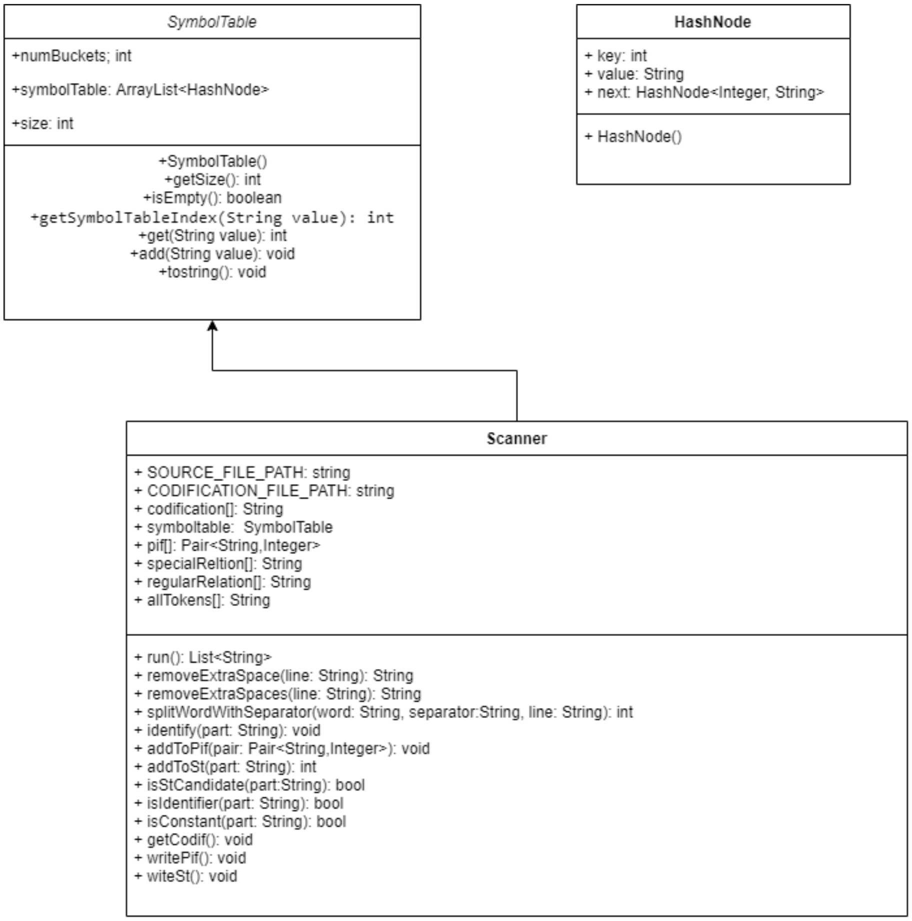
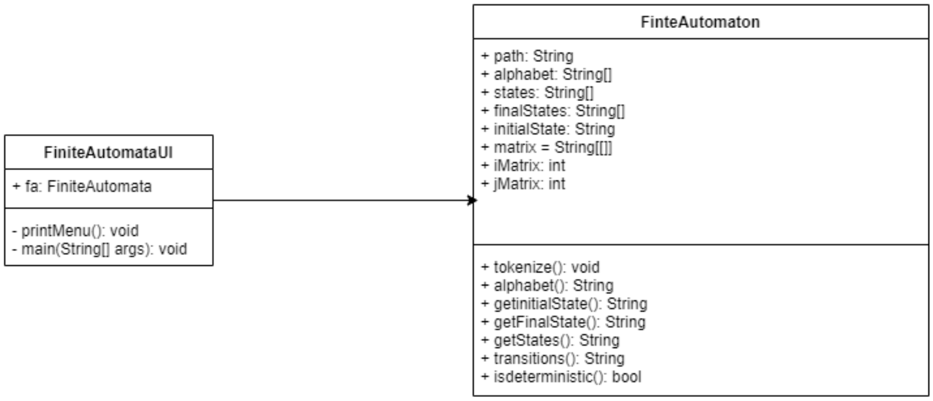
Git source: https://github.com/teodoradra/FLCD-lab/tree/lab4


SymbolTable
+numBuckets; int
+symbolTable: ArrayList<HashNode>
+size: int
+SymbolTable()
+getSize(): int
+isEmpty(): boolean
+getSymbolTableIndex(String value): int
+get(String value): int
+add(String value): void
+tostring(): void
HashNode
+ key: int
+ value: String
+ next: HashNode<Integer, String>
+ HashNode()
Scanner
+ SOURCE_FILE_PATH: string
+ CODIFICATION_FILE_PATH: string
+ codification[]: String
+ symboltable: SymbolTable
+ pif[]: Pair<String,Integer>
+ specialReltion[]: String
+ regularRelation[]: String
+ allTokens[]: String
+ run(): List<String>
+ removeExtraSpace(line: String): String
+ removeExtraSpaces(line: String): String
+ splitWordWithSeparator(word: String, separator:String, line: String): int
+ identify(part: String): void
+ addToPif(pair: Pair<String,Integer>): void
+ addToSt(part: String): int
+ isStCandidate(part:String): bool
+ isIdentifier(part: String): bool
+ isConstant(part: String): bool
+ getCodif(): void
+ writePif(): void
+ witeSt(): void



FA.in

```
     + - 0 1 2 3 4 5 6 7 8 9
-> p q q r s s s s s s s s
   q ε ε ε s s s s s s s s
*  s ε ε ε s s s s s s s s
*  r ε ε ε ε ε ε ε ε ε ε ε ε
```

“->” initial state

“*” final state

“ε” empty set

It is represented as a matrix. First line are the elements of the alphabet. The next lines are initial/final states and result states.

EBNF:

```
input = alphabet “\n” {space} Lines
Character = “a” | … | “z” | “A” | … | “Z” | 0 | .. | 9
alphabet = character | (character Space {Space} stateLine
Space = “ “
State = (letter | digit | “_”) | (letter | digit | “_” ) state
Lines = StateLine | stateLine “\n” {Space} Lines
StateLine = [“->”]{Space}[“*”]{Space}States
States = (Character | stateSet) Space {Space} States | State | stateSet
StateSet = state | state “,” {Space} StateSet
```

FiniteAutomata:

```
% pre:
% post: file is valid
private void tokenize()
```

adds all symbols in alphabet. Computes all transitions between states, symbols with a resultState.

```
% pre:
% post:
public String transitions()
```

parses the matrix to print all the transitions found in FA.in

```
% pre:
% post: true if is deterministic, false otherwise
public boolean isDeterministic()
```

Determins is a state and symbol has more than one result state.
Returns false if it has, false otherwise.

```
% pre:
% post: true if corresponds to FA, false otherwise
public boolean isOk(String str)
```

Checks if a string matches the automaton build previous. Takes every letter with corresponding state and checks if is different than empty set until the end of the string. If it is empty set return false.

FiniteAutomataUI:

```
% pre:
% post:
public static void printMenu()
```

Prints all possible actions.

%pre:
%post:

```
public static void main(String[] args)
```

Begins the execution.