

CMSC 204

Assignment #2

Notation

Infix notation is the **notation** commonly used in arithmetical and logical formulae and statements. It is characterized by the placement of operators between operands – "infix operators" – such as the plus sign in "2 + 2".

Postfix notation is a **notation** for writing arithmetic expressions in which the operands appear before their operators. There are no precedence rules to learn, and parentheses are never needed.

You will be creating a utility class that converts an infix expression to a postfix expression, a postfix expression to an infix expression and evaluates a postfix expression.

The GUI is provided.

Concepts tested:

Generic Queue

Generic Stack

Exception handling

Data Element

String

Data Structures

1. Create a generic queue class called *NotationQueue*. *NotationQueue* will implement the *QueueInterface* given you. You will be creating *NotationQueue* from scratch (do not use an internal object derived from the *Queue* interface from the Java API, from *java.util*)
2. Create a generic stack class called *NotationStack*. *NotationStack* will implement the *Stack Interface* given you. You will be creating *NotationStack* from scratch (do not use an internal object of the *Stack* class from the Java API, *java.util*)

Utility Class

The *Notation* class will have a method *infixToPostfix* to convert infix notation to postfix notation that will take in a string and return a string, a method *postfixToInfix* to convert postfix notation to infix notation that will take in a string and return a string, and a method *evaluatePostfix* to evaluate the postfix expression. It will take in a string and return a double.

In the infixToPostfix method, you MUST use a queue for the internal structure that holds the postfix solution. Then use the toString method of the Queue to return the solution as a string.

For simplicity sake:

- a. operands will be single digit numbers
- b. only the following arithmetic operations will be allowed in an expression:
 - + addition
 - subtraction
 - * multiplication
 - / division

Exception Classes

Provide the following exception classes:

1. InvalidNotationFormatException – occurs when a Notation format is incorrect
2. StackOverflowException – occurs when a top or pop method is called on an empty stack. **(SWITCH THIS WITH 3)**
3. StackUnderflowException – occurs when a push method is called on a full stack.
4. QueueOverflowException – occurs when a dequeue method is called on an empty queue.
5. QueueUnderflowException – occurs when a enqueue method is called on a full queue.

GUI Driver (provided)

1. Initially neither radio button for notation is selected. When a radio button is selected, the Convert button is enabled and the appropriate label and field are visible for the user input.
2. When the user selects the Convert button, the appropriate label and field with the conversion will be displayed.
3. When the user selects the Evaluate button, the “answer” to the expression will be displayed.
4. When the Exit button is selected, the application will close.

ALGORITHMS

Infix expression to postfix expression:

Read the infix expression from left to right and do the following:

If the current character in the infix is a space, ignore it.

If the current character in the infix is a digit, copy it to the postfix solution queue

If the current character in the infix is a left parenthesis, push it onto the stack

If the current character in the infix is an operator,

1. Pop operators (if there are any) at the top of the stack while they have equal or higher precedence than the current operator, and insert the popped operators in postfix solution queue
2. Push the current character in the infix onto the stack

If the current character in the infix is a right parenthesis

1. Pop operators from the top of the stack and insert them in postfix solution queue until a left parenthesis is at the top of the stack, if no left parenthesis-throw an error
2. Pop (and discard) the left parenthesis from the stack

When the infix expression has been read, Pop any remaining operators and insert them in postfix solution queue.

Postfix expression to infix expression:

Read the postfix expression from left to right and do the following:

If the current character in the postfix is a space, ignore it.

If the current character is an operand, push it on the stack

If the current character is an operator,

1. Pop the top 2 values from the stack. If there are fewer than 2 values throw an error
2. Create a string with 1st value and then the operator and then the 2nd value.
3. Encapsulate the resulting string within parenthesis
4. Push the resulting string back to the stack

When the postfix expression has been read:

If there is only one value in the stack – it is the infix string, if more than one value, throw an error

Evaluating a postfix expression

Read the postfix expression from left to right and to the following:

If the current character in the postfix expression is a space, ignore it.

