Assignment 2: Solving Expressions in Postfix Notation using Stacks

Paulius Zabinskas

Date: 07/02/2023

ID: 20120267

Assignment Report:

- **Problem Statement:**

The assignment problem is to read a string, string read must contain integers, consist of length 3-20 characters. Using stack Data Structure transform infix notation to postfix notation. Then, evaluate the postfix notation using stack again, return double value.

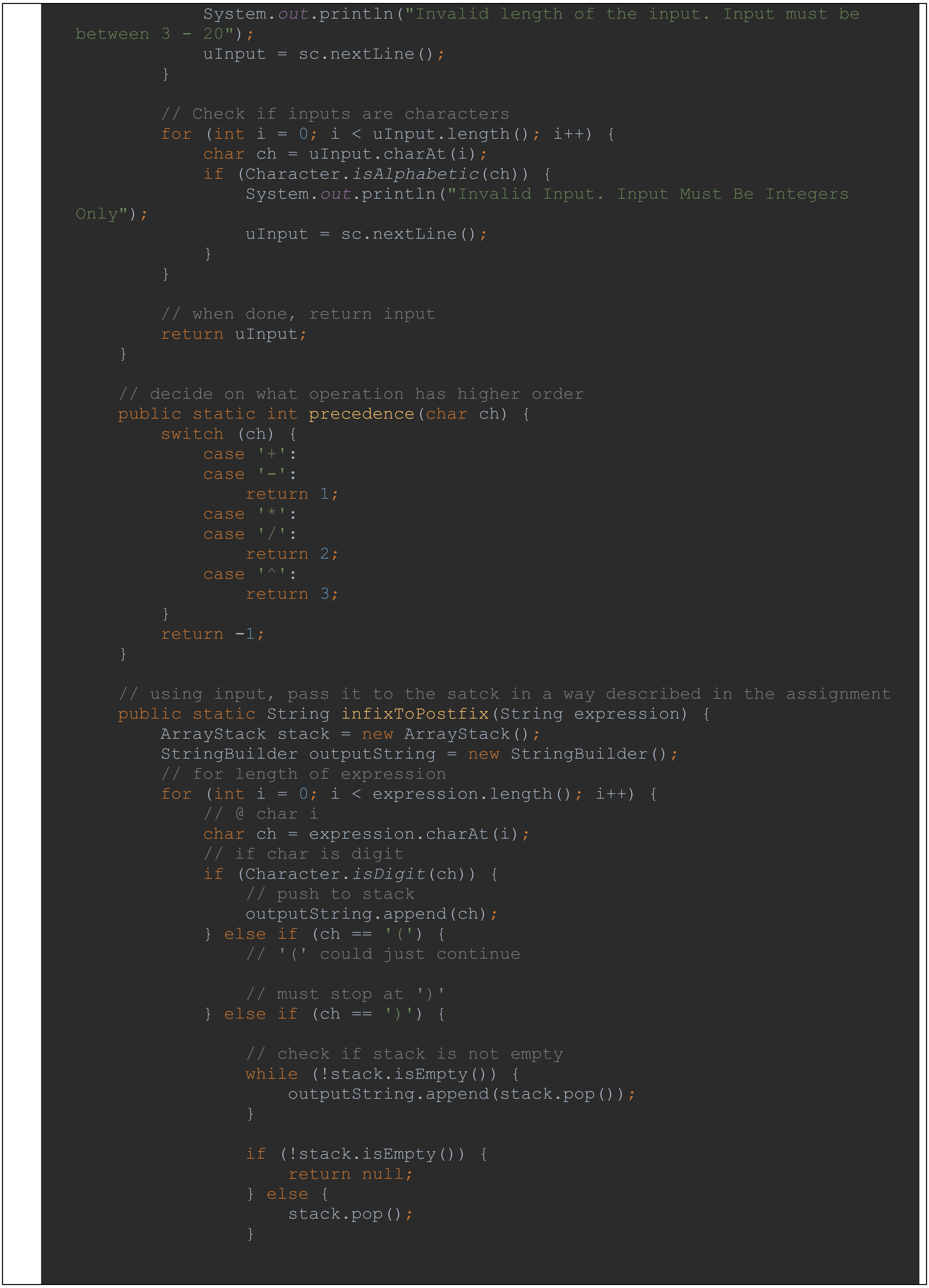
- **Analysis and Design Notes:**

The methods I will need:

Methods	Functionality
Read Input	Read in input. While input contains a letter, or is shorter than 3 characters, or longer than 20 characters, ask again for valid input.
Precedence	When reading operation symbols, return an integer of precedence to measure what operation should be performed first.
Infix To Postfix	Build a new string using stack data structure. Logic is defined in assignment notes. Precedence method will be needed.
Evaluate Postfix	Evaluation of postfix expression. Logic is defined in in assignment notes. Perform operation method will be used.
Perform Operation	When two integers are read, use following operator on numbers.

