GitHub repo: https://github.com/neniscamaria1/Parser

The changes are on the branch FA.

FA.in: Specification := States Alphabet InitialState FinalStates Transitions

- First line: set of states:
  Alphanumeric := '0' | .. | '9' | alpha
  alpha := 'a' | … | 'z' | 'A' | … | 'Z'
  sep := ";"
  State := alpha{alphanumeric}
  States := State sep {State sep} \n

- Second line: alphabet:
  Alphabet := alpha sep {alpha sep} \n

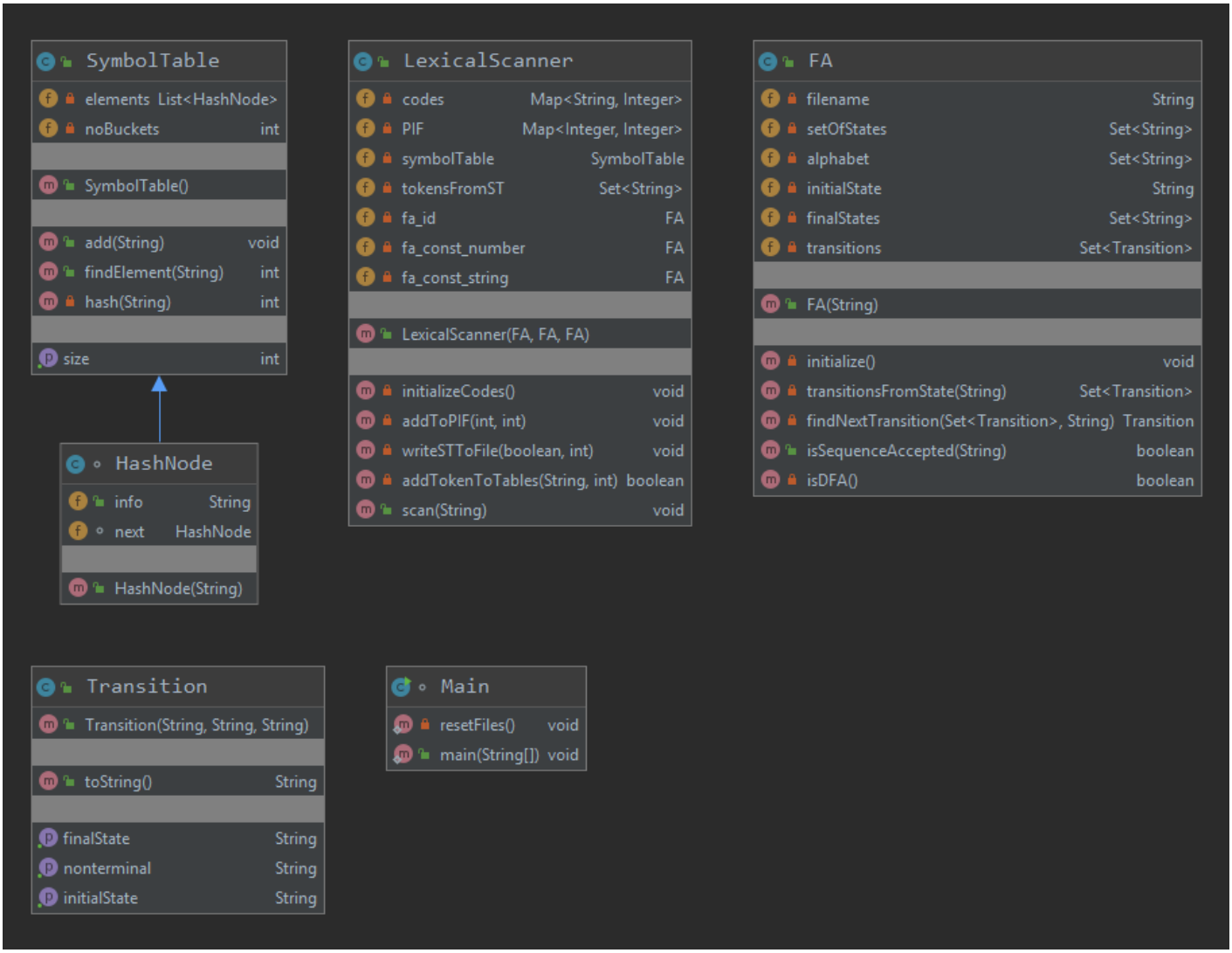
- Third line: initial state
  InitialState := State \n

- Forth line: final states
  FinalStates := State sep {State sep} \n

- The next lines: transitions: on each line a new transition
  Transitions := {State "," alpha "->" State \n}

We added to the LexicalScanner class new attributes:

- fa_id: the FA for identifiers with EBNF:
- fa__const_number: the FA for constant numbers
- fa_const_string: the FA for constant strings



FA class:

- attributes:
  - filename = the file where the FA is defined
  - setOfStates = the set of states of the FA
  - alphabet = the alphabet of the FA
  - initialState = the initial state of the FA
  - finalStates = the final states of the FA
  - transitions = the transitions of the FA
- methods:
  - the constructor receives as a string the filename and initializes the rest of the attributes by calling the method initialize
  - initialize:

        - PRE:-
        - POST:- the attributes are initializes with values from the filename
        - THROWS: -
        - DESCR: It reads the file line by line and initializes the corresponding attribute. If an error happened, a message will be displayed
    - transitionsFromState:
        - PRE: - state = string = the state from which we search for transitions
        - POST:- returns a set of transitions that have as initial state state. It can be empty, but not null.
        - THROWS:-
    - findNextTransition:
        - PRE:
            - Trans = the set of transitions among which we will search
            - Nonterminal = string = the nonterminal for which we search in the set
        - POST: returns the transition found with the given nonterminal, null if it was not found
    - isSequenceAccepted:
        - PRE: sequence = string = the sequence we verify
        - POST: return true if the sequence is accepted, false otherwise
        - DESCR:
            - First it checks if we have a DFA. If it is not, we return false.
            - Then we go through the transitions starting from the initial state and search for a transition with the first nonterminal from the given sequence. If no transitions were found, we return false.
            - Starting with the found transitions, we consider as initial state the initial state of this transitions and repeat the step above and this step until the sequence is empty. If we reached the empty sequence, we return true.
    - isDFA:
        - PRE:-
        - POST:-returns true if the FA is a DFA, false otherwise
        - DESCR: it will search in the transitions read from file if there is any transition with the same initial state and nonterminal. If there exists such pair, we return false and true otherwise.

Transition class:

- Attributes:
    - initalState = string
    - nonterminal = string

    - finalState = string
- methods:
    - constructor – receives all the attributes and initializes them with these values
    - the next 3 methods are getters on the attributes
    - the final method, is toString method used for displaying the transitions