CS 102: Data Structures

Project Three - Super Strong Stack

Student Name	Mohammad El-Abid
Student ID	8905652
Professor	Subrina Thompson
Date	Sunday, March 25, 2012

Table of Contents

Abstract

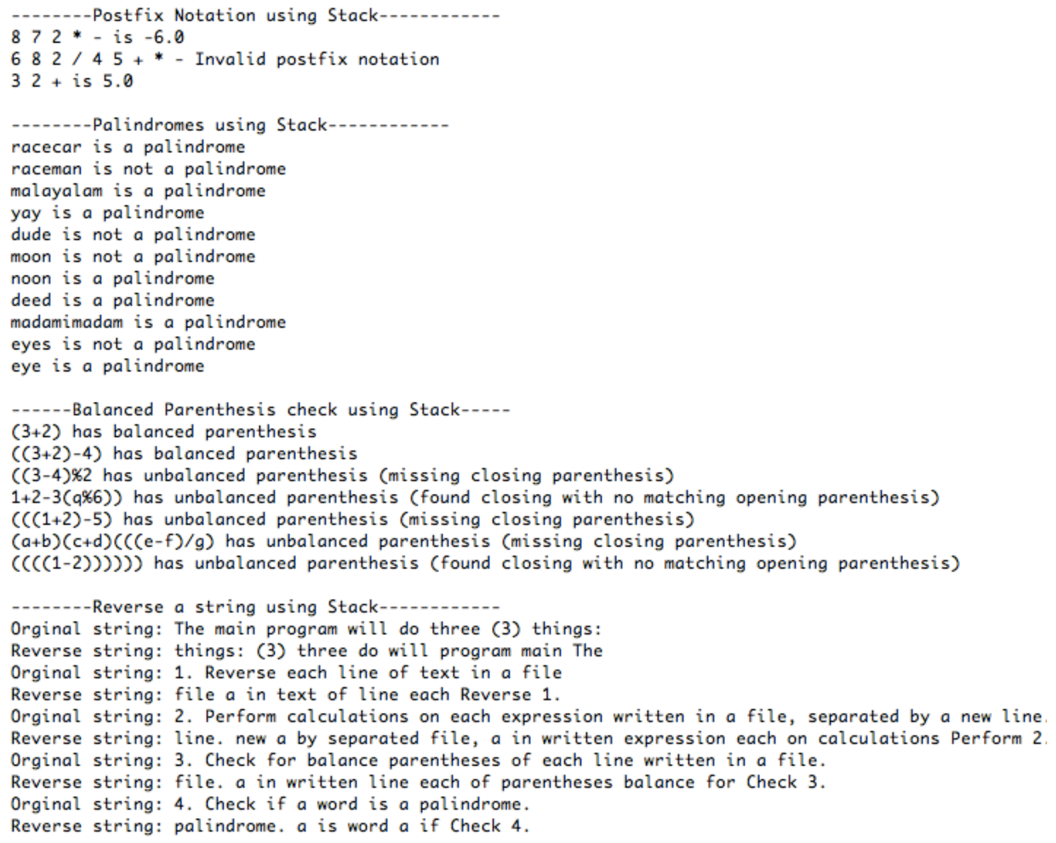
This application demonstrates our Stack class and by interpreting post-fix notation, checking palindromes, balanced parenthesis in expressions, and reversing a string by word, not character.

Introduction

I did all of these in reverse order, seemed to be more easiest to hardest. Other then that, simple `pushing` to a `Stack` class and understanding when to stop `pushing` and start `poping`.

Screenshots

Application output to `stdout`:



```
--------Postfix Notation using Stack------------
8 7 2 * - is -6.0
6 8 2 / 4 5 + * - Invalid postfix notation
3 2 + is 5.0

--------Palindromes using Stack------------
racecar is a palindrome
raceman is not a palindrome
malayalam is a palindrome
yay is a palindrome
dude is not a palindrome
moon is not a palindrome
noon is a palindrome
deed is a palindrome
madamimadam is a palindrome
eyes is not a palindrome
eye is a palindrome

------Balanced Parenthesis check using Stack-----
(3+2) has balanced parenthesis
((3+2)-4) has balanced parenthesis
((3-4)%2 has unbalanced parenthesis (missing closing parenthesis)
1+2-3(q%6)) has unbalanced parenthesis (found closing with no matching opening parenthesis)
(((1+2)-5) has unbalanced parenthesis (missing closing parenthesis)
(a+b)(c+d)(((e-f)/g) has unbalanced parenthesis (missing closing parenthesis)
((((1-2)))))) has unbalanced parenthesis (found closing with no matching opening parenthesis)

--------Reverse a string using Stack------------
Orginal string: The main program will do three (3) things:
Reverse string: things: (3) three do will program main The
Orginal string: 1. Reverse each line of text in a file
Reverse string: file a in text of line each Reverse 1.
Orginal string: 2. Perform calculations on each expression written in a file, separated by a new line.
Reverse string: line. new a by separated file, a in written expression each on calculations Perform 2.
Orginal string: 3. Check for balance parentheses of each line written in a file.
Reverse string: file. a in written line each of parentheses balance for Check 3.
Orginal string: 4. Check if a word is a palindrome.
Reverse string: palindrome. a is word a if Check 4.
```

