public class ShuntingYardAlgorithm

{

public ShuntingYardAlgorithm()

{

\_operatorsStack = new Stack<OperatorToken>();

\_postfixNotationTokens = new List<IToken>();

}

public IEnumerable<IToken> Apply(IEnumerable<IToken> infixNotationTokens)

{

Reset();

foreach (var token in infixNotationTokens)

{

ProcessToken(token);

}

return GetResult();

}

private void Reset()

{

\_operatorsStack.Clear();

\_postfixNotationTokens.Clear();

}

private void ProcessToken(IToken token)

{

switch (token)

{

case OperandToken operandToken:

StoreOperand(operandToken);

break;

case OperatorToken operatorToken:

ProcessOperator(operatorToken);

break;

default:

var exMessage = $"An unknown token type: {token.GetType().ToString()}.";

throw new SyntaxException(exMessage);

}

}

private void StoreOperand(OperandToken operandToken)

{

\_postfixNotationTokens.Add(operandToken);

}

private void ProcessOperator(OperatorToken operatorToken)

{

switch (operatorToken.OperatorType)

{

case OperatorType.OpeningBracket:

PushOpeningBracket(operatorToken);

break;

case OperatorType.ClosingBracket:

PushClosingBracket(operatorToken);

break;

default:

PushOperator(operatorToken);

break;

}

}

private void PushOpeningBracket(OperatorToken operatorToken)

{

\_operatorsStack.Push(operatorToken);

}

private void PushClosingBracket(OperatorToken operatorToken)

{

bool openingBracketFound = false;

while (\_operatorsStack.Count > 0)

{

var stackOperatorToken = \_operatorsStack.Pop();

if (stackOperatorToken.OperatorType == OperatorType.OpeningBracket)

{

openingBracketFound = true;

break;

}

\_postfixNotationTokens.Add(stackOperatorToken);

}

if (!openingBracketFound)

{

throw new SyntaxException("An unexpected closing bracket.");

}

}

private void PushOperator(OperatorToken operatorToken)

{

var operatorPriority = GetOperatorPriority(operatorToken);

while (\_operatorsStack.Count > 0)

{

var stackOperatorToken = \_operatorsStack.Peek();

if (stackOperatorToken.OperatorType == OperatorType.OpeningBracket)

{

break;

}

var stackOperatorPriority = GetOperatorPriority(stackOperatorToken);

if (stackOperatorPriority < operatorPriority)

{

break;

}

\_postfixNotationTokens.Add(\_operatorsStack.Pop());

}

\_operatorsStack.Push(operatorToken);

}

private int GetOperatorPriority(OperatorToken operatorToken)

{

switch (operatorToken.OperatorType)

{

case OperatorType.Addition:

case OperatorType.Subtraction:

return 1;

case OperatorType.Multiplication:

case OperatorType.Division:

return 2;

default:

var exMessage = "An unexpected action for the operator: " +

$"{operatorToken.OperatorType}.";

throw new SyntaxException(exMessage);

}

}

private List<IToken> GetResult()

{

while (\_operatorsStack.Count > 0)

{

var token = \_operatorsStack.Pop();

if (token.OperatorType == OperatorType.OpeningBracket)

{

throw new SyntaxException("A redundant opening bracket.");

}

\_postfixNotationTokens.Add(token);

}

return \_postfixNotationTokens.ToList();

}

private readonly Stack<OperatorToken> \_operatorsStack;

private readonly List<IToken> \_postfixNotationTokens;

}