https://github.com/ComanacDragos/ToyLanguageCompiler

Statement:

Write a program that:

1. Reads the elements of a FA (from file)

2. Displays the elements of a finite automata, using a menu: the set of states, the alphabet, all the transitions, the set of final states.

3. For a DFA, verify if a sequence is accepted by the FA.

Deliverables:

1. FA.in - input file (*on Github*)
2. Source code (*on Github*)
3. Documentation. It should also include in BNF or EBNF format the form in which the FA.in file should be written (*on Moodle and Github*)

FA.in structure:

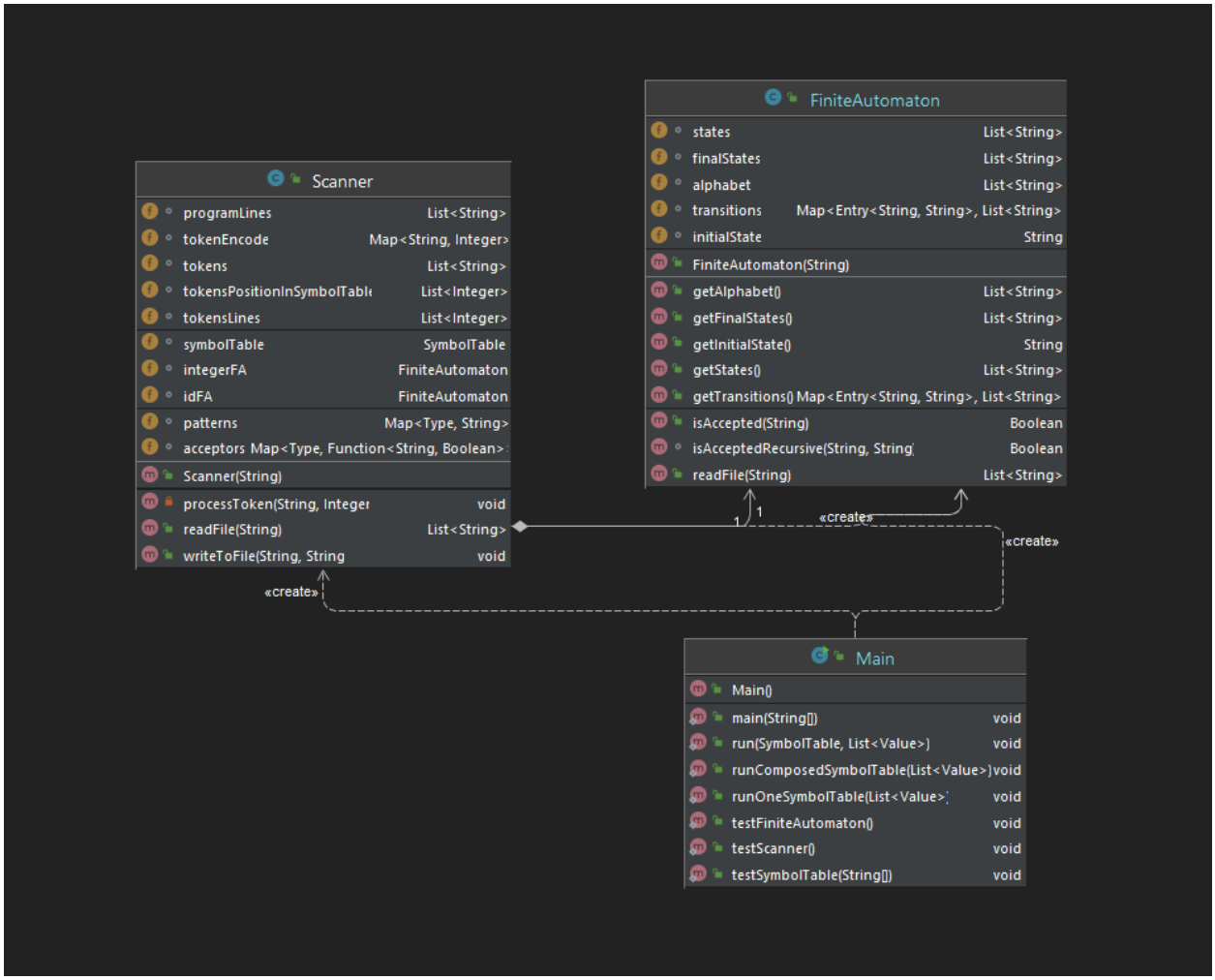
id ::= letter{letter|digit|symbol}
letter ::= "a"|"b"|...|"z"|"A"|...|"Z"
digit ::= "0"|"1"|"2"|...|"9"
symbols ::= "_"
character = letter|digit|symbol|">"|"<"|"="|"!"|"-"|"+"|"*"|"/"|"%"|";"|"^"|","|"}"|"{"|"("|")"|"["|"]"

