Project: Using Java API Classes

Deliverables

Your project files should be submitted to Web-CAT by the due date and time specified. 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:

- MathFormula.java
- HotelBill.java

Specifications

Overview: You will write two programs this week. The first will solve for the result of a specified expression using methods from the Math class, and the second will read data for a hotel bill code and then interpret and print the formatted hotel bill information.

- **MathFormula.java**

  Requirements: Solve for the result of the expression in the formula below for a value x of type double which is read in from the keyboard, and save the result in a variable of the type double. You must use the `pow`, `sqrt`, and `abs` methods of the Math class to perform the calculation. You may use a single assignment statement with a single expression, or you may break the expression into appropriate multiple assignment statements. The latter may easier to debug if you are not getting the correct result.

$$\frac{x^9 + 10}{|5x^3 - 3x^2| + \sqrt{(4x^6 - 2x^2 + 1)}}$$

  Next, determine the number of characters (mostly digits) to the left and to the right of the decimal point in the unformatted result. [Hint: You should consider converting the type *double* result into a String using the static method `Double.toString(result)` and storing it into a String variable. Then, on this String variable invoke the `indexOf(".")` method from the String class to find the position of the period (i.e., decimal point) and the `length()` method to find the length of the String. Knowing the location of the decimal point and the length, you should be able to determine the number of digits on each side of the decimal point.]

  Finally, the result should be printed using the class java.text.DecimalFormat so that to the right of the decimal there are at most three digits and to the left of the decimal each group of three digits is separated by a comma in the traditional way. Also, there should also be at least one digit on each side of the decimal (e.g., 0 should be printed as 0.0). Hint: Use the pattern `"#,##0.0##"` when you create your DecimalFormat object. Make sure you know what this pattern means and how to modify and use it in the future.

Design: Several examples of input/output for the MathFormula program are shown below.

Example #1

Line #	Program output
1	`Enter a value for x: 0`
2	`Result: 10.0`
3	`# digits to left of decimal point: 2`
4	`# digits to right of decimal point: 1`
5	`Formatted Result: 10.0`
6	

Example #2

Line #	Program output
1	`Enter a value for x: 1.0`
2	`Result: 2.94744111674235`
3	`# digits to left of decimal point: 1`
4	`# digits to right of decimal point: 14`
5	`Formatted Result: 2.947`
6	

Example #3

Line #	Program output
1	`Enter a value for x: -12.5`
2	`Result: -526893.3724869725`
3	`# digits to left of decimal point: 7`
4	`# digits to right of decimal point: 10`
5	`Formatted Result: -526,893.372`
6	

Example #4

Line #	Program output
1	`Enter a value for x: 9876.54321`
2	`Result: 1.3260216485287738E23`
3	`# digits to left of decimal point: 1`
4	`# digits to right of decimal point: 19`
5	`Formatted Result: 132,602,164,852,877,380,000,000.0`
6	

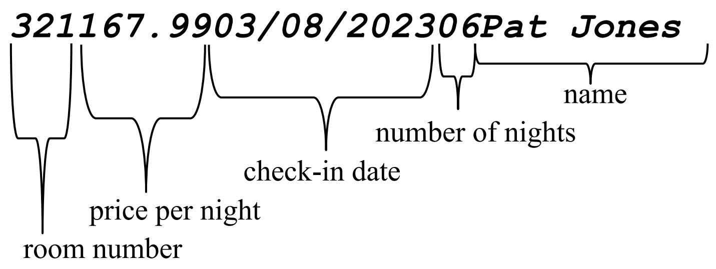
In Example #3, note that the minus sign (-) denoting a negative value should be counted in the number of digits to the left of the decimal (i.e., `-526893.3724869725` has 7 digits to the left). In Example #4, note that when the characters to the right of the decimal in the unformatted result end with E followed by one or more digits, the 'E' should be included in the count of the digits to the right of the decimal point. In this example, E23 indicates an exponent of 23 as in the value preceding the E23 is multiplied by 10^{23}. This usually occurs in unformatted floating point notation for very large or very small values.

Code: To receive full credit for this assignment, you must use the appropriate Java API classes and method to do the calculation and formatting. It is recommended as a practice that you do not modify the input value once it is 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 positive or negative floating-point number.