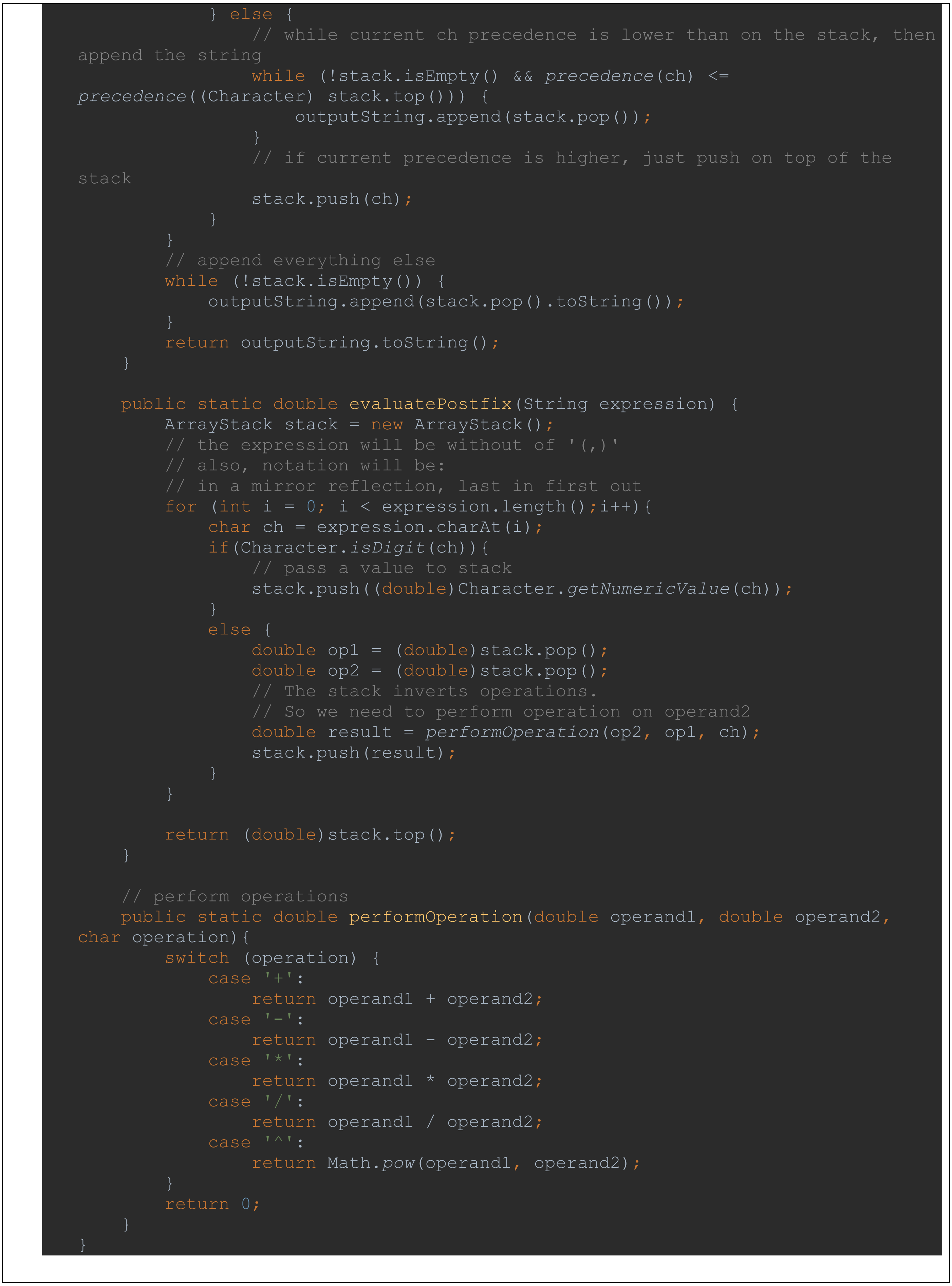
- **Code:**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        String infix = readInput();
        String postfix = infixToPostfix(infix);
        System.out.println("Infix Expression: " + infix + "\nPostfix
Expression: " + postfix);
        System.out.println("The Value: "+ evaluatePostfix(postfix));

