Comp 271 - Assignment
Compiling Expressions
As in the course syllabus, late assignments will not be accepted unless you have completed the process for using a late pass before the due date.

1. Submit
The zipped project file in Sakai assignment following the correct file naming convention.

2. Format
Java program using proper programming style and with appropriate java doc comments in the code. Code must compile and run in the tools used in class.

3. Content
Summary: Write a complete Java project using correct Object Oriented Programming techniques. The purpose of this assignment work with stacks appropriately from the java collections and to get familiar with the some of the techniques and issues in compiling source code into java. The latter will be useful as we explore how the java virtual machine works with byte code.

Your program, using good OO design and structure, is to

1. Implement expression compiling as described in the text, Chp4, programming project 8, page 204.
2. You are required to use two java data structures from the collections framework and to use the two stack approach described in the text assignment.
3. You must implement at least one class (don’t do it all in main). Use main only for user input and to display the results in a suitable fashion. One approach is an ExpressionCompiler class that takes an expression, returns the resulting code, and indicates if there were syntax errors. You may wish to have more than one class.
4. You decide how to store and display the resulting code shown in the assignment text.
5. You have to detect invalid expressions, for example a+b+ (missing final operand), a b+c (missing operator between a and b).
6. Not required, but interesting, you should generate error messages for invalid expressions that attempt to explain what was wrong with the user input. Doing so will give you an appreciation for java compile error messages.

7. **Extra Credit**. Add parenthesis to the expressions (as options, they work to change the precedence and are optional and can be used only where the user picks, not required everywhere). It becomes much more realistic and really uses the stack much more fully than the basic assignment in the text (think about how deep the stack can ever get with expressions in the base assignment such as \( a + b \times c + d \)). Parenthesis force the expression within them to be compiled completely before whatever is on either side. Think about \( a + b \times (c + d) \) or, even more interesting \( a + b \times (c \times (d + e) / (f + g) - h) \). Each expression inside a pair will get replaced by a temporary result name (z, x, ...) generated as in the base assignment.

Test your program fully and carefully. Make sure the output from the program looks good and is easy to follow. Junit testing is recommended but not required.

4. Hints
   1. The book text assignment is written in paragraph form and may be complex to transfer to code. This is certainly a Think (lots) before Code. I strongly recommend you write careful and precise step by step pseudo code; if you do so, include in your code as comments.
   2. If you want to know some of the background about this approach to compiling expressions, look up Dijkstra’s Shunting Yard Algorithm (like switching train cars); for example: [http://introcs.cs.princeton.edu/java/43stack/Evaluate.java.html](http://introcs.cs.princeton.edu/java/43stack/Evaluate.java.html) and [http://www.oxfordmathcenter.com/drupal7/node/628](http://www.oxfordmathcenter.com/drupal7/node/628)
   3. You *cannot* switch to doing an expression evaluator or a post fix scheme. You *cannot* switch to doing a recursive descent implementation (we will look at later in class) You must compile the expression into some form of operations and operands as shown in the text using two stacks.
   4. Do *NOT* copy code you find. You may use it for background and ideas. There are many solutions out there for this kind of thing. You need to learn to do it on your own!

5. Grading
   WARNING: there are many similar solutions on the web. Don’t copy any of them. You will get zero points and it will be a breach of the course academic integrity policy.
20 points total. No points if will not compile. -5 to -10 for logic errors such as incorrect code generated, missing syntax errors, etc. depending on severity. -15 for doing it all in main without suitable class and object structure. Up to -10 for poor programming style or lack of documentation.