LookaheadScanner.java

```
1  // Copyright 2013 Bill Campbell, Swami Iyer and Bahar Akbal-Delibas
2
3  package jminusminus;
4
5  import java.io.FileNotFoundException;
6  import java.util.Stack;
7  import java.util.Vector;
8
9  /**
10  * A lexical analyzer for j-- that interfaces with the hand-written parser
11  * (Parser.java). It provides a backtracking mechanism, and makes use of the
12  * underlying hand-written Scanner.
13  */
14
15 class LookaheadScanner {
16
17     /** The underlying hand-written scanner. */
18     private Scanner scanner;
19
20     /** Backtracking queue. */
21     private Vector<TokenInfo> backtrackingQueue;
22
23     /** Token queue. */
24     private Vector<TokenInfo> nextQueue;
25
26     /** Stack of token queues for nested lookahead. */
27     private Stack<Vector<TokenInfo>> queueStack;
28
29     /** Whether we are looking ahead. */
30     public boolean isLookingAhead;
31
32     /** Previous token. */
33     private TokenInfo previousToken;
34
35     /** Current token. */
36     private TokenInfo token;
37
38     /**
39      * Construct a LookaheadScanner from a file name.
40      *
41      * @param fileName
42      *            the name of the file containing the source.
43      * @exception FileNotFoundException
44      *                when the named file cannot be found.
45      */
46
47     public LookaheadScanner(String fileName) throws FileNotFoundException {
48         scanner = new Scanner(fileName);
49         backtrackingQueue = new Vector<TokenInfo>();
50         nextQueue = new Vector<TokenInfo>();
51         queueStack = new Stack<Vector<TokenInfo>>();
52         isLookingAhead = false;
53     }
54
55     /**
56      * Scan to the next token in the input.
57      */
58
59     public void next() {
60         previousToken = token;
61         if (backtrackingQueue.size() == 0) {
62             token = scanner.getNextToken();
63         } else {
64             token = backtrackingQueue.remove(0);
65         }
66         if (isLookingAhead) {
```

```
            nextQueue.add(token);
        }
    }

    /**
     * Record the current position in the input, so that we can start looking
     * ahead in the input (and later return to this position). We'll queue up
     * the current and subsequent tokens until returnToPosition() is invoked.
     * These recordPosition's can be nested.
     */

    public void recordPosition() {
        isLookingAhead = true;
        queueStack.push(nextQueue);
        nextQueue = new Vector<TokenInfo>();
        nextQueue.add(previousToken);
        nextQueue.add(token);
    }

    /**
     * Return to the previously recorded position in the input stream of tokens.
     * If this is a nested lookahead, then return to the previous token queue.
     */

    public void returnToPosition() {
        while (backtrackingQueue.size() > 0) {
            nextQueue.add(backtrackingQueue.remove(0));
        }
        backtrackingQueue = nextQueue;
        nextQueue = queueStack.pop();
        isLookingAhead = !(queueStack.empty());

        // Restore previous and current tokens
        previousToken = backtrackingQueue.remove(0);
        token = backtrackingQueue.remove(0);
    }

    /**
     * The currently scanned token.
     *
     * @return the current token.
     */

    public TokenInfo token() {
        return token;
    }

    /**
     * The previously scanned token. We use this in the parser to get at a
     * token's semantic info (for example an identifier's name), after we've
     * scanned it.
     *
     * @return the previous token.
     */

    public TokenInfo previousToken() {
        return previousToken;
    }

    /**
     * Has an error occurred up to now in lexical analysis?
     *
     * @return true or false.
     */

    public boolean errorHasOccured() {
        return scanner.errorHasOccurred();
    }

```

```
136     /**
137      * Return the name of the source file.
138      *
139      * @return name of the source file.
140      */
141
142     public String fileName() {
143         return scanner.fileName();
144     }
145
146 }
147
```