    }

    // read input
    // try to get rid of all invalid inputs here
    public static String readInput() {

        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Infix Expression");
        String uInput = sc.nextLine();  // Read user input
        // check if input length is 3-20 chars
        while ((uInput.length() < 3) || (uInput.length() > 20)) {
```



```
            System.out.println("Invalid length of the input. Input must be 
between 3 - 20");
            uInput = sc.nextLine();
        }

        // Check if inputs are characters
        for (int i = 0; i < uInput.length(); i++) {
            char ch = uInput.charAt(i);
            if (Character.isAlphabetic(ch)) {
                System.out.println("Invalid Input. Input Must Be Integers 
Only");

                uInput = sc.nextLine();
            }
        }

        // when done, return input
        return uInput;
    }

    // decide on what operation has higher order
    public static int precedence(char ch) {
        switch (ch) {
            case '+':
            case '-':
                return 1;
            case '*':
            case '/':
                return 2;
            case '^':
                return 3;
        }
        return -1;
    }

    // using input, pass it to the satck in a way described in the assignment
    public static String infixToPostfix(String expression) {
        ArrayStack stack = new ArrayStack();
        StringBuilder outputString = new StringBuilder();
        // for length of expression
        for (int i = 0; i < expression.length(); i++) {
            // @ char i
            char ch = expression.charAt(i);
            // if char is digit
            if (Character.isDigit(ch)) {
                // push to stack
                outputString.append(ch);
            } else if (ch == '(') {
                // '(' could just continue

                // must stop at ')'
            } else if (ch == ')') {

                // check if stack is not empty
                while (!stack.isEmpty()) {
                    outputString.append(stack.pop());
                }

                if (!stack.isEmpty()) {
                    return null;
                } else {
                    stack.pop();
                }
```

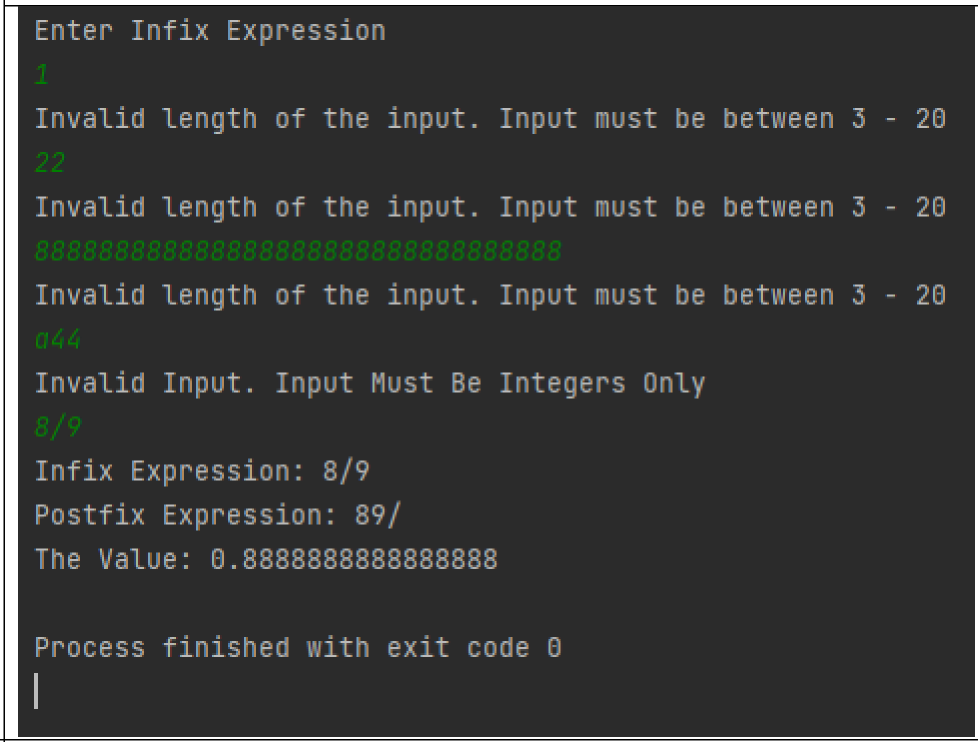
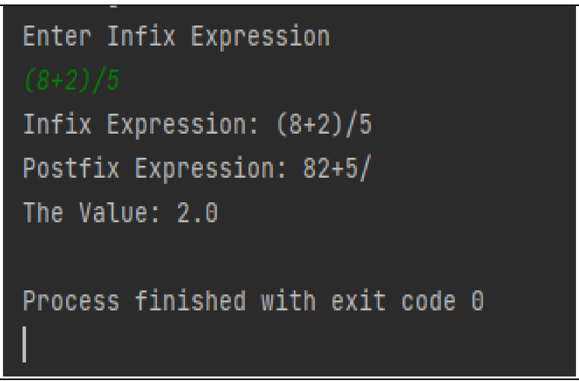
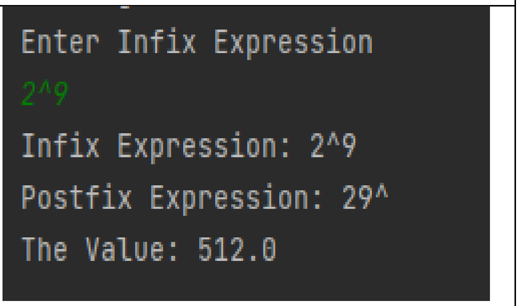
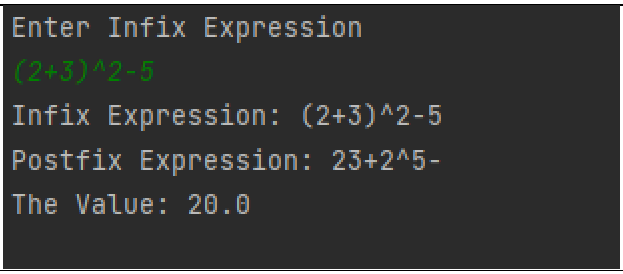
Assignment 2: Solving Expressions in Postfix Notation using Stacks



