**User Guide for the Mini Java Compiler Project**

**Introduction**

This user guide explains the functionalities and operational procedures of the Mini Java Compiler, built using Python 3 and the PLY library. The project is aimed at analyzing and understanding a predefined subset of the Java programming language, specifically focusing on the lexical and syntactic analysis phases.

**Requirements:**

Programming Language: Python 3

Library: PLY (Python Lex-Yacc)

System Requirements: Minimum of 4 GB RAM and 120 GB of hard drive space, running on Windows 11.

Installation

**Python Installation:**

Ensure Python 3 is installed on your system. You can download it from python.org.

**Library Installation:**

**Install PLY using Python's pip package manager:**

Copy code

pip install ply

Usage

**Lexical Analysis:**

The lexical analyzer (lexer) processes input code into tokens such as keywords, identifiers, operators, delimiters, and literals.

The lex.py module in PLY aids in developing regular expressions that establish the patterns for identifying these tokens.

**Parsing:**

The parser analyzes tokens produced by the lexer employing grammar rules to ascertain the input's syntactic arrangement.

The yacc.py module in PLY defines the grammar utilizing Backus-Naur Form (BNF) notation to construct an Abstract Syntax Tree (AST).

**Executing the Compiler:**

To run the compiler, ensure the source file (e.g., java.txt) is in the same directory as the compiler script.

**Execute the script using the following command:**

python java\_main.py

**The compiler will parse the input file and display errors or the resulting parse tree (AST).**

File Structure

Tokenizer: Handles the breakdown of raw input into recognizable tokens.

Parser: Analyzes token stream to verify adherence to grammar rules and constructs the AST.

ASTNode: Represents nodes within the AST, each corresponding to language constructs.

**Error Handling**

Errors are identified during both the lexical and parsing phases. The lexer flags unrecognized symbols, while the parser flags syntactic errors based on the defined grammar.

Example of Lexical Analysis Output

[{token: "Identifier", value: 'x'}, {token: 'ASSIGN', value: '='}, {token: 'Digits', value: '5.9'}]

Viewing Results

After parsing, the compiler will display the AST or any syntax errors encountered. This output helps in understanding the structure of the parsed code and the accuracy of the compilation process.

Future Enhancements

Future updates may include expanding the subset of Java supported, implementing semantic analysis, and optimizing the grammar and data structures used for the parse tree.

Troubleshooting

Ensure that all required files are in the correct directory.

Check Python and PLY versions if there are issues during installation or runtime.

For syntax errors, ensure that the input Java code adheres to the supported subset of grammar rules defined in the compiler.

Support

For additional help, consult the detailed project documentation provided with the compiler or contact the development team via your academic institution's support channels.

This guide provides a comprehensive overview for users needing to operate the Mini Java Compiler, ensuring effective utilization of its features for educational or testing purposes.