CS 303

Project 1A

Lia Vang

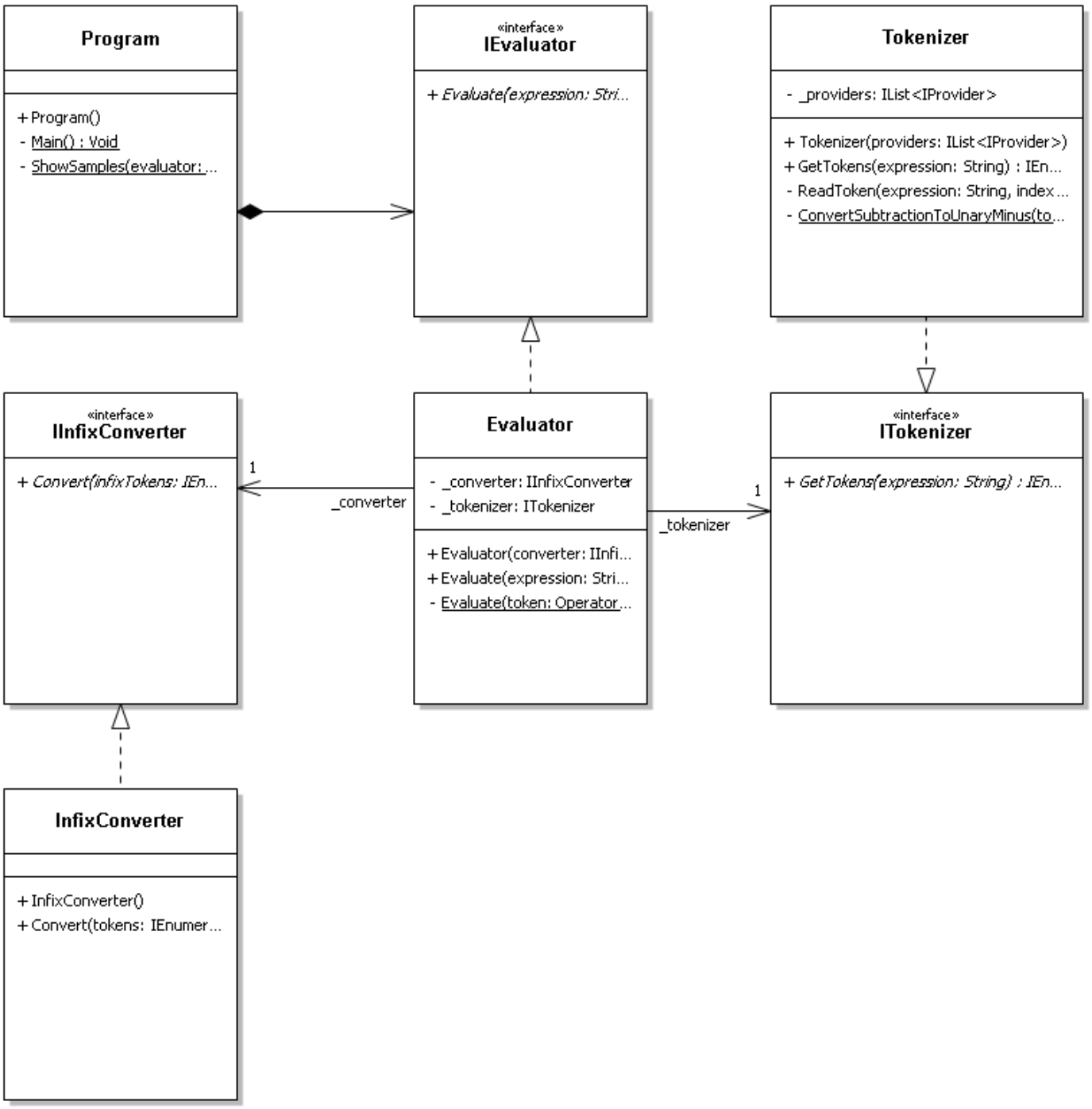
February 28, 2017

Assumptions:

