

Laboratory 4 – Finite Automaton

1. FA.java

- Class Structure
 - **FA** represents a Finite Automaton (FA)
 - Attributes:
 - **ELEM_SEPARATOR**: Separator used for splitting strings
 - **isDeterministic**: Flag indicating if the FA is deterministic
 - **initialState**: Initial state of the FA
 - **states**: List of all states in the FA
 - **alphabet**: List of symbols in the alphabet
 - **finalStates**: List of final states in the FA
 - **transitions**: Map representing transitions between states.
- Methods
 - **FA(String filePath)**: Constructor that initializes the FA using a file
 - **readFromFile(String filePath)**: Reads the FA details and transitions from the file
 - **initializeAutomatonDetails(Scanner scanner)**: Initializes FA details from the scanner
 - **readLineAsList(Scanner scanner)**: Helper method to read a line from the scanner
 - **processTransitions(Scanner scanner)**: Processes transitions from the scanner
 - **processTransitionComponents(String[] transitionComponents)**: Populates the transitions map
 - **isValidTransition(String[] transitionComponents)**: Checks if a transition is valid
 - **checkIfDeterministic()**: Checks if the FA is deterministic based on transitions
 - **writeTransitions()**: Generates a formatted string representing FA transitions
 - **checkSequence(String sequence)**: Checks if a sequence is accepted by the FA

2. HashTable.java

- Class Structure
 - **HashTable** represents a hash table implementation
- Attributes
 - **size**: Size of the hash table
 - **table**: ArrayList implementing the hash table.
- Methods

- **findPositionOfTerm(String elem)**: Finds the position of an element in the hash table
- **hash(String key)**: Computes the hash of an element
- **containsTerm(String elem)**: Checks if the hash table contains an element
- **add(String elem)**: Adds an element to the hash table
- **toString()**: Returns a string representation of the hash table.

3. Main.java

- Class Structure
 - **Main** contains the main method for program execution and user interaction.

4. MyScanner.java

- Class Structure
 - **MyScanner** tokenizes and scans the input program
- Attributes
 - **operators, separators, keywords**: Lists of operators, separators, and keywords
 - **filePath**: File path of the program
 - **symbolTable**: Symbol table instance
 - **pif**: Program Internal Form instance.
- Methods
 - **readFile()**: Reads the content of the file
 - **createListOfProgramsElems()**: Prepares the list of program elements
 - **tokenize(List<String> tokensToBe)**: Tokenizes the program elements
 - **scan()**: Scans the program and performs lexical analysis.

5. Pair.java

- Class Structure
 - **Pair** represents a generic pair of elements
- Attributes
 - **first, second**: First and second elements of the pair.

6. ProgramInternalForm.java

- Class Structure
 - **ProgramInternalForm** maintains the Program Internal Form representation
- Attributes
 - **tokenPositionPair**: List of pairs representing tokens and positions
 - **types**: List of types/categories of tokens
- Methods
 - **add(Pair<String, Pair<Integer, Integer>> pair, Integer type)**: Adds tokens and their types to the PIF
 - **toString()**: Returns a string representation of the PIF.

7. SymbolTable.java

- Class Structure
 - **SymbolTable** manages a symbol table using a hash table
- Attributes
 - **hashTable**: Hash table instance
- Methods
 - **findPositionOfTerm(String term)**: Finds the position of a term in the symbol table
 - **add(String term)**: Adds a term to the symbol table
 - **toString()**: Returns a string representation of the symbol table.

8. FA – BNF

```
<faFile> ::= <states> <newline> <initialState> <newline> <alphabet> <newline> <finalStates>
<newline> <transitions>
<newline> ::= '\n'
<states> ::= <state> | <stateList>
<stateList> ::= <state> | <stateList> | ε
<initialState> ::= <state>
<alphabet> ::= <symbolList>
<symbolList> ::= <symbol> | <symbolList> | ε
<finalStates> ::= <state> <stateList>
<transitions> ::= <transition> <transitionList>
<transitionList> ::= <transition> | <transitionList> | ε
<transition> ::= <state> <symbol> <state>
<state> ::= A | B | ... | Z
<symbol> ::= a | b | ... | z
```