Java Compiler: j--

Griffin Ryan

Winter 2024

Enhancements to the j-- Compiler's Scanner: Implementation and Testing

Abstract

This document details the recent enhancements made to the scanner component of the j-- compiler. These enhancements include the handling of multi-line comments, recognition of a broader set of Java operators and reserved words, and the addition of support for double-precision, float, long, and various integer literals. The aim of these modifications is to bring the scanner closer to Java standards, improving both its functionality and educational utility.

Introduction

The j-- project serves as an educational tool for understanding the intricacies of Java programming and compiler theory. The scanner, a crucial component of the compiler, is responsible for the lexical analysis, converting the input program text into a series of tokens. This document outlines the recent improvements made to the scanner to handle more complex Java features and literals.

Enhancements to the Scanner

Multi-Line Comments

The scanner has been enhanced to recognize and appropriately ignore multi-line comments, denoted by /* and */. This feature allows users to include extensive comments in their j-- programs, making the code more understandable and maintainable.

Java Operators

A significant enhancement to the scanner is the addition of various Java operators. These include arithmetic, logical, and bitwise operators. The scanner can now correctly identify and tokenize these operators, allowing for more complex expressions in j-- programs.

Reserved Words

The scanner's ability to recognize reserved words has been expanded. This update includes all standard Java reserved words, ensuring that j-- remains consistent with Java syntax and preventing these keywords from being used as identifiers.

Double-Precision Literals

Support for double-precision literals, or DOUBLE_LITERAL, has been introduced. The scanner can now distinguish and correctly tokenize literals such as 123.45 and 1.2e3.

Other Literals

The scanner now supports FLOAT_LITERAL, LONG_LITERAL, and various integer representations like hexadecimal (prefixed with 0x), octal (beginning with 0), and binary (prefixed with 0b). This addition enhances the versatility and depth of numerical operations in j--.

Implementation Details

Code Changes

The Scanner.java file underwent significant modifications. These include the addition of new methods and the extension of existing ones to accommodate the recognition of multi-line comments, various operators, reserved words, and different literal types.

Challenges and Solutions

One of the challenges faced was ensuring that the new features did not interfere with the existing scanning logic. This was overcome by carefully structuring the tokenization process and extensively testing the new features for compatibility.

Design Decisions

The design decisions were primarily driven by the goal of aligning j-- more closely with Java standards while maintaining its simplicity and educational focus.

Testing Strategy

Test Overview

A comprehensive test suite was developed to ensure that all new features function correctly and do not disrupt existing functionalities.

Test Cases

Test cases included various scenarios for multi-line comments, each Java operator, reserved words, and different types of literals. Each test was designed to validate the correct identification and tokenization by the scanner.

Results and Analysis

The tests successfully demonstrated the scanner's enhanced capabilities. All new features performed as expected, and no conflicts with existing functionalities were observed.

Conclusion

Summary of Achievements

The enhancements to the j-- compiler's scanner represent a significant step towards a more robust and feature-rich educational tool. These improvements not only extend the scanner's capabilities but also provide users with a better platform for understanding Java programming and compiler construction.

Future Work

Future work could focus on further aligning j-- with the latest Java standards, possibly introducing more complex features like lambda expressions and generics.

6. Appendices

Appendix A: Source Code Listings

Included here the modified Scanner.java file.

Appendix B: Test Files

The complete test file TestScanner.java is provided.