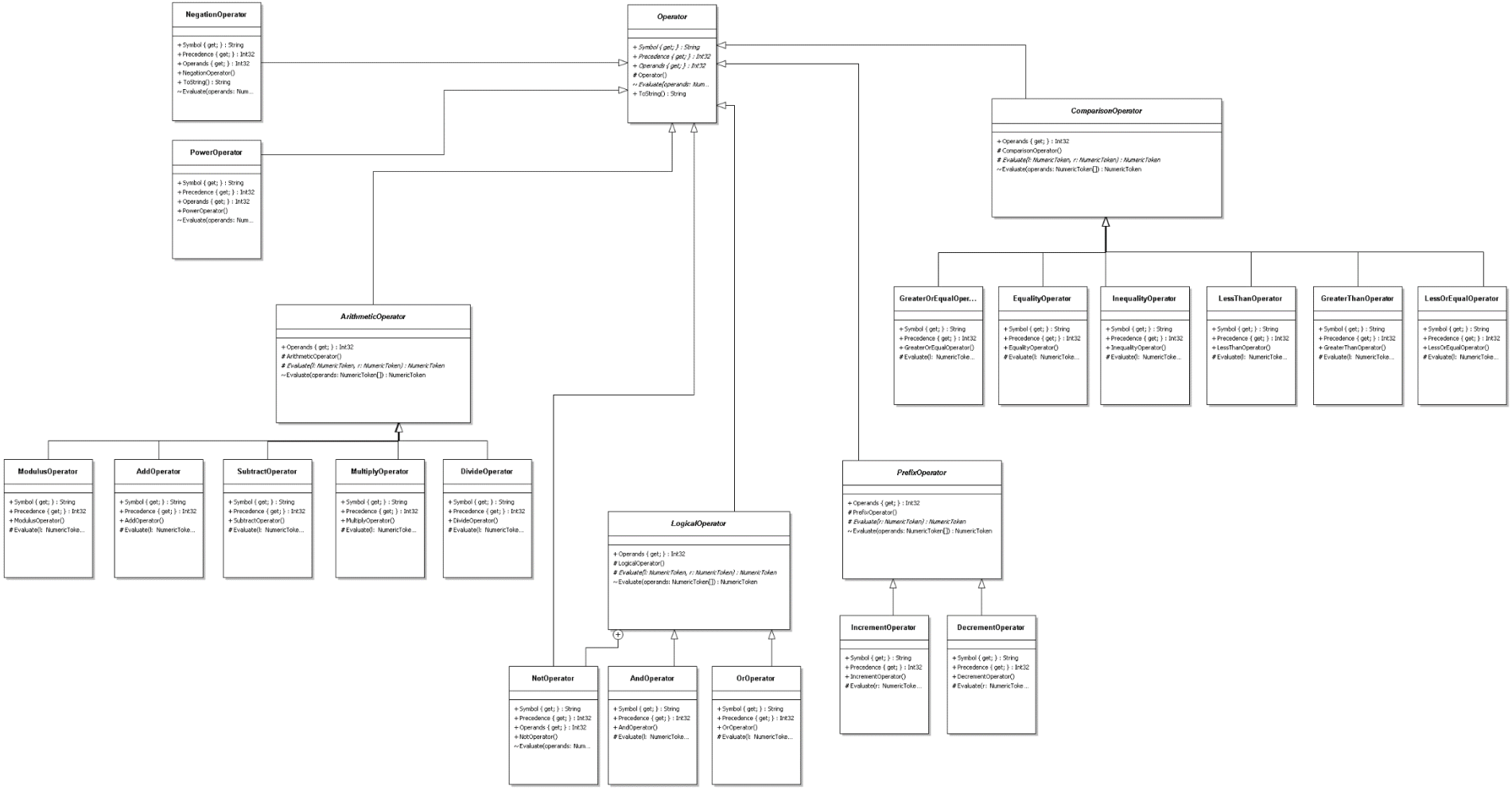
- User will not enter large numbers greater than 2^{32}-1 or smaller than -2^{32}
- Intermediate values will not be greater than 2^{32}-1 or smaller than -2^{32}
    - Large values result in an evaluation error

