Stack

**Source Code:**

public class Stack<E> {  
 private static class Node<E>{  
 E data;  
 Node<E> next;  
  
 public Node(E data) {  
 this.data = data;  
 this.next = null;  
 }  
 }  
  
 private Node<E> top;  
 private int size;  
  
 public Stack(int size) {  
 this.top = null;  
 this.size = size;  
 }  
  
 public boolean isEmpty(){  
 return top == null;  
 }  
  
 // push  
 public void push(E data){  
 Node<E> newNode = new Node<>(data);  
 newNode.next = top;  
 top = newNode;  
 size++;  
 }  
  
 // pop  
 public E pop(){  
 if (isEmpty()){  
 throw new IllegalStateException("Stack is empty.");  
 }  
  
 E popData = top.data;  
 top = top.next;  
 size--;  
 return popData;  
 }  
  
 // peak  
 public E peek(){  
 if(isEmpty()){  
 throw new IllegalStateException("Stack is empty.");  
 }  
 return top.data;  
 }  
  
 public int size(){  
 return size;  
 }  
}

01)

**Source Code:**

import java.util.Scanner;  
  
public class NumberReverser {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter the length of the number sequence: ");  
 int len = scanner.nextInt();  
 System.*out*.print("Enter a number sequence: ");  
 int numberSeq = scanner.nextInt();  
 int reversedSequence = *numberReverser*(len, numberSeq);  
 System.*out*.println("Reversed number sequence: " + reversedSequence);  
 }  
  
 public static int numberReverser(int len, int numberSeq){  
 Stack<Integer> stack = new Stack<>(len);  
  
 while (numberSeq != 0){  
 int num = numberSeq % 10;  
 stack.push(num);  
 numberSeq /= 10;  
 }  
  
 int reversedNumber = 0;  
 int multiplier = 1;  
 while (!stack.isEmpty()){  
 int num = stack.pop();  
 reversedNumber += num \* multiplier;  
 multiplier \*= 10;  
 }  
 return reversedNumber;  
 }  
}

**Output:**



02)

**Source Code:**

public class Calculator {  
 public static void main(String[] args) {  
 java.util.Scanner scanner = new java.util.Scanner(System.*in*);  
 System.*out*.print("Enter a mathematical expression: ");  
 String expression = scanner.nextLine();  
 int result = *calculateExpression*(expression);  
 System.*out*.println("Output: " + result);  
 }  
  
 public static int calculateExpression(String expression) {  
 int len = expression.length();  
 Stack<Integer> numbers = new Stack<>(len);  
 Stack<Character> operators = new Stack<>(len);  
  
 for (int i = 0; i < len; i++) {  
 char character = expression.charAt(i);  
  
 if (Character.*isDigit*(character)) {  
 int num = 0;  
 while (i < expression.length() && Character.*isDigit*(expression.charAt(i))) {  
 num = num \* 10 + (expression.charAt(i) - '0');  
 i++;  
 }  
 i--;  
 numbers.push(num);  
 } else if (character == ' ') {  
 continue;  
 } else if (character == '+' || character == '-' || character == '\*' || character == '/') {  
 while (!operators.isEmpty() && *hasPrecedence*(character, operators.peek())) {  
 char operator = operators.pop();  
 int num2 = numbers.pop();  
 int num1 = numbers.pop();  
 int result = *applyOperation*(num1, num2, operator);  
 numbers.push(result);  
 }  
 operators.push(character);  
 }  
 }  
  
 while (!operators.isEmpty()) {  
 char operator = operators.pop();  
 int num2 = numbers.pop();  
 int num1 = numbers.pop();  
 int result = *applyOperation*(num1, num2, operator);  
 numbers.push(result);  
 }  
 return numbers.pop();  
 }  
  
 private static boolean hasPrecedence(char op1, char op2) {  
 return (op2 == '\*' || op2 == '/') && (op1 == '+' || op1 == '-');  
 }  
  
 private static int applyOperation(int num1, int num2, char operator) {  
 return switch (operator) {  
 case '+' -> num1 + num2;  
 case '-' -> num1 - num2;  
 case '\*' -> num1 \* num2;  
 case '/' -> num1 / num2;  
 default -> throw new IllegalArgumentException("Invalid operator: " + operator);  
 };  
 }  
}

**Output:**

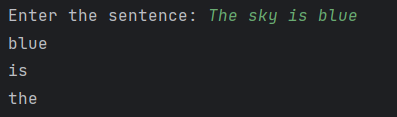


03)