fa ::= states"\n"alphabet"\n"initial_state"\n"final_states"\n"transitions

states ::= id | id","id
alphabet ::= character | character","character
initial_state ::= id
final_states ::= id | id","id

transitions ::= transition | transition"\n"transition
transition ::= id","set_of_chars","set_of_transitions
set_of_chars ::= character | character{"."character}
set_of_transitions ::= id | id"."id

Implementation:




```
class FiniteAutomaton
    //states of the FA
    List<String> states;
    //final states of the FA
    List<String> finalStates;
    //the alphabet
    List<String> alphabet;
    //Map of transitions: the key is composed of the start state and the character
    // and the value is a list of the destination states
   Map<Map.Entry<String, String>, List<String>> transitions = new HashMap<>();

    //initial state
    String initialState;


//Constructor that reads the FA from a file
public FiniteAutomaton(String file)
```

//Wrapper that calls the recursive function that accepts the sequence with the initial state
public Boolean isAccepted(String sequence)

//Recursive function that accepts the sequence
//if the sequence is empty and the current state is in the final states it returns true, otherwise false
//if there is no transition available from the current state then returns false

//if there are transitions available it checks recursive all possible transitions available with the given char
Boolean isAcceptedRecursive(String sequence, String currentState)

class Scanner

//similar to patterns but each value is a function that returns true if the string is of the type in the key

//replaces the patterns map in the implementation of the scanner

```
Map<Type, Function<String, Boolean>> acceptors = Map.ofEntries(
    new AbstractMap.SimpleEntry<Type, Function<String, Boolean>>(Type.ID, (s)->
idFA.isAccepted(s)),
    new AbstractMap.SimpleEntry<Type, Function<String, Boolean>>(Type.INT, (s)->
integerFA.isAccepted(s)),
    new AbstractMap.SimpleEntry<Type, Function<String, Boolean>>(Type.CHAR, (s)-
>s.matches("^'[a-zA-Z0-9_]'$")),
    new AbstractMap.SimpleEntry<Type, Function<String, Boolean>>(Type.BOOL, (s)-
>s.matches("^true|false$")),
    new AbstractMap.SimpleEntry<Type, Function<String, Boolean>>(Type.STRING, (s)-
>s.matches("^\"[a-zA-Z0-9_\\s]*\"$")),
    new AbstractMap.SimpleEntry<Type, Function<String, Boolean>>(Type.FLOAT, (s)-
>s.matches("^[+-]?(([1-9][0-9]*)|0)\\.([0-9][0-9]*)$"))
);
```

Testing

FA_id.in

```
q0,q1
a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,
0,1,2,3,4,5,6,7,8,9,_
q0
q1
q0,a.b.c.d.e.f.g.h.i.j.k.l.m.n.o.p.q.r.s.t.u.v.w.x.y.z.A.B.C.D.E.F.G.H.I.J.K.L.M.N.O.P.Q.R.S.T.U.V.W.X.Y
```

.Z,q1
q1,a.b.c.d.e.f.g.h.i.j.k.l.m.n.o.p.q.r.s.t.u.v.w.x.y.z.A.B.C.D.E.F.G.H.I.J.K.L.M.N.O.P.Q.R.S.T.U.V.W.X.Y
.Z.0.1.2.3.4.5.6.7.8.9._,q1

FA_integer.in

q0,q1,q2,q3
+,-,0,1,2,3,4,5,6,7,8,9
q0
q1,q3
q0,0,q1
q0,+.-,q2
q0,1.2.3.4.5.6.7.8.9,q3
q2,1.2.3.4.5.6.7.8.9,q3
q3,0.1.2.3.4.5.6.7.8.9,q3