e., your output should use the **escape sequence for a tab**).

Code: This program in about integer division and remainders. All variables should be of type integer. To receive full credit for this assignment, you must calculate the number of each denomination and store the value in a variable. 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 examples 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 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.