If the current character is an operand or left parenthesis, push on the stack

If the current character is an operator,

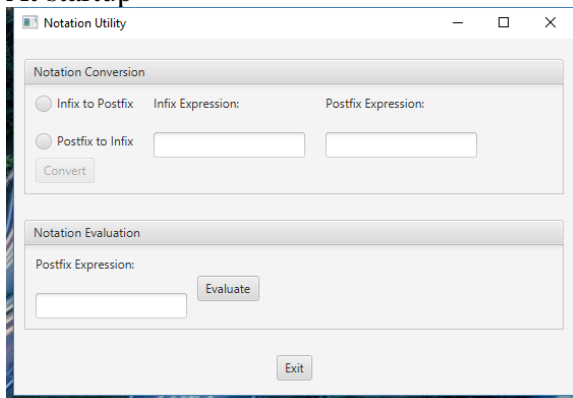
1. Pop the top 2 values from the stack. If there are fewer than 2 values throw an error
2. Perform the arithmetic calculation of the operator with the first popped value as the right operand and the second popped value as the left operand
3. Push the resulting value onto the stack

When the postfix expression has been read:

If there is only one value in the stack – it is the result of the postfix expression, if more than one value, throw an error

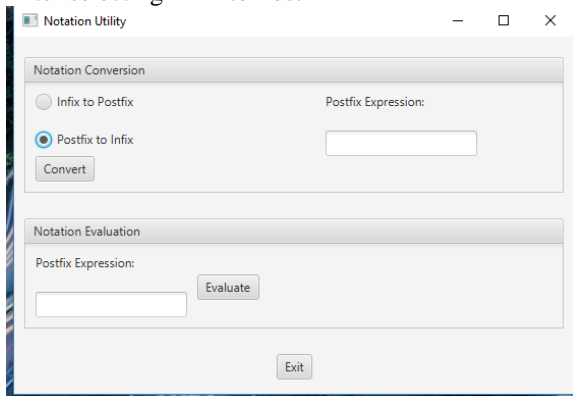
Examples:

At startup



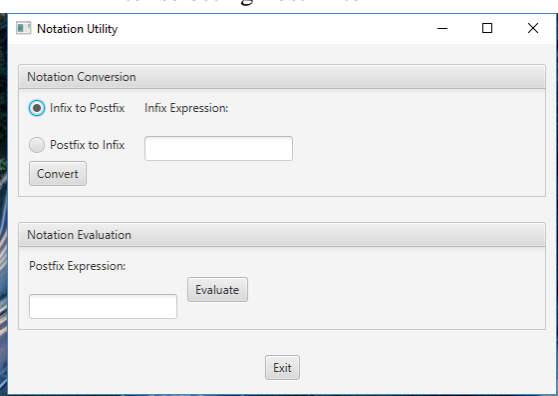
The screenshot shows the 'Notation Utility' window at startup. It has two main sections: 'Notation Conversion' and 'Notation Evaluation'. In the 'Notation Conversion' section, there are two radio buttons: 'Infix to Postfix' (selected) and 'Postfix to Infix'. To the right of these are two empty text boxes labeled 'Infix Expression:' and 'Postfix Expression:'. Below the radio buttons is a 'Convert' button. In the 'Notation Evaluation' section, there is a text box labeled 'Postfix Expression:' and an 'Evaluate' button. At the bottom right of the window is an 'Exit' button.

After selecting Infix to Postfix



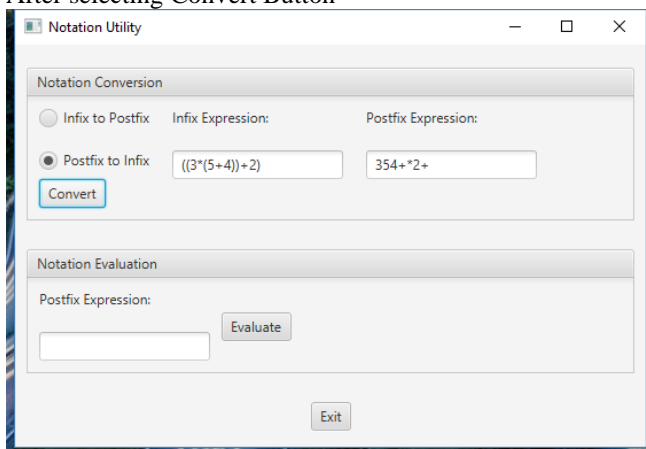
The screenshot shows the 'Notation Utility' window after selecting 'Infix to Postfix'. The 'Infix to Postfix' radio button is now selected, and the 'Postfix to Infix' radio button is unselected. The 'Convert' button is still present. The 'Notation Evaluation' section remains unchanged with its 'Postfix Expression:' text box and 'Evaluate' button. The 'Exit' button is at the bottom right.

