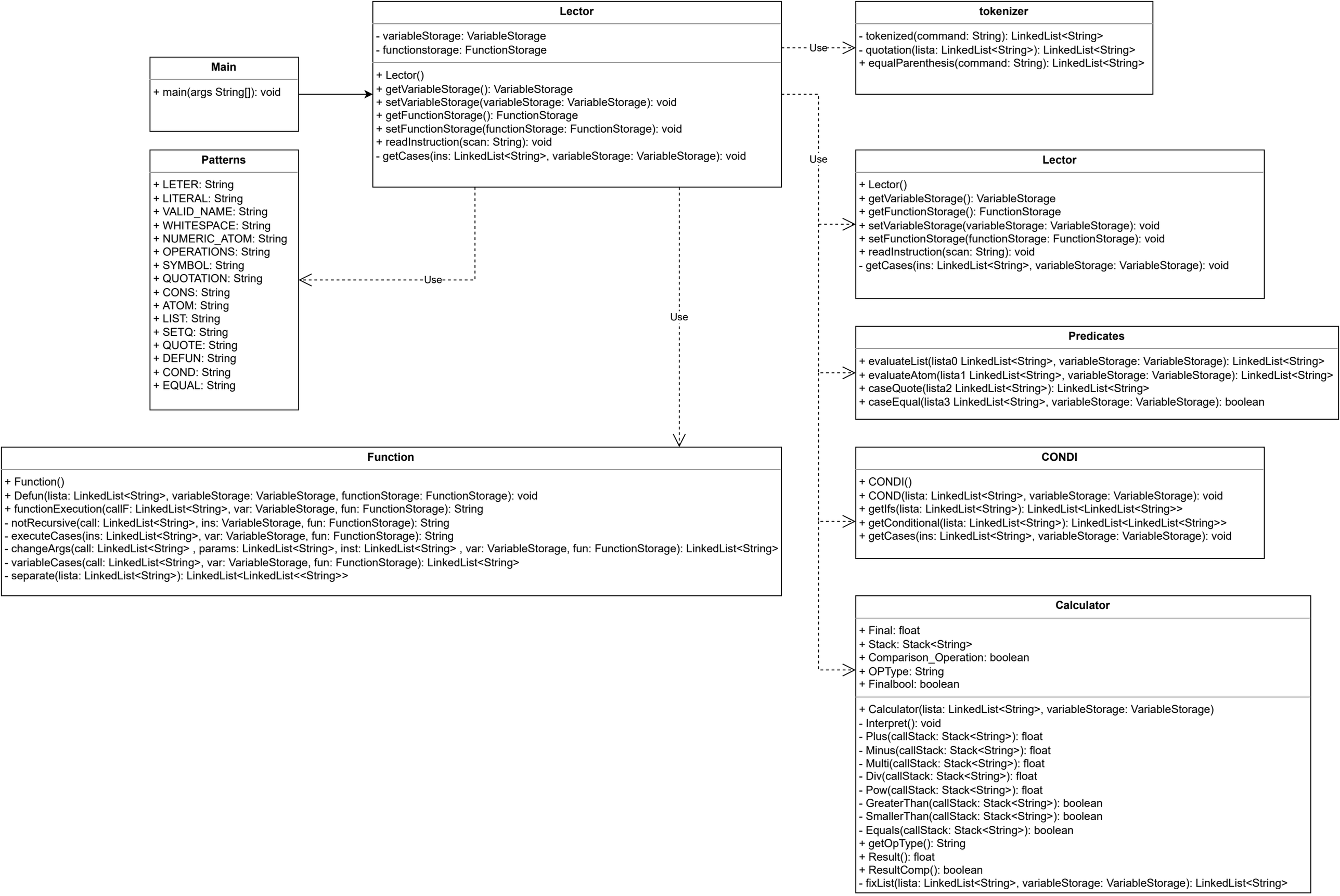
Lector
- variableStorage: VariableStorage
- functionstorage: FunctionStorage
+ Lector()
+ getVariableStorage(): VariableStorage
+ setVariableStorage(variableStorage: VariableStorage): void
+ getFunctionStorage(): FunctionStorage
+ setFunctionStorage(functionStorage: FunctionStorage): void
+ readInstruction(scan: String): void
- getCases(ins: LinkedList<String>, variableStorage: VariableStorage): void
tokenizer
- tokenized(command: String): LinkedList<String>
- quotation(lista: LinkedList<String>): LinkedList<String>
+ equalParenthesis(command: String): LinkedList<String>
Use
Main
+ main(args String[]): void
Patterns
+ LETER: String
+ LITERAL: String
+ VALID_NAME: String
+ WHITESPACE: String
+ NUMERIC_ATOM: String
+ OPERATIONS: String
+ SYMBOL: String
+ QUOTATION: String
+ CONS: String
+ ATOM: String
+ LIST: String
+ SETQ: String
+ QUOTE: String
+ DEFUN: String
+ COND: String
+ EQUAL: String
Use
Use
Use
Lector
+ Lector()
+ getVariableStorage(): VariableStorage
+ getFunctionStorage(): FunctionStorage
+ setVariableStorage(variableStorage: VariableStorage): void
+ setFunctionStorage(functionStorage: FunctionStorage): void
+ readInstruction(scan: String): void
- getCases(ins: LinkedList<String>, variableStorage: VariableStorage): void
Predicates
+ evaluateList(lista0 LinkedList<String>, variableStorage: VariableStorage): LinkedList<String>
+ evaluateAtom(lista1 LinkedList<String>, variableStorage: VariableStorage): LinkedList<String>
+ caseQuote(lista2 LinkedList<String>): LinkedList<String>
+ caseEqual(lista3 LinkedList<String>, variableStorage: VariableStorage): boolean
Function
+ Function()
