Aufgabe 7: Erkennung geschachtelter Klammerkommentare

Ändern Sie die Implementierung der Methode next so, dass neben Zeilenendekommentaren auch (geschachtelte) Klammerkommentare (/* ... /* ... */ ... */) erkannt und aus dem Tokenstrom herausgefiltert werden.

Lösung

Es gibt nun drei Konstrukte, die mit '/' beginnen können: der Divisionsoperator (slash), ein Zeilenende-Kommentar (//... eol) und ein Klammerkommentar (/* ... /* ... */ ... */). Die Methode next muss daher wie folgt geändert werden:

```
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 == '*') {  // bracketed comment
                skipComment();
                t = next();  // call scanner recursively
            } else t.kind = slash;
            break;
        ...
    }
    return t;
}
```

Die Methode skipComment hat die Aufgabe, den Rest des Klammerkommentars zu überlesen und dabei auch geschachtelte Kommentare zu berücksichtigen:

```
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 == first character after comment
}
```