Documentation

link to git:

https://github.com/cs-ubbcluj-ro/lab-work-computer-science-2024-carla-mirea/tree/main/1-Mini-Language-And-Scanner

---

Scanner

My ***Scanner*** implements a lexical analyzer that processes an input file, identifying tokens, constants, and symbols. The class generates a *Program Internal Form (PIF)* for tokens and constants, and a *Symbol Table (ST)* for identifiers.

Attributes:

- ***symbolTable***: An instance of the ***SymbolTable*** class, used to store identifiers, integer constants, and string constants.
- ***PIF***: A list of token entries in the format ***Pair<String, Pair<Integer, Integer>>***, where each entry includes a token type and its position in the symbol table or a placeholder if not applicable.
- ***tokens***: A list containing all possible tokens for easy lookup.
- ***reservedWords***: A list of reserved words to check against user-defined identifiers.
- ***filePath***: A string representing the file path of the source code to be scanned.
- ***program***: The entire source code, loaded as a single string.
- ***currentLine***: Tracks the current line number in the file, used for error reporting.
- ***index***: Keeps the position within the ***program*** string for parsing.

Constructor:

- ***Scanner(String filePath)***:

- Initializes the ***Scanner*** with the file path of the program to scan.
- Loads tokens and reserved words from an external file (***token.in***).
- **Parameters**:
  - ***filePath***: The path to the program file to be analyzed.
- **Throws**: ***IOException*** if the tokens file cannot be read.

Operations:

- ***scan()***:
  - Initiates the scanning process, iterating through the program and identifying tokens, constants, and identifiers.
  - Saves the PIF and ST to output files named ***PIF<filePath>.out*** and ***ST<filePath>.out***, respectively.
  - Prints on screen if an invalid token is found during scanning (the PIF and ST are not generated in this case)
- ***checkIfIdentifier()***:
  - Identifies if the current token is a valid identifier based on its format.
  - Updates the symbol table and PIF accordingly.
  - **Returns**: ***true*** if an identifier is found, ***false*** otherwise.
- ***checkIfIntegerConstant()***:
  - Checks if the current token is a valid integer constant.
  - Prevents invalid integer constants (e.g., integers immediately followed by letters).
  - **Returns**: ***true*** if an integer constant is found, ***false*** otherwise.
- ***checkIfStringConstant()***:
  - Checks if the current token is a valid string constant.
  - Detects unclosed strings and other formatting issues.
  - **Returns**: ***true*** if a string constant is found, ***false*** otherwise.

    - **Throws**: ***Exception*** if the string constant is invalid or improperly formatted.
- ***checkIfToken()***:
    - Matches the current token against a list of predefined tokens and reserved words.
    - **Returns**: ***true*** if a matching token is found, ***false*** otherwise.
- ***checkIfValid(String identifier, String subProgram)***:
    - Checks if an identifier is valid by ensuring it does not overlap with reserved words or keywords.
    - **Returns**: ***true*** if the identifier is valid, ***false*** otherwise.
- ***skipSpacesAndComments()***:
    - Skips whitespace characters and comments, updating the ***currentLine*** as necessary for new lines.
    - Moves the ***index*** to the next relevant character.

*For HashTable and SymbolTable the documentation is the same as for Lab2, excepting that now instead of (Key, Value) pairs I kept just the value, as a second element in this case wouldn't give us any relevant information (in most cases it would have defined the type of an identifier, which we don't need for the lexical scanning).