**Source Code:**

import java.util.List;  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class VowelWords {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter the sentence: ");  
 String sentence = scanner.nextLine();  
 *vowelChecker*(sentence.toLowerCase());  
 }  
  
 private static void vowelChecker(String sentence){  
 String[] words = sentence.split(" ");  
 Stack<String> newStack = new Stack<>(sentence.length());  
 for (String word : words){  
 if (word.contains("a") || word.contains("e") || word.contains("i") || word.contains("o") || word.contains("u")){  
 newStack.push(word);  
 } else {  
 continue;  
 }  
 }  
  
 List<String> sortedWords = new ArrayList<>();  
 while (!newStack.isEmpty()){  
 sortedWords.add(newStack.pop());  
 }  
 sortedWords.sort(String::compareTo);  
  
 for (String word : sortedWords){  
 System.*out*.println(word);  
 }  
 }  
}

**Output:**



04)

**Source Code:**

import java.util.Scanner;  
  
public class SentenceReverser {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter the sentence: ");  
 String sentence = scanner.nextLine();  
 *sentenceReverser*(sentence);  
 }  
  
 private static void sentenceReverser(String sentence){  
 Stack<String> newStack = new Stack<>(sentence.length());  
 String[] words = sentence.split(" ");  
 for (String word : words){  
 newStack.push(word);  
 }  
 while (!newStack.isEmpty()){  
 System.*out*.print(newStack.pop() + " ");  
 }  
 }  
}

**Output:**



05)

**Source Code:**

import java.util.Scanner;  
import java.util.Stack;  
  
public class PalindromeChecker {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter the word: ");  
 String word = scanner.nextLine();  
 boolean isPalindrome = *palindromeChecker*(word.toLowerCase());  
  
 if (isPalindrome){  
 System.*out*.println(word + " is a palindrome.");  
 } else {  
 System.*out*.println(word + " is not a palindrome.");  
 }  
 }  
  
 private static boolean palindromeChecker(String word){  
 Stack<Character> newStack = new Stack<>();  
  
 for (int i = 0; i < word.length(); i++) {  
 newStack.push(word.charAt(i));  
 }  
  
 for (int i = 0; i < word.length(); i++) {  
 if (newStack.pop() != word.charAt(i)){  
 return false;  
 }  
 }  
 return true;  
 }  
}

**Output:**





07)

**Source Code:**

import java.util.Scanner;  
  
public class BracketsChecker {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter the mathematical expression: ");  
 String expression = scanner.nextLine();  
 boolean result = *bracketsChecker*(expression);  
 if (result){  
 System.*out*.println("Brackets are correctly balanced");  
 } else {  
 System.*out*.println("Brackets aren't correctly balanced");  
 }  
 }  
  
 private static boolean bracketsChecker(String expression){  
 Stack<Character> newStack = new Stack<>(expression.length());  
 for (char letter : expression.toCharArray()){  
 if (letter == '(' || letter == '{' || letter == '['){  
 newStack.push(letter);  
 } else if (letter == ')' || letter == '}' || letter == ']'){  
 if (newStack.isEmpty()){  
 return false;  
 }  
 char top = newStack.pop();  
 if (letter == ')' && top != '('){  
 return false;  
 } else if (letter == '}' && top != '{'){  
 return false;  
 } else if (letter == ']' && top != '['){  
 return false;  
 }  
 }  
 }  
 return newStack.isEmpty();  
 }  
}

**Output:**





08)

**Source Code:**

import java.util.Scanner;  
  
public class MaxNumberStack {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter number sequence: ");  
 String numberSeq = scanner.nextLine();  
 *maxNumberStack*(numberSeq);  
 }  
  
 private static void maxNumberStack(String numberSeq){  
 Stack<Integer> newStack = new Stack<>(numberSeq.length());  
 int temp;  
 String[] numbers = numberSeq.split(", ");  
 for (String number : numbers){  
 int num = Integer.*parseInt*(number);  
 if (!newStack.isEmpty()){  
 if (newStack.peek() > num){  
 temp = newStack.pop();  
 newStack.push(num);  
 newStack.push(temp);  
 } else {  
 newStack.push(num);  
 }  
 } else if (newStack.isEmpty()) {  
 newStack.push(num);  
 }  
 }  
 System.*out*.println("Highest number: " + newStack.pop());  
 }  
}

**Output:**

