Due: Wednesday, September 14, 2016 by 11:59 PM

Deliverables

Your project files should be submitted to Web-CAT by the due date and time specified above (see the Lab Guidelines for information on submitting project files). In order to avoid a late penalty for the project, you must submit your completed code files to Web-CAT no later than 11:59 PM on the due date for the completed code. You may submit your project up to 24 hours after the due date, but there is a late penalty of 15 points. No projects will be accepted after the one day grace period. If you are unable to submit via Web-CAT, you should e-mail your project Java files in a zip file to your lab instructor before the deadline.

Files to submit to Web-CAT:

- ExpressionEvaluator.java
- GameTicket.java

Specifications

Overview: You will write two programs this week. The first will compute the value generated by a specified expression and the second will read game ticket data and then interpret and print the formatted ticket information.

- **ExpressionEvaluator.java**

Requirements: Calculate the following expression for any value of the variable x, and save the result in a variable of the type double. You must use the sqrt(), abs() and pow() 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.

$$\sqrt{(5x^7 - 4x^6)^2 + \sqrt{|3x^5|}}$$

Next, determine the number of 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 use the *indexOf( )* method from the String class to find the position of the period and the *length( )* method to find the length. 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).

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

Line number	Program output
1	`Enter a value for x: 1`
2	`Result: 1.6528916502810695`
3	`# digits to left of decimal point: 1`
4	`# digits to right of decimal point: 16`
5	`Formatted Result: 1.6529`

Line number	Program output
1	`Enter a value for x: -2.5`
2	`Result: 4028.3224369990567`
3	`# digits to left of decimal point: 4`
4	`# digits to right of decimal point: 13`
5	`Formatted Result: 4,028.3224`

Line number	Program output
1	`Enter a value for x: 5.1`
2	`Result: 378320.1878559609`
3	`# digits to left of decimal point: 6`
4	`# digits to right of decimal point: 10`
5	`Formatted Result: 378,320.1879`

Line number	Program output
1	`Enter a value for x: 1234.567`
2	`Result: 2.184193642567907E22`
3	`# digits to left of decimal point: 1`
4	`# digits to right of decimal point: 18`
5	`Formatted Result: 21,841,936,425,679,070,000,000.0`

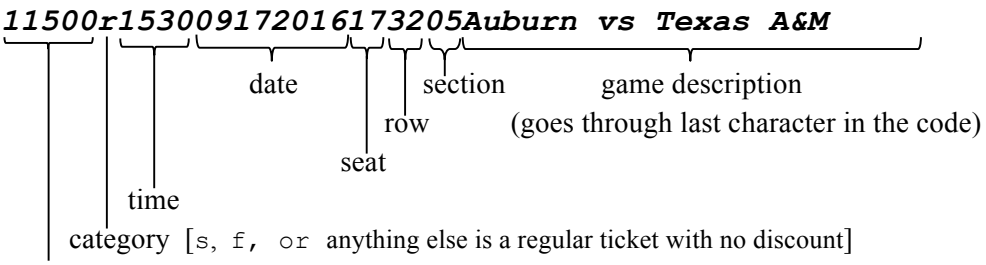
Note that in the example above the digits to the right of the decimal in the unformatted result are followed by E25 (indicating an exponent of 25). For this program, the "E25" should be included in the count of the digits to the right of the decimal point.

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.

- **GameTicket.java**

Requirements: The purpose of this program is to accept coded game ticket information as input that includes the ticket price, category, time, date, seat, row, section, followed by the description of the game. Note that the five digits for price have an implied decimal point. The program should then print the ticket information including the actual cost, which is the price with discount applied (none for regular tickets (`anything but s or f`), 85% for student tickets (`s`), and 25% for faculty/staff tickets (`f`). The last line of the ticket should contain a "prize number" between 1 and 9999999 inclusive that should always be printed as five digits (e.g., 1 should be printed as 0000001). 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 game description, the date and time, the section, row, and seat number, the ticket price, the ticket category, the actual cost, and a random prize number in the range 1 to 9999999. If the user enters a code that does not have at least 25 characters, then an error message should be printed. [The 25th character of the code is part of the game description.]

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

Line #	Program output
1	`Enter your ticket code: 123456789`
2	
3	`Invalid Ticket Code.`
4	`Ticket code must have at least 25 characters.`

Note that the ticket code 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., game, 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 your ticket code: 11500x153009172016173205Auburn vs Texas A&M`
2	
3	`Game: Auburn vs Texas A&M  Date: 09/17/2016  Time: 15:30`
4	`Section: 5  Row: 32  Seat: 17`
5	`Price: $115.00  Category: x  Cost: $115.00`
6	`Prize Number: 4615594`

Line #	Program output
1	Enter your ticket code: 11500s15300917201617320 5Auburn vs Texas A&M
2	
3	Game: Auburn vs Texas A&M Date: 09/17/2016 Time: 15:30
4	Section: 5 Row: 32 Seat: 17
5	Price: $115.00 Category: s Cost: $17.25
6	Prize Number: 4408231

Note that the ticket code below has five leading spaces (be sure you are trimming the input code).

Line #	Program output
1	Enter your ticket code: 11500f15300917201617320 5Auburn vs Texas A&M
2	
3	Game: Auburn vs Texas A&M Date: 09/17/2016 Time: 15:30
4	Section: 5 Row: 32 Seat: 17
5	Price: $115.00 Category: f Cost: $86.25
6	Prize Number: 0647870

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 do the extraction of the category character, extraction of the substrings, conversion of substrings of digits to numeric values as appropriate, and formatting. These include the String methods trim, charAt, and substring, as well as wrapper class methods such as Integer.parseInt and Double.parseDouble which can be used to convert a String of digits into a numeric value. 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. While not a requirement, you should consider making the student discount and faculty/staff discount constants. For example, the following statements could be placed above the main method.

```
static final double STUDENT_DISCOUNT = .85;
static final double FACULTY_STAFF_DISCOUNT = .25;
```

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 info data each time you run your program.

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 have passed Checkstyle. **If you do not submit both programs at the same time, the submission will receive zero points for correctness because Web-CAT will attempt to compile both programs with the JUnit test cases but will fail**. Activity 1 describes how to create a jGRASP project containing both of your files.

Hints

GameTicket class

1. Since char values are primitive types, == and != can be used to compare two values for equality and inequality respectively. The category should be extracted from the input using the String method charAt().

   Note that if category is a String, you should not use == and != to compare two String values. String values should be compared for equality using the String equals method which has a boolean return type. For example, if s1 and s2 are String objects, to check to see if their respective character strings are equal you should use

   ```
   s1.equals(s2)
   ```

   rather than

   ```
   s1 == s2
   ```

   which is only true if s1 and s2 are aliases.

2. Items that should be printed without leading zeros include section, row, seat, price, and cost. The values for section, row, and seat should be converted to type int (using Integer.parseInt) before printing. The values for price, and cost should be converted to type double (using Double.parseDouble) and then formatted properly by creating an appropreate DecimalFormat object and calling the format method.

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