- **HotelBill.java**

Requirements: The purpose of this program is to accept a hotel bill code as input that includes the room number, price per night, check-in date, and number of nights, followed by the guest's name. The program should then print the hotel bill containing the information read in along with the cost per night price (room and 15 % tax) and the total cost for the stay. The last line of the reservation should contain a "lucky number" between 1 and 999999 inclusive that should always be printed as six digits (e.g., 1 should be printed as 000001). The coded input is formatted as follows:



```
321167.9903/08/202306Pat Jones
```

The entire line with the hotel bill code should be read in at one time, and any whitespace before or after the coded information should be trimmed (e.g., if the user enters spaces before the coded information or after it, the spaces should be disregarded by using the trim method). Your program will need to print the name, room, check-in date, number of nights, cost per night (including room and tax), total for the stay, and a random "lucky" number in the range 1 to 999999. If the user enters a code that does not have at least 22 characters, then an error message should be printed. [The numeric values for price and nights must be provided in the 21 characters; otherwise, a runtime exception may occur.]

Design: Several examples of input/output for the program are shown below.

Example #1

Line #	Program output
1	`Enter hotel bill code: 123456789`
2	
3	`Invalid Hotel Bill Code.`
4	`Hotel Bill Code must have at least 22 characters.`
5	

In the following examples, note that on line 7 there are three spaces (rather than a tab character) after the value for cost per night and after the value for room cost. On line 8, the value for the lucky number is random so it is unlikely it will match your output for the indicated hotel bill code. Each time you run your program, the lucky number should be different than the previous run with the same hotel bill code.

Example #2

Line #	Program output
1	`Enter hotel bill code: 321167.9903/08/202306Pat Jones`
2	
3	`Name: Pat Jones`
4	`Room: 321`
5	`Check-in Date: 03/08/2023`
6	`Nights: 6`
7	`Cost Per Night: $193.19   (Room: $167.99   Tax: $25.20)`
8	`Total: $1,159.13`
9	`Lucky Number: 273693`
10	

Example #3 - Note that the hotel bill code entered below has five leading spaces.

Line #	Program output
1	`Enter hotel bill code:      421199.9904/05/202310Sami Smith`
2	
3	`Name: Sami Smith`
4	`Room: 421`
5	`Check-in Date: 04/05/2023`
6	`Nights: 10`
7	`Cost Per Night: $229.99   (Room: $199.99   Tax: $30.00)`
8	`Total: $2,299.88`
9	`Lucky Number: 226269`
10	

Example #4

Line #	Program output
1	`Enter hotel bill code: 521250.0005/24/202303Jo Abel`
2	
3	`Name: Jo Abel`
4	`Room: 521`
5	`Check-in Date: 05/24/2023`
6	`Nights: 3`
7	`Cost Per Night: $287.50   (Room: $250.00   Tax: $37.50)`
8	`Total: $862.50`
9	`Lucky Number: 688438`
10	

Code: After the entire line with the hotel bill code has been read in and any whitespace before or after the coded information has been trimmed, you must use the appropriate Java API classes and methods to do the extraction of the substrings, conversion of substrings to numeric values as appropriate, and formatting. These include the String methods trim and substring as well as wrapper class methods such as Integer.parseInt and Double.parseDouble which can be used to convert a String representing an integer or floating-point value into its corresponding numeric value. The dollar amounts should be formatted so that both small and large amounts are displayed properly, and the lucky number should be formatted so that six digits are displayed including leading zeroes, if needed, as shown in the examples above. It is recommended as a practice that you do not modify input values once they are stored.

Test: When you run your program, you should be able to copy any of the hotel bill codes above and paste it as the input to your program (rather than entering the code manually). You are responsible for testing your program, and it is important to not rely only on the examples above.

Hints:

1. The coded hotel bill should be read in all at once using the Scanner's nextLine method and stored in a variable of type String. Then the individual values should be extracted using the substring method. The String value for the room price should be converted to type double (using Double.parseDouble) and the String value for number of nights should be converted to type int (using Integer.parseInt) so that these can be used to calculate the Total cost. When printing the values for cost per night, room price, tax, total, and prize number, they should be formatted properly by creating an appropriate DecimalFormat object (see patterns below) and calling its format method.

   To format meal double values, use the pattern `"$#,##0.00"` when you create your DecimalFormat object. The number of nights does not require formatting.

   For prize number, use the pattern `"000000"` when you create your DecimalFormat object.

2. Since only the price and number of nights from the hotel bill code will be used in arithmetic expressions, all other items in the coded hotel bill can be left as type String.

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.