Algorithm for converting an infix expression to postfix:

Get the text from the text field, and split it into tokens:

```
String expression = textfield.getText();
String delims = "+-*/() ";
StringTokenizer strToken = new StringTokenizer(expression, delims, true);
while(strToken.hasMoreTokens()){
      token = strToken.nextToken();
}
```

http://docs.oracle.com/javase/7/docs/api/java/util/StringTokenizer.html

- For each String (token) in the array of Strings
  - If the next token is a number, append it to the result
  - Else if the next token is a left parenthesis, push it on the stack
  - Else if the next token is a right parenthesis,
    - Pop elements off of the stack, append them to the result until the top element of the stack is a matching left parenthesis
    - Pop the left parenthesis off of the stack
  - Else if the next token is an operator (+, -, *, /)
    - Then compare the token to the top of the stack to determine precedence
    - While the operator on the stack has precedence
      - Pop the top element off of the stack and append it to the result
    - Push the current operator on the stack
- While there are elements remaining on the stack
  - Pop the top element off of the stack and append it to the result

Algorithm for Evaluating a Postfix Expression

Tokenize the String

```
import java.util.StringTokenizer;

StringTokenizer token = new StringTokenizer(postfixStr, delims, true);
```

- While there are more tokens
  - Remove the first token from the list
  - If the token is an operand, push it onto the stack
  - Else if the token is an operator, pop the top 2 elements, 1st goes to the right, 2nd to the left, apply the operation, push the result onto the stack
- Pop the remaining element off of the stack and store it in the result