+ Defun(lista: LinkedList<String>, variableStorage: VariableStorage, functionStorage: FunctionStorage): void
+ functionExecution(callF: LinkedList<String>, var: VariableStorage, fun: FunctionStorage): String
- notRecursive(call: LinkedList<String>, ins: VariableStorage, fun: FunctionStorage): String
- executeCases(ins: LinkedList<String>, var: VariableStorage, fun: FunctionStorage): String
- changeArgs(call: LinkedList<String> , params: LinkedList<String>, inst: LinkedList<String> , var: VariableStorage, fun: FunctionStorage): LinkedList<String>
- variableCases(call: LinkedList<String>, var: VariableStorage, fun: FunctionStorage): LinkedList<String>
- separate(lista: LinkedList<String>): LinkedList<LinkedList<<String>>
CONDI
+ CONDI()
+ COND(lista: LinkedList<String>, variableStorage: VariableStorage): void
+ getIfs(lista: LinkedList<String>): LinkedList<LinkedList<String>>
+ getConditional(lista: LinkedList<String>): LinkedList<LinkedList<String>>
+ getCases(ins: LinkedList<String>, variableStorage: VariableStorage): void
Calculator
+ Final: float
+ Stack: Stack<String>
+ Comparison_Operation: boolean
+ OPType: String
+ Finalbool: boolean
+ Calculator(lista: LinkedList<String>, variableStorage: VariableStorage)
- Interpret(): void
- Plus(callStack: Stack<String>): float
- Minus(callStack: Stack<String>): float
- Multi(callStack: Stack<String>): float
- Div(callStack: Stack<String>): float
- Pow(callStack: Stack<String>): float
- GreaterThan(callStack: Stack<String>): boolean
- SmallerThan(callStack: Stack<String>): boolean
- Equals(callStack: Stack<String>): boolean
+ getOpType(): String
+ Result(): float
+ ResultComp(): boolean
- fixList(lista: LinkedList<String>, variableStorage: VariableStorage): LinkedList<String>