Exercise 7: Processing Nested Bracket Comments

Modify the implementation of the method next() so that, in addition to end-of-line comments, (nested) bracket comments (/* ... /* ... */ ... */) are recognized and filtered out of the token stream.

Solution

There are now three language elements that can begin with '/': the division operator (slash), an end-of-line comment (//... eol) and a bracket comment (/* ... /* ... */ ... */). The next() method must therefore be modified as follows:

```
public static Token next() {
    ...
    switch (ch) {
        ...
        case '/': nextCh();
            if (ch == '/') {  // end-of-line comment
                do nextCh();while (ch != '\n' && ch != eofCh);
                t = next();  // call scanner recursively
            } else if (ch == '*') {  // bracket comment
                skipComment();
                t = next();  // call scanner recursively
            } else t.kind = slash;
            break;
        ...
    }
    return t;
}
```

The method skipComment() has the task of skipping the rest of the bracket comment and also taking nested comments into account:

```
private static void skipComment() {
    // ch == '*'
    nextCh();
    do {
        do {
            if (ch == '/') {                    // possible start of nested comment
                nextCh();
                if (ch == '*') skipComment();   // nested comment
            } else if (ch == '*') {             // possible end of comment
                break;
            } else nextCh();                    // character within comment => skip it
        } while (ch != eofCh);
        nextCh();
    } while (ch != '/' && ch != eofCh);
    nextCh();
    // ch holds the first character after the comment
}
```