UML Diagrams

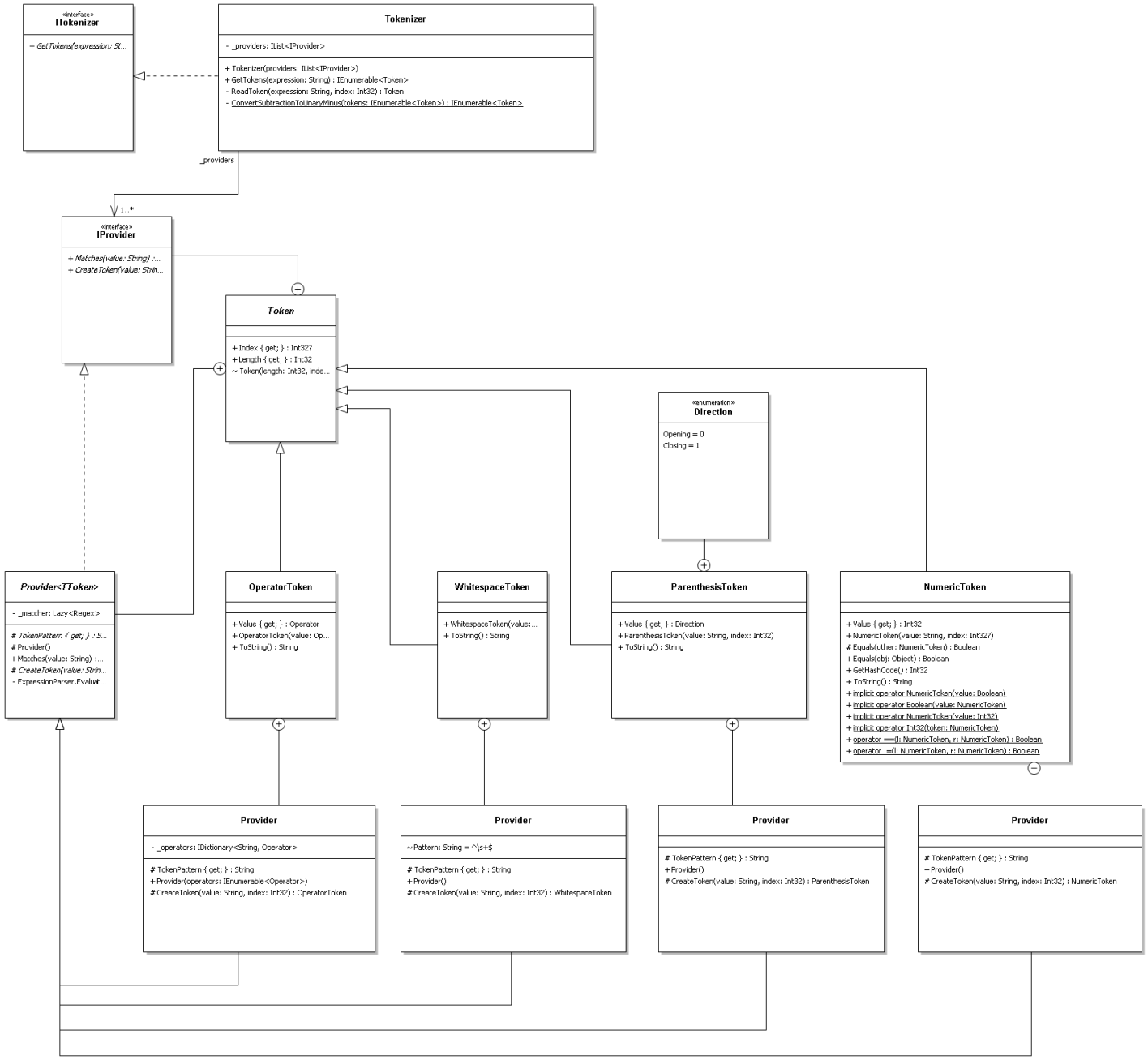
Program



Operators


NegationOperator
+ Symbol { get; } : String
+ Precedence { get; } : Int32
+ Operands { get; } : Int32
+ NegationOperator()
+ ToString() : String
~ Evaluate(operands: Num...
PowerOperator
+ Symbol { get; } : String
+ Precedence { get; } : Int32
+ Operands { get; } : Int32
+ PowerOperator()
~ Evaluate(operands: Num...
Operator
+ Symbol { get; } : String
+ Precedence { get; } : Int32
+ Operands { get; } : Int32
# Operator()
~ Evaluate(operands: Num...
+ ToString() : String
ComparisonOperator
+ Operands { get; } : Int32
# ComparisonOperator()
# Evaluate(l: NumericToken, r: NumericToken) : NumericToken
~ Evaluate(operands: NumericToken[]) : NumericToken
GreaterOrEqualOper...
EqualityOperator
InequalityOperator
LessThanOperator
GreaterThanOperator
LessOrEqualOperator
ArithmeticOperator
+ Operands { get; } : Int32
# ArithmeticOperator()
# Evaluate(l: NumericToken, r: NumericToken) : NumericToken
~ Evaluate(operands: NumericToken[]) : NumericToken
ModulusOperator
AddOperator
SubtractOperator
MultiplyOperator
DivideOperator
LogicalOperator
+ Operands { get; } : Int32
# LogicalOperator()
# Evaluate(l: NumericToken, r: NumericToken) : NumericToken
~ Evaluate(operands: NumericToken[]) : NumericToken
NotOperator
AndOperator
OrOperator
PrefixOperator
+ Operands { get; } : Int32
# PrefixOperator()
# Evaluate(r: NumericToken) : NumericToken
~ Evaluate(operands: NumericToken[]) : NumericToken
IncrementOperator
DecrementOperator

