DOCUMENTATION: FORMAL LANGUAGES AND COMPILER DESIGN
LAB 4

HTTPS://GITHUB.COM/MPELAE9/FORMAL-LANGUAGES-AND-COMPILER-DESIGN

The project comprises three classes and a .in file, which serves as the input for the Finite Automata (FA). This file includes information on the set of states, the alphabet, a list of transitions (consisting of initial state, value, and final state), the final states, and the initial state.

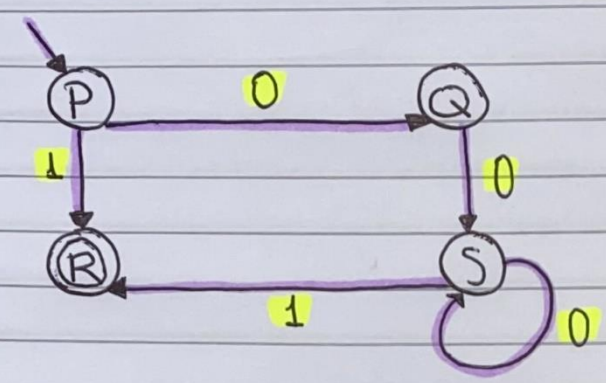
Referring to the Java classes, there's the Main class responsible for displaying the menu, handling user-input options, and presenting the resulting operation outcomes.

The Transition class encompasses the properties: initial state, value, and a list of final states. Its methods primarily consist of setters, getters, and the toString method.

The FiniteAutomata class contains five properties: the initial state, the transitions list, the final states list, the alphabet, and the states list. Its methods include necessary setters and getters along with readFile() to extract FA.in file information. Additionally, it holds a private method, nextState(), which, given an initial state and a transition value, identifies the final state by searching within the transitions list. It also features the public method AfIsDFA () to assess whether the finite automata is deterministic (if it is DFA or not). Lastly, the isAccepted() method accepts a sequence and, by evaluating transition by transition, determines if, upon completion of the sequence, the FA resides in a final state or not.

I have defined this Finite Automata (page 2) which is also reflected in the FA.in to demonstrate the correct functioning of the source code. It is explained to understand the result of what will be printed by the console.

There is also an UML class diagram (page 3) for a better understanding of the Project.



$FA = (Q, \Sigma, \delta, q_0, F)$

$Q = \{P, Q, R, S\}$ state> : list < String >

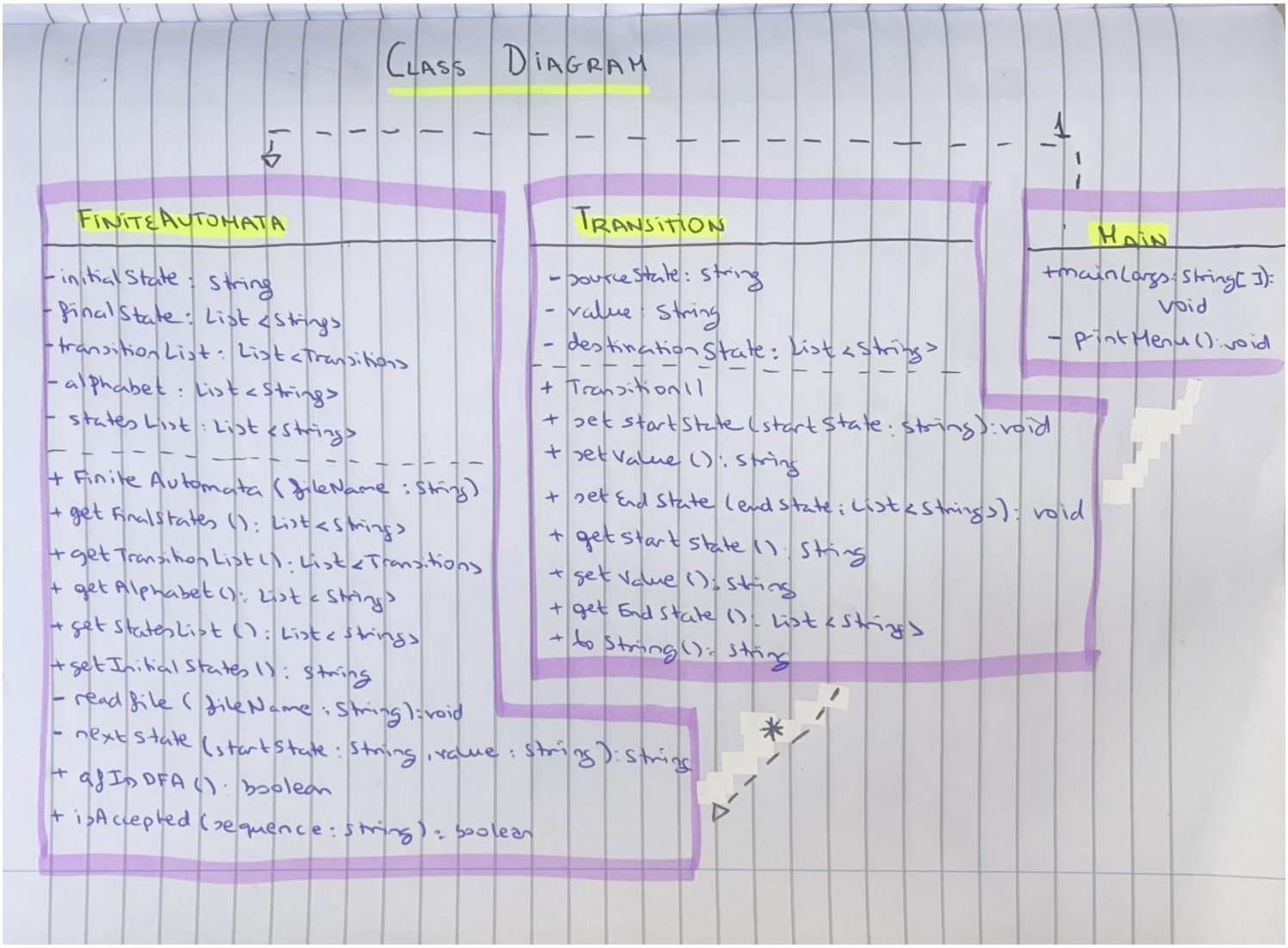
$\Sigma = \{0, 1\}$ alphabet : list < String >

$\delta(P, 0) = Q$
$\delta(Q, 0) = S$
$\delta(S, 0) = S$
$\delta(S, 1) = R$
$\delta(P, 1) = R$

TRANSITIONS {
source : P
value : 0
destination : Q

INITIAL STATE → $q_0 = P$

$F = \{R\}$ → finalState (list < String >)



- The association from *Main* to *FiniteAutomata* implies a dependency relationship, where *Main* relies on *FiniteAutomata* for its functionality. The 1 means that *Main* class has a single instance of *FiniteAutomata* class.

- The association from *Transition* to *FinateAutomata* represents that *FiniteAutomata* contains and uses a collection of Transition instances as a part of its structure. The *FiniteAutomata* class holds a list of Transition objects (*transitionsList*). The * shows that *FiniteAutomata* can have multiple instances of *Transition*.