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:

- SolveIt.java
- MarsTicket.java

Specifications

Overview: You will write two programs this week. The first will solve for the result of a specified expression, and the second will read data for a ticket to Mars and then interpret and print the formatted ticket information.

- **SolveIt.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.

$$\boldsymbol{result} = \frac{\boldsymbol{6x^3 + \sqrt{3x^2 + 2x + 1}}}{\boldsymbol{(|2x| + 4)}}$$

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 four 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. However, 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 SolveIt program are shown below.

Line #	Program output
1	Enter a value for x: 0
2	Result: 0.25
3	# of characters to left of decimal point: 1
4	# of characters to right of decimal point: 2
5	Formatted Result: 0.25

Line #	Program output
1	Enter a value for x: 10
2	Result: 250.74651970279868
3	# of characters to left of decimal point: 3
4	# of characters to right of decimal point: 14
5	Formatted Result: 250.747

Line #	Program output
1	Enter a value for x: -12.345
2	Result: -392.72830076799255
3	# of characters to left of decimal point: 4
4	# of characters to right of decimal point: 14
5	Formatted Result: -392.728

Line #	Program output
1	Enter a value for x: 987654321.0
2	Result: 2.9263831674439869E18
3	# of characters to left of decimal point: 1
4	# of characters to right of decimal point: 19
5	Formatted Result: 2,926,383,167,443,986,900.0

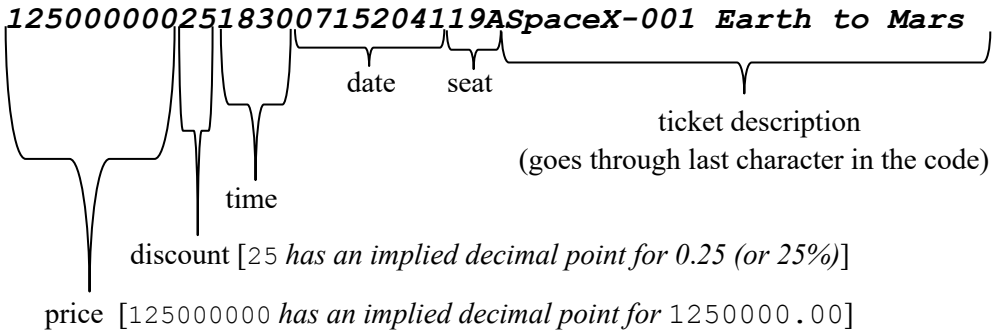
When the characters to the right of the decimal in the unformatted result end with E followed by one or more digits (e.g., E18 indicates an exponent of 18), the 'E' should be included in the count of the characters to the right of the decimal point.

Code: In order 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.

- **MarsTicket.java**

Requirements: The purpose of this program is to accept coded ticket information as input that includes the price, discount, time, date, seat, followed by the description of the ticket. Note that the nine digits for price and two digits for discount have an implied decimal point. The program should then print the ticket information including the actual cost, which is the price with discount applied. The last line of the ticket information should contain a random "prize 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:



Whitespace before or after the coded information should be disregarded (e.g., if the user enters spaces or tabs before or after the coded information, these should be disregarded). Your program will need to print the ticket description, date, time, seat number, price, discount, and cost, and a random prize number in the range 1 to 999999 as shown in the examples below. If the user enters a code that does not have at least 27 characters, then an error message should be printed. [The 27th character of the code is part of the ticket description.]

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

Line #	Program output
1	`Enter ticket code: War Eagle to all on Mars!`
2	
3	`*** Invalid Ticket Code ***`
4	`Ticket code must have at least 27 characters.`

Note that the ticket code entered below results in the indicated output except for the prize number which is random. When more than one item is shown on the same line (e.g., ticket, date, and time on line 3), there are three spaces between them (do not use the tab escape sequence \t).

Line #	Program output
1	`Enter ticket code: 125000000251830071520411 9ASpaceX-001 Earth to Mars`
2	
3	`Ticket: SpaceX-001 Earth to Mars  Date: 07/15/2041  Time: 18:30`
4	`Seat: 19A  Price: $1,250,000.00  Discount: 25%  Cost: $937,500.00`
5	`Prize Number: 867318`
6	

Note that the ticket code entered below includes a 50% discount.

Line #	Program output
1	Enter ticket code: 125000000501830071520411 9ASpaceX-001 Earth to Mars
2	
3	Ticket: SpaceX-001 Earth to Mars Date: 07/15/2041 Time: 18:30
4	Seat: 19A Price: $1,250,000.00 Discount: 50% Cost: $625,000.00
5	Prize Number: 479194
6	

Note that the ticket code below has 10 leading spaces (be sure you are trimming the input code). It also includes a 75% discount.

Line #	Program output
1	Enter ticket code: 125000000751830071520411 9ASpaceX-001 Earth to Mars
2	
3	Ticket: SpaceX-001 Earth to Mars Date: 07/15/2041 Time: 18:30
4	Seat: 19A Price: $1,250,000.00 Discount: 75% Cost: $312,500.00
5	Prize Number: 432239
6	

Code: In order to receive full credit for this assignment, you must use the appropriate Java API classes and methods to trim the input string, to extract the substrings, conversion of substrings of digits to numeric values as appropriate, and formatting. These include the String methods trim, and substring, as well as wrapper class methods such Double.parseDouble which can be used to convert a String of digits into a numeric value for price and discount. The dollar amounts should be formatted so that both small and large amounts are displayed properly, and the prize number should be formatted so that seven digits are displayed including leading zeroes, if needed, as shown in the examples above. It is recommended as a practice that you not modify input values once they are stored.

Test: You are responsible for testing your program, and it is important to not rely only on the examples above. Remember, when entering standard input in the Run I/O window, you can use the up-arrow on the keyboard to get the previous values you have entered. This will avoid having to retype the ticket code each time you run your program.

Hints:

1. The ticket code 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 price and discount should be converted to type double (using Double.parseDouble) so that it can be used to calculate cost. When printing the values for price, discount, cost, and prize number, they should be formatted properly by creating an appropriate DecimalFormat object (see patterns below) and calling its format method.

   To format price and cost, use the pattern `"$#,##0.00"` when you create your DecimalFormat object.

   To format discount, use the pattern `"0%"` when you create your DecimalFormat object.

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

2. Since all items in the ticket code other than the price and discount will not be used in arithmetic expressions, they can and should be left as type String.

3. The time and date should have leading zeros as appropriate. Therefore, these can be printed as String values by concatenating their components with `":"` in the time and `"/"` in the date as needed.

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.