Compiler Construction CS424

Assignment #2

Ammar Ahmad 2020071

Introduction:

This report offers an overview and analysis of the parser developed for the MiniLang programming language, including the construction of an Abstract Syntax Tree (AST). The parser is engineered to analyze the syntax of MiniLang programs and construct a corresponding AST based on the provided tokens.

Grammar:

```
Program         -> StatementList
StatementList   -> Statement StatementList | ε
Statement       -> Assignment | IfStatement | PrintStatement
Assignment      -> Identifier '=' Expression ';'
IfStatement     -> 'if' '(' Expression ')' '{' StatementList '}' ['else' '{'
StatementList '}']
PrintStatement  -> 'print' '(' Expression ')' ';'
Expression      -> Term { ('+' | '-') Term }
Term            -> Factor { ('*' | '/') Factor }
Factor          -> Identifier | IntegerLiteral | BooleanLiteral | '(' Expression ')'
```

Parser Components:

1. **Token Types:**
    - The parser employs an enumeration class **TokenType** to define the various types of tokens recognized by the MiniLang language. These token types encompass integers, booleans, operators, keywords, identifiers, literals, comments, and various delimiters.
2. **Token Class:**
    - The **Token** class represents individual tokens identified by the scanner. Each token object contains a token type and its associated lexeme (value).
3. **Parser Class:**
    - The **Parser** class is tasked with parsing MiniLang programs using the provided tokens. It incorporates methods to parse different constructs of the language, including statements, expressions, and control flow structures.

Parsing Process:

1. **Initialization:**
    - The parser is initialized with a list of tokens obtained from the scanner.
2. **Parsing Methods:**
    - The parser implements methods to parse various components of MiniLang programs:
        - **parse()**: Initiates the parsing process by calling the **program()** method.
        - **program()**: Parses the entire program, consisting of a sequence of statements.
        - **statement_list()**: Parses a list of statements until the end-of-line (EOL) token is encountered, handling comments along the way.

- **statement()**: Parses individual statements, including assignment statements, if-else statements, and print statements.
- **expression()**, **term()**, **factor()**: Parse arithmetic expressions and their components, including operators, terms, and factors.

3. **Abstract Syntax Tree (AST):**
    - As the parser traverses the MiniLang program, it constructs an Abstract Syntax Tree (AST) representing the hierarchical structure of the program.
    - Each node in the AST corresponds to a language construct, such as statements, expressions, or control flow structures.
    - The AST serves as an intermediate representation of the parsed program, facilitating further stages of compilation.
4. **Error Handling:**
    - The parser includes error handling mechanisms to detect and report syntax errors. If an unexpected token is encountered during parsing, a **SyntaxError** is raised with a meaningful error message indicating the expected and found token types.

Execution:

The **main()** function orchestrates the parsing process. It obtains the token types from the scanner, converts them into **Token** objects, and initializes the parser with these tokens.

The parser is then invoked to parse the MiniLang program, and the resulting AST or parsed statements are printed as output.

Screen Shots: