Finite automaton documentation

1. **Objective:**

   A program that can interpret the components of a finite automaton from an input file, check if the automaton is a DFA, and if it is, then be able to check the validity of sequences given through console input.

2. **Input file EBNF:**

   digit ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

   letter ::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"

   state ::= {letter | digit}

   set_of_states ::= {state ";"} state

   alphabet_element ::= letter | digit

   alphabet ::= {alphabet_element ";"} alphabet_element

   transition ::= "d{" state alphabet_element "} = " state

   transitions ::= {transition ";"} transition

   FA.in ::=

   "Set of states: " set_of_states

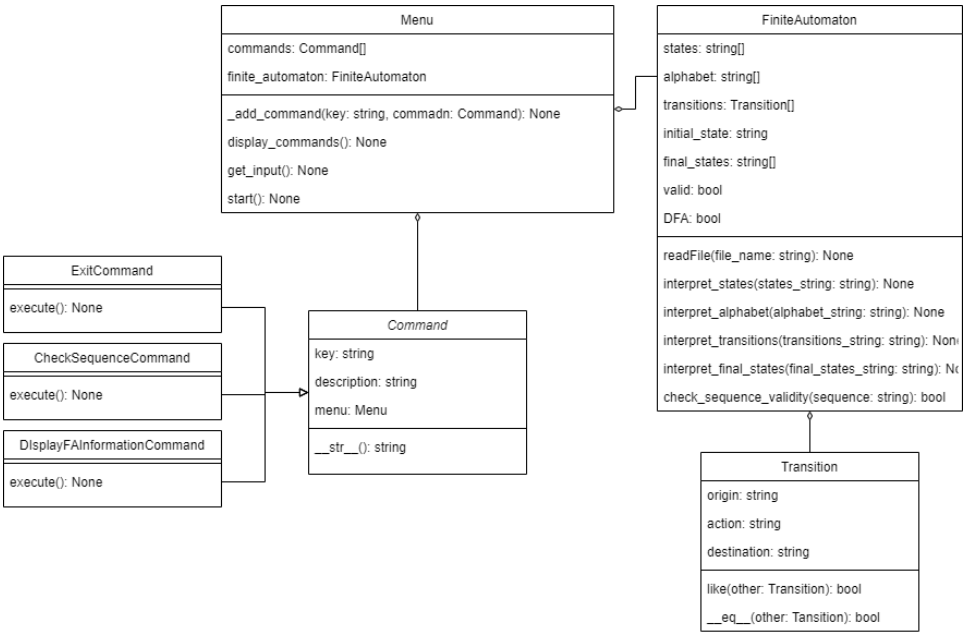
   "Alphabet: " alphabet

   "Transitions: " transitions

   "Final states: " set_of_states

3. **System documentation:**

   https://github.com/Bizso96/Formal-Languages-FA/tree/master



Class FiniteAutomaton – represents a finite automaton

Functions:

- readfile(file_name) - called when the class is initialized. Reads the finite automaton data from the input file, calls the interpret_… functions for the corresponding lines
- interpret_states(states_string) - parses the received string that should contain states separated by semicolons. The first state is considered the initial state
- interpret_alphabet(states_alphabet) - parses the received string that should contain alphabet elements separated by semicolons
- interpret_transitions(transitions_string) - parses the received string that should contain alphabet elments separated by semicolons. Transitions should be in the format d(q1, a) = q2 where q1 represents the origin state, q1 represents the final state and a represents the action. If any of these elements are not found in the list of states or alphabet, then an exception is raised, and the finite automaton's validity is set to False. If two transitions with the same origin and action are found, then the FA is not a DFA and the corresponding boolean is set to False
- interpret_final_states(final_states_string) - parses the received string that should contain final states separated by semicolons. If a final state does not appear in the states list, an exception is raised, and the finite automaton's validity is set to False
- check_sequence_validity(sequence) - if the FA is not a DFA, then a message is displayed. Otherwise, the function checks if the give sequence is valid in the context of the FA. The actions appearing in the sequence must be in a valid order and the last state reached must be a final state

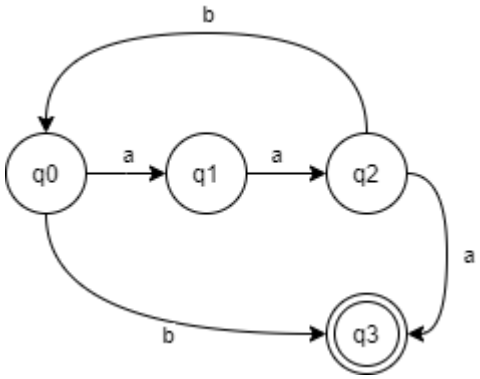
Class Transition – represents a transition

Functions:

- like(transitions) - compares the transition to another transition and if their origin and action is the same, returns True. Otherwise returns False

4. Examples:

For the examples, the following FA is considered:



The FA as it appears in the program:

```
States: q0, q1, q2, q3,
Alphabet: a, b,
Transitions: d(q0, a) = q1, d(q0, b) = q3, d(q1, a) = q2, d(q2, a) = q3, d(q2, b) = q0,
Final states: q3,
DFA: True
```

When reading valid sequences:

```
>check
Sequence: aaa
Current state: q0
Transition: d(q0, a) = q1
Current state: q1
Transition: d(q1, a) = q2
Current state: q2
Transition: d(q2, a) = q3
Valid sequence
```

```
>check
Sequence: aabb
Current state: q0
Transition: d(q0, a) = q1
Current state: q1
Transition: d(q1, a) = q2
Current state: q2
Transition: d(q2, b) = q0
Current state: q0
Transition: d(q0, b) = q3
Valid sequence
```

When reading invalid sequences:

```
>check
Sequence: aaaa
Current state: q0
Transition: d(q0, a) = q1
Current state: q1
Transition: d(q1, a) = q2
Current state: q2
Transition: d(q2, a) = q3
Current state: q3
Invalid sequence
```

After reaching state q3 the checking stops because there is no transition having the origin q3 and action a

```
>check
Sequence: aa
Current state: q0
Transition: d(q0, a) = q1
Current state: q1
Transition: d(q1, a) = q2
Invalid sequence
```

The last state reached is not a final state

5. **UI documentation:**

When launching the program, a console is started. The following commands are available

- exit – exit the application
- help – display the available commands
- fa – display information about the finite automaton
- check – check the validity of a sequence. After using the command, you will be prompted to enter the sequence you want analyzed