Code

File: Application.java

```java
package edu.bridgeport.mohammad;

import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class Application {
    public static void main(String[] args) throws IOException {
          // Prefix Notation
          System.out.println("--------Postfix Notation using
Stack------------");
          FileReader fob = new FileReader("postfix.txt");
          Scanner in = new Scanner(fob);
          while(in.hasNextLine()) postfix(in.nextLine());
          in.close();

          // Palindrome
          System.out.println();
          System.out.println("--------Palindromes using
Stack------------");
          fob = new FileReader("palindrome.txt");
          in = new Scanner(fob);
          while(in.hasNextLine()) palindrome(in.nextLine());
          in.close();

          // Parenthesis
          System.out.println();
          System.out.println("------Balanced Parenthesis check using
Stack-----");
          fob = new FileReader("paren.txt");
          in = new Scanner(fob);
          while(in.hasNextLine()) parenthesis(in.nextLine());
          in.close();


          // Reverse
          System.out.println();
          System.out.println("--------Reverse a string using
Stack------------");
          fob = new FileReader("reverse.txt");
          in = new Scanner(fob);
          while(in.hasNextLine()) reverse(in.nextLine());
          in.close();
    }
    private static void reverse(String line) {
          Stack<String> stack = new Stack<String>();
          Scanner input = new Scanner(line);

          System.out.print("Orginal string: ");
          while(input.hasNext()){
               String next = input.next();
               System.out.print(next + " ");
               stack.push(next);
          }
          System.out.println();

          System.out.print("Reverse string: ");
          while(!stack.isEmpty()){
               System.out.print(stack.pop() + " ");
          }
          System.out.println();
    }

    private static void parenthesis(String line) {
          Stack<Character> stack = new Stack<Character>();

          System.out.print(line + " ");

          for( char token : line.toCharArray() ) {
               if(token == '(') {
                    stack.push(token);
               } else if(token == ')') {
                    if(stack.isEmpty()) {
                         System.out.println("has unbalanced
parenthesis (found closing with no matching opening parenthesis)");
                         return;
                    } else {
                         stack.pop();
                    }
               }
          }

          if(stack.isEmpty()) {
               System.out.println("has balanced parenthesis");
          } else {
               System.out.println("has unbalanced parenthesis
(missing closing parenthesis)");
          }
    }

    private static void palindrome(String line) {
          int stop = line.length() / 2;
```

```java
		boolean odd = line.length() % 2 == 1;
		Stack<Character> stack = new Stack<Character>();

		for(int i = 0; i < stop; i++) {
			stack.push(line.charAt(i));
		}

		System.out.print(line + " ");
		int start = stop;
		if(odd) start++;

		for(int i = start; i < line.length(); i++) {
			if(stack.pop() != line.charAt(i)){
				System.out.println("is not a palindrome");
				return;
			}
		}

		System.out.println("is a palindrome");
	}

	private static void postfix(String line) {
		Stack<Double> tokens = new Stack<Double>();
		Scanner input = new Scanner(line);
		System.out.print(line);

		while(input.hasNextDouble()) {
			tokens.push(input.nextDouble());
		}

		while(input.hasNextLine()){
			for(char op : input.nextLine().toCharArray()) {
				if(op != ' '){
					double rightSide;
					double leftSide = rightSide = 0.0;
					boolean noTokens = false;

					if(!tokens.isEmpty()){
						rightSide = tokens.pop();
						if(!tokens.isEmpty()){
							leftSide = tokens.pop();
						} else {
							noTokens = true;
						}
					} else {
						noTokens = true;
					}

					if(noTokens) {
```

```
                                System.out.println(" - Invalid postfix
notation");
                                return;
                            }

                            if(op == '+') {
                                tokens.push(leftSide + rightSide);
                            } else if(op == '-') {
                                tokens.push(leftSide - rightSide);
                            } else if(op == '*') {
                                tokens.push(leftSide * rightSide);
                            } else if(op == '/') {
                                tokens.push(leftSide / rightSide);
                            }
                        }
                    }
                }

                double last = 0.0;
                if(!tokens.isEmpty()){
                    last = tokens.pop();
                }

                if(tokens.isEmpty()) {
                    System.out.println(" is " + last);
                } else {
                    System.out.println(" is invalid");
                }
            }
}
```

File: Stack.java

```
package edu.bridgeport.mohammad;
public class Stack <T> {
    private T[] elements;
    int insert_index = 0;

    public Stack(){
        constructElementArray(10);
    }

    public Stack(T[] objs) {
        this.elements = objs;

        // find first null index
        for(int i = 0; i < objs.length; i++) {
            if(objs[i] == null) {
                break;
            } else {
```

```
                        insert_index++;
                    }
                }
        }

        public Stack(int size) {
                constructElementArray(size);
        }

        @SuppressWarnings("unchecked")
        private void constructElementArray(int size) {
                elements = (T[]) new Object[size];
                insert_index = 0;
        }

        @SuppressWarnings("unchecked")
        private void addElementLength(int addLength) {
                T[] new_elements = (T[]) new Object[elements.length +
addLength];
                System.arraycopy(elements, 0, new_elements, 0,
elements.length);
                elements = new_elements;
        }

        public boolean push(T obj) {
                if(insert_index >= elements.length){
                    addElementLength(15);
                }
                elements[insert_index++] = obj;
                return true;
        }

        public T pop() {
                if(insert_index > 0) {
                    insert_index--;
                    T obj = elements[insert_index];
                    elements[insert_index] = null;
                    return obj;
                } else {
                    return null;
                }
        }

        public T top() {
                if(insert_index > 0) {
                    return elements[insert_index - 1];
                } else {
                    return null;
                }
```

```
    }

    public void reset() {
          constructElementArray(elements.length);
    }

    public boolean isEmpty() {
          return insert_index == 0;
    }

    public boolean isFull() {
          return insert_index >= elements.length;
    }

    public int size() {
          return insert_index;
    }
}
```

Conclusion

Stacks are powerful and cool. They’re also easy to implement and use. Other then that, this application was pretty easy since we’ve already talked about all of these in class.