Wesley Kwan

CS 2400.02

Project 2

March 9, 2020

Section 1:

**Class Expression**

* + **Method Summary**

|  |  |
| --- | --- |
|  | |
| **Modifier and Type** | **Method and Description** |
| static java.lang.String[] | [convertToPostfix](file:///C:\Users\wkwan\OneDrive\Desktop\Expression.html#convertToPostfix-java.lang.String:A-)(java.lang.String[] infixExpression)  Converts a given infix expression to a postfix expression using a stack. |
| static int | [evaluatePostfix](file:///C:\Users\wkwan\OneDrive\Desktop\Expression.html#evaluatePostfix-java.lang.String:A-BagInterface-)(java.lang.String[] postfixExpression, BagInterface<Name> nameBag)  Evaluates a given postfix expression using a stack. |

* + **Method Detail**
    - **convertToPostfix**

public static java.lang.String[] convertToPostfix(java.lang.String[] infixExpression)

Converts a given infix expression to a postfix expression using a stack.

Parameters:

infixExpression - the expression to be converted into postfix

Returns:

the converted postfix expression

Throws:

java.lang.RuntimeException - if the infix expression is not well formed

* + - **evaluatePostfix**

public static int evaluatePostfix(java.lang.String[] postfixExpression,

BagInterface<Name> nameBag)

Evaluates a given postfix expression using a stack.

Parameters:

postfixExpression - the postfix expression to be evaluated

nameBag - a bag of Names (the variables in the expression)

Returns:

the value of the postfix expression

Throws:

java.lang.RuntimeException - if the postfix expression is not well formed or a name is not found in nameBag

Section 2:

The methods convertToPostfix and evaluatePostfix were tested with multiple cases, whereby after each method call the expected and actual result of the method are printed for comparison. And all expected exceptions were caught and dealt with so the program could continue.

|  |  |
| --- | --- |
| Case 1:  Infix expression: a + 10 / b  Variables: a = 5, b = 10  Converting to from infix to postfix...  Expected result: a10b/+  Actual result: a10b/+  Evaluating the postfix expression...  Expected result: 6  Actual result: 6 | Case 2:  Infix expression: x + y + z  Variables: x = 1, y = 2, z = 3  Converting to from infix to postfix...  Expected result: xy+z+  Actual result: xy+z+  Evaluating the postfix expression...  Expected result: 6  Actual result: 6 |

Note: convertToPostfix can identify operators, variable names, and integer literals, and correctly determines the precedence of operators. evaluatePostfix can read the given expression and use a bag of Names to evaluate the expression.

|  |  |
| --- | --- |
| Case 3:  Infix expression: ( m + n ) \* ( r - s )  Variables: m = 8, n = -3, r = 12, s = 10  Converting to from infix to postfix...  Expected result: mn+rs-\*  Actual result: mn+rs-\*  Evaluating the postfix expression...  Expected result: 10  Actual result: 10 | Case 4:  Infix expression: alpha + beta ^ ( gamma - 1 )  Variables: alpha = 1, beta = 2, gamma = 3  Converting to from infix to postfix...  Expected result: alphabetagamma1-^+  Actual result: alphabetagamma1-^+  Evaluating the postfix expression...  Expected result: 5  Actual result: 5 |

Note: convertToPostfix can identify a variety of variable names, and correctly determines the precedence of operators with the presence parentheses.

|  |  |
| --- | --- |
| Case 5:  Infix expression: a ^ ( b - c  Variables: a = 7, b = 4, c = 2  Converting to from infix to postfix...  RuntimeException: Infix expression is not well formed. | New infix expression: a ^ ( b - c )  Variables: a = 7, b = 4, c = 2  Converting to from infix to postfix...  Expected result: abc-^  Actual result: abc-^  Evaluating the postfix expression...  Expected result: 49  Actual result: 49 |

Note: convertToPostfix throws a RuntimeException when the given infix expression is not well formed, in this case it had an unmatched parenthesis. The exception was caught: the infix expression was replaced and reconverted. Afterwards, the program continues.

|  |  |
| --- | --- |
| Case 6:  Infix expression: a / \* c  Variables: a = 12, b = 6, c = 0  Converting to from infix to postfix...  Expected result: ab/c\*  Actual result: a/c\*  Evaluating the postfix expression...  RuntimeException: Postfix expression is not well formed. | New infix expression: a / b \* c  Variables: a = 12, b = 6, c = 0  Converting to from infix to postfix...  Expected result: ab/c\*  Actual result: ab/c\*  Evaluating the postfix expression...  Expected result: 0  Actual result: 0 |

Note: evaluatePostfix throws a RuntimeException when the given postfix expression is not well formed, in this case it had a missing operand (“b”) which caused a pop on an empty stack. The exception was caught: the infix expression was replaced and reconverted to postfix, and the postfix expression was reevaluated. Afterwards, the program continues.

|  |  |
| --- | --- |
| Case 7:  Infix expression: a + b  Variables: a = 24 (b not provided)  Converting to from infix to postfix...  Expected result: ab+  Actual result: ab+  Evaluating the postfix expression...  RuntimeException: Name not found. | New variables: a = 24, b = 36  Evaluating the postfix expression...  Expected result: 60  Actual result: 60 |

Note: evaluatePostfix throws a RuntimeException when a variable name in the postfix expression is not found in the given bag, in this case variable name “b” was not added to the bag. The exception was caught: “b” was added to the bag and the postfix expression was reevaluated. Afterwards, the program continues.

Section 3:

From this project I learned how to convert infix to postfix and evaluate postfix expressions using a stack. I also furthered my understanding of and practice with generics and polymorphism while managing objects in the methods.