After selecting Postfix to Infix



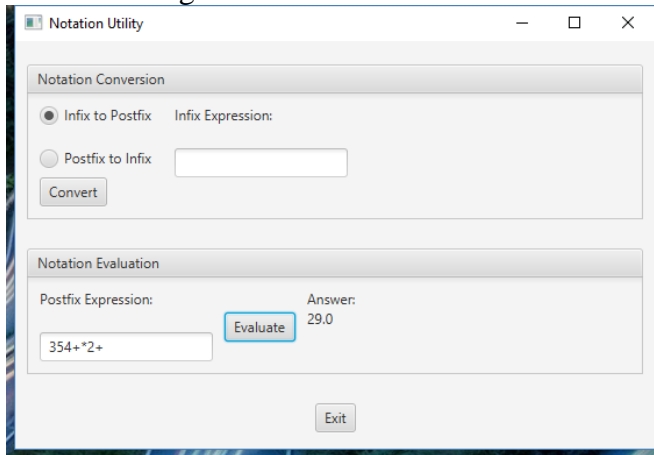
The screenshot shows the 'Notation Utility' window after selecting 'Postfix to Infix'. The 'Postfix to Infix' radio button is now selected, and the 'Infix to Postfix' radio button is unselected. The 'Convert' button is still present. The 'Notation Evaluation' section remains unchanged with its 'Postfix Expression:' text box and 'Evaluate' button. The 'Exit' button is at the bottom right.

After selecting Convert Button



The screenshot shows the 'Notation Utility' window after selecting the 'Convert' button. The 'Postfix to Infix' radio button is selected. The 'Infix Expression:' text box now contains the expression $((3*(5+4))+2)$. The 'Postfix Expression:' text box now contains the expression $354+*2+$. The 'Convert' button is highlighted with a blue border. The 'Notation Evaluation' section remains unchanged with its 'Postfix Expression:' text box and 'Evaluate' button. The 'Exit' button is at the bottom right.

After selecting Evaluate button



The screenshot shows a window titled "Notation Utility" with standard window controls (minimize, maximize, close). The window is divided into two main sections: "Notation Conversion" and "Notation Evaluation".

Notation Conversion Section:

- It contains two radio buttons: "Infix to Postfix" (which is selected) and "Postfix to Infix".
- Next to the "Infix to Postfix" radio button is the label "Infix Expression:".
- Below the radio buttons is a "Convert" button.
- There is an empty text input field next to the "Postfix to Infix" radio button.

Notation Evaluation Section:

- It contains a label "Postfix Expression:" followed by a text input field containing the expression "354*2+".
- To the right of the input field is a blue "Evaluate" button.
- To the right of the "Evaluate" button is the label "Answer:" followed by the text "29.0".
- At the bottom center of the window is an "Exit" button.

Grading Rubric - CMSC 204 Project #2

Name _____ .

PROGRAMMING (100 pts)

Compiles 40 pts _____

Accuracy

Passes public JUnit tests 15 pts _____

Passes STUDENT JUnit tests 10 pts _____

Passes private instructor tests 15 pts _____

Execution: runs without errors (either run-time or logic errors) 20 pts _____

Possible Sub-total 100 pts _____

REQUIREMENTS (Subtracts from Programming total)

Documentation:

Javadoc was not provided for all student generated classes - 5 pts _____

Documentation within source code was missing or incorrect - 5 pts _____

Description of what class does was missing

Author's Name, @author, was missing

Methods not commented properly using Javadoc @param, @return

JUnit STUDENT methods were not implemented -5 pts _____

Learning Experience (text document) -4 pts _____

In 3+ paragraphs, highlight your lessons learned and learning experience from working on this project. What have you learned? What did you struggle with? What would you do differently on your next project?

Programming Style:

Incorrect use of indentation, statements, structures -10 pts _____

Design: Classes do not have the functionality specified, i.e.,

1. Data Structures classes - 16 pts _____

- Generic Stack class

- Generic Queue class

2. Utility class - Notation -15 pts _____

- Does not follow provided Javadoc

- Does not correctly handle exceptions

Possible decrements: -60 pts _____

Possible total grade: 100 pts _____