```
            } else {
                // while current ch precedence is lower than on the stack, then
append the string
                while (!stack.isEmpty() && precedence(ch) <=
precedence((Character) stack.top())) {
                    outputString.append(stack.pop());
                }
                // if current precedence is higher, just push on top of the
stack
                stack.push(ch);
            }
        }
        // append everything else
        while (!stack.isEmpty()) {
            outputString.append(stack.pop().toString());
        }
        return outputString.toString();
    }

    public static double evaluatePostfix(String expression) {
        ArrayStack stack = new ArrayStack();
        // the expression will be without of '(,)'
        // also, notation will be:
        // in a mirror reflection, last in first out
        for (int i = 0; i < expression.length();i++){
            char ch = expression.charAt(i);
            if(Character.isDigit(ch)){
                // pass a value to stack
                stack.push((double)Character.getNumericValue(ch));
            }
            else {
                double op1 = (double)stack.pop();
                double op2 = (double)stack.pop();
                // The stack inverts operations.
                // So we need to perform operation on operand2
                double result = performOperation(op2, op1, ch);
                stack.push(result);
            }
        }

        return (double)stack.top();
    }

    // perform operations
    public static double performOperation(double operand1, double operand2,
char operation){
        switch (operation) {
            case '+':
                return operand1 + operand2;
            case '-':
                return operand1 - operand2;
            case '*':
                return operand1 * operand2;
            case '/':
                return operand1 / operand2;
            case '^':
                return Math.pow(operand1, operand2);
        }
        return 0;
    }
}
```

Assignment 2: Solving Expressions in Postfix Notation using Stacks

- **Testing:**

Testing:	Result:	
Invalid input:	• If input values will contain letters, length won’t be between 3-20, then will be asked to pass new input.	
	 Enter Infix Expression 1 Invalid length of the input. Input must be between 3 - 20 22 Invalid length of the input. Input must be between 3 - 20 888888888888888888888888888888888 Invalid length of the input. Input must be between 3 - 20 a44 Invalid Input. Input Must Be Integers Only 8/9 Infix Expression: 8/9 Postfix Expression: 89/ The Value: 0.8888888888888888 Process finished with exit code 0	
Infix To Postfix	• Output string should be containing 2 digits, operation. Then it may contain 1 or 2 digits, operation.	
	 Enter Infix Expression (8+2)/5 Infix Expression: (8+2)/5 Postfix Expression: 82+5/ The Value: 2.0 Process finished with exit code 0	 Enter Infix Expression 2^9 Infix Expression: 2^9 Postfix Expression: 29^ The Value: 512.0
Symbols are interpreted correctly.	I expect the mathematical operations to be interpreted correctly.	
	 Enter Infix Expression (2+3)^2-5 Infix Expression: (2+3)^2-5 Postfix Expression: 23+2^5- The Value: 20.0	

Assignment 2: Solving Expressions in Postfix Notation using Stacks

- **Challenges:**

The Infix to Postfix method functionality, logic implementation was difficult and time consuming.