You must complete the textbook sections 1.3 and 1.4 before beginning this assignment.

CS 278 / 465
Programming Assignment 2
Logical Connectives, Tautologies, and Contradictions

Logical Connectives

Logical operators are sometimes called "connectives". You've read about these connectives in the textbook:

¬	negation	"not"
∨	disjunction	"or"
∧	conjunction	"and"
⟶	implication	"implies"

You will need to have the truth tables for these operations readily available.

You will also need to be familiar with the order of operations for logical operators.

Definition of Tautology and Contradiction

A compound proposition is created by connecting individual propositions with logical operations.

When the truth value of a compound proposition is always true, the proposition is a tautology.

When the truth value of a compound proposition is always false, the proposition is a contradiction.

The Big Picture

For this assignment you will write a Java program that contains methods for the following four logical operations:

1) the negation function "not" (denoted ¬)
2) the disjunctive function "or" (denoted ∨)
3) the conjunctive function "and" (denoted ∧)
4) the implicative function "implies" (denoted →)

Next, you will write a test program to evaluate several compound propositions and print their truth tables.

Finally, you will determine if any of those compound propositions are tautologies or if any are contradictions.

Definitions:

Tautology:	a compound proposition whose truth value is always true.
Contradiction:	a compound proposition whose truth value is always false.

Let's Break It Down

Part A: Reusing code from PA 1

Note: You should have the methods not, and, booleanToChar, charToBoolean, and buildTableRows from PA 1. If you did not get them working correctly in PA 1, you must get them working now.

1. Create a new folder to store your work for this assignment.
2. Open the program you created for PA 1.
3. Change the class name to PA2Methods.
4. Use Save As to save the file as PA2Methods.java (in the appropriate folder).
5. Create a new program in the PA 2 folder named PA2Test.java. **CUT** the main method from PA2Methods and **PASTE** it into PA2Test.
6. At this point, PA2Methods should not have a main method in it. PA2Test should have ONLY the main method.
7. Modify the main method in PA2Test so that it will correctly call the buildTableRows method.
8. Compile PA2Test. Debug as needed. Run PA2Test.

Part B: Update the Documentation and Prepare for PA 2

1. Edit the header comments and make them appropriate for PA 2.

   Header comments are required in every program you write for this class. The following must be included in single-line comments (using //) at the top of the file:
   a) your name
   b) the course and the section you are enrolled in
   c) the assignment this program applies to
   d) the date you last modified the program
   e) purpose of the program

2. Review the method documentation that you wrote in PA 1 and improve it as needed.

   Documentation is required for every method. The following must be included in single-line comments (using //) immediately before the header of each method:
   a) purpose of the method
   b) description of parameters
   c) description of the return value

Part C: Write additional static methods

Note: unless specifically instructed otherwise, you are required to put **public** and **static** on all methods.

Also note: for full credit, in each of the following methods you must correctly use the charToBoolean and booleanToChar methods and catch any exception thrown by charToBoolean.

Need help with exception handling? Refer to the Java Bytes provided with PA 1.

1. Add to PA2Methods.java. Write a method named **or** (logical operator ∨) that will accept two parameters. The parameter types are char. Be sure to handle the possible illegal argument exception from charToBoolean.

   The method should return a char value. The return value should be the disjunction of the two parameters.

2. Add to PA2Methods.java. Write a method named **implies** that will accept two parameters. The parameter types are char. Be sure to handle the possible illegal argument exception from charToBoolean.

   The method should return a char value. The return value should be the conditional parameter1 ⟶ parameter2.

Part D: Modify the main method

What you're about to do is evaluate some compound propositions. The truth value of a compound proposition depends on the values of p, q, and r.

You will call buildTableRows only one time.

For each compound proposition, you will print the truth table. Then, print a statement that tells whether the proposition is a tautology, a contradiction, or neither.

Here are the propositions you must evaluate:

```
Proposition 1: (¬ p ⟶ q)  ∧  (r ⟶ p)
Proposition 2: (p ∨ (¬ q)) ∧ (r ∨ ¬ (p ⟶ q))
Proposition 3: p ⟶ ( (¬ (p ∨ ¬ q)) ⟶ (p ∧ q) )
Proposition 4: (p ∧ (p ⟶ q)) ∧ ¬ q
```

1. The main method will print 4 truth tables (see the list of compound propositions on the next page).

   For each truth table, print the headings as shown in this example:

   ```
   p  q  r  Proposition 1
   -- -- -- -------------
   ```

   In the last column print "Proposition 1" for the first truth table, "Proposition 2" for the second truth table, and so on.

   You must follow the order of operations for logical operators and you must use the 4 methods above (not, or, and, implies). (Read ahead to page 5 to see example output.)

2. Using a for-each loop that iterates through the array returned by buildTableRows, write the code to print the truth table for Proposition 1. Verify by hand that the last column of the truth table contains the correct truth values.
3. Debug as needed and run again until you are sure that it works correctly for Proposition 1.
4. Add a count variable before the loop you wrote in step 2. You will count the number of times that the compound proposition's truth value is 'T'.
5. Add some print statements after the loop so that the program will print:
   - number of truth-value combinations of p, q, and r that make the proposition true
   - number of truth-value combinations of p, q, and r that make it false
   - whether it is a tautology, a contradiction, or neither
6. Debug as needed and run again until you are sure that it works correctly for Proposition 1.

Example Output :

Suppose we have a compound proposition called "Proposition 5" that is the logical expression
$(p \vee (p \longrightarrow q)) \longrightarrow \neg q$

You can evaluate the truth value of this proposition using nested method calls:

```
value = PA2Methods.implies(PA2Methods.or(p,PA2Methods.implies(p,
        q)), PA2Methods.not(q));
```

Note that the variables p, q, and r contain characters, either 'T' or 'F' depending on which row of the truth table is being evaluated.

Here's the output of my Proposition 5 truth table:

```
p  q  r  Proposition 5
-- -- -- -------------
T  T  T  F
T  T  F  F
T  F  T  T
T  F  F  T
F  T  T  F
F  T  F  F
F  F  T  T
F  F  F  T

4 combinations result in Proposition 5 being T.
4 combinations result in Proposition 5 being F.
Proposition 5 is neither a tautology nor a contradiction.
```

7. After you have it working for Proposition 1, repeat the process for the other Propositions. Do not create additional counter variables. Just set the counts back to 0.

The output of the finished program must contain the truth tables and summaries for all 4 compound propositions.

Submit PA2Methods.java and PA2Test.java on Canvas.