Tokenizer


«interface»
ITokenizer
+ GetTokens(expression: St...
Tokenizer
- _providers: IList<IProvider>
+ Tokenizer(providers: IList<IProvider>)
+ GetTokens(expression: String) : IEnumerable<Token>
- ReadToken(expression: String, index: Int32) : Token
- ConvertSubtractionToUnaryMinus(tokens: IEnumerable<Token>) : IEnumerable<Token>
_providers
1..*
«interface»
IProvider
+ Matches(value: String) :...
+ CreateToken(value: Strin...
Token
+ Index { get; } : Int32?
+ Length { get; } : Int32
~ Token(length: Int32, inde...
«enumeration»
Direction
Opening = 0
Closing = 1
Provider<TToken>
- _matcher: Lazy<Regex>
TokenPattern { get; } : S...
Provider()
+ Matches(value: String) :...
# CreateToken(value: Strin...
- ExpressionParser.Evaluat...
OperatorToken
+ Value { get; } : Operator
+ OperatorToken(value: Op...
+ ToString() : String
WhitespaceToken
+ WhitespaceToken(value:...
+ ToString() : String
ParenthesisToken
+ Value { get; } : Direction
+ ParenthesisToken(value: String, index: Int32)
+ ToString() : String
NumericToken
+ Value { get; } : Int32
+ NumericToken(value: String, index: Int32?)
# Equals(other: NumericToken) : Boolean
+ Equals(obj: Object) : Boolean
+ GetHashCode() : Int32
+ ToString() : String
+ implicit operator NumericToken(value: Boolean)
+ implicit operator Boolean(value: NumericToken)
+ implicit operator NumericToken(value: Int32)
+ implicit operator Int32(token: NumericToken)
+ operator ==(l: NumericToken, r: NumericToken) : Boolean
+ operator !=(l: NumericToken, r: NumericToken) : Boolean
Provider
- _operators: IDictionary<String, Operator>
TokenPattern { get; } : String
+ Provider(operators: IEnumerable<Operator>)
# CreateToken(value: String, index: Int32) : OperatorToken
Provider
~ Pattern: String = ^\s+$
# TokenPattern { get; } : String
+ Provider()
# CreateToken(value: String, index: Int32) : WhitespaceToken
Provider
TokenPattern { get; } : String
+ Provider()
# CreateToken(value: String, index: Int32) : ParenthesisToken
Provider
TokenPattern { get; } : String
+ Provider()
# CreateToken(value: String, index: Int32) : NumericToken

Efficiency of Algorithms

- ExpressionParser.Evaluator.Tokens.Tokenizer::GetTokens(string expression)
  - O(n) – Reads every character of the expression approximately once
- ExpressionParser.Evaluator.Toens.Tokenizer::ReadToken(sting expression, int index)
  - O(n) – Reads the largest consumable token in the input, matching against a small but fixed number of token providers
- ExpressionParser.Evaluator.Tokens.Tokenizer::ConvertSubtractionToUnaryMinus(IEnumerable<Token> tokens)
  - O(n) – Reads each token once
- ExpressionParser.Evaluator.InfixToPostfix.InfixConverter::Convert(IEnumerable<Token> tokens)
  - O(n) – Reads, pushes, and outputs every token at most once
- ExpressionParser.Evaluator.Evaluator.Evaluate(string expression)
  - O(n) – Dependencies are O(n), evaluation algorithm reads each postfix token in the input once, and evaluates each operator in the input once

References

- Shunting-yard algorithm. (2017, January 21). In *Wikipedia, The Free Encyclopedia*. Retrieved 04:22, March 1, 2017, from https://en.wikipedia.org/w/index.php?title=Shunting-yard_algorithm&oldid=761259190