EC4070: Data Structures and Algorithms

LAB 03

FINAL

K.J.M.U.G.S. Eranda Jayasinghe

2021/E/075

SEMESTER 4

EC4070

11.10.2023

Q1.a.

import java.util.Stack;

public class Stack1 {

public static void main(String[] args) {

Stack<String> stack = new Stack<>();

stack.push("cat");

stack.push("dog");

stack.push("ant");

System.out.print("Stack elements: ");

System.out.println(stack);

String topn = stack.peek();

System.out.println("Peek (top): " + topn);

String popn = stack.pop();

System.out.println("Pop: " + popn);

System.out.print("Stack after pop: ");

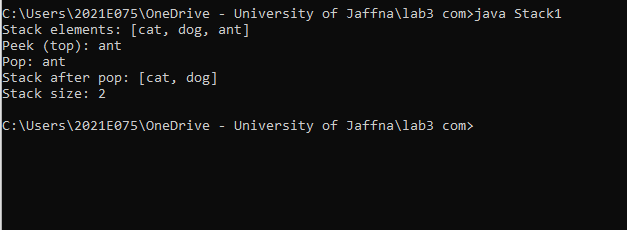
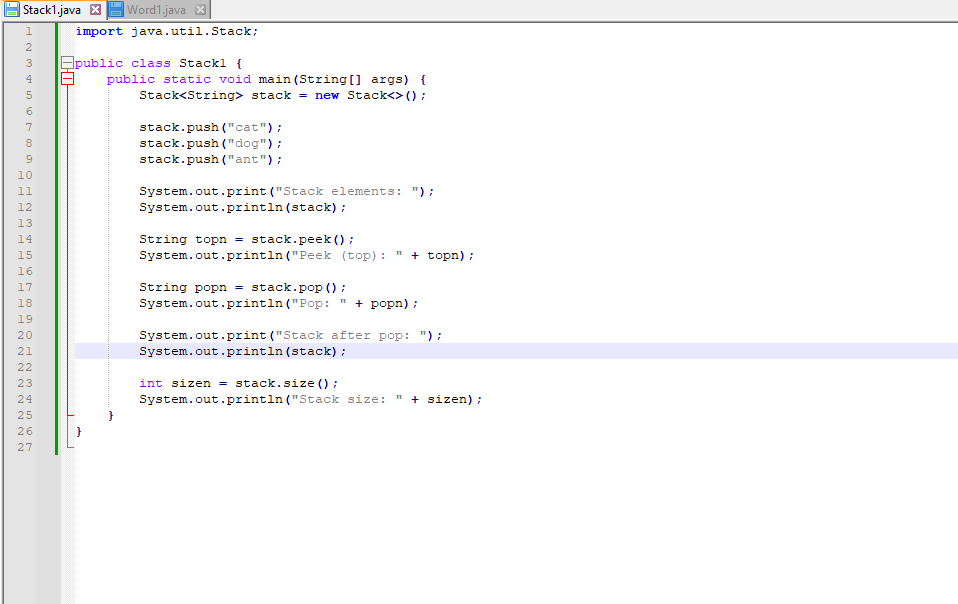
System.out.println(stack);

int sizen = stack.size();

System.out.println("Stack size: " + sizen);

}

}



Q1. b.

import java.util.Scanner;

import java.util.Stack;

public class Word1{

public static void main(String[] args){

Scanner x = new Scanner(System.in);

System.out.print("Enter your word : ");

String word = x.nextLine();

Stack<Character> stack = new Stack<>();

for(int i=0; i < word.length();i++){

stack.push(word.charAt(i));

}

System.out.print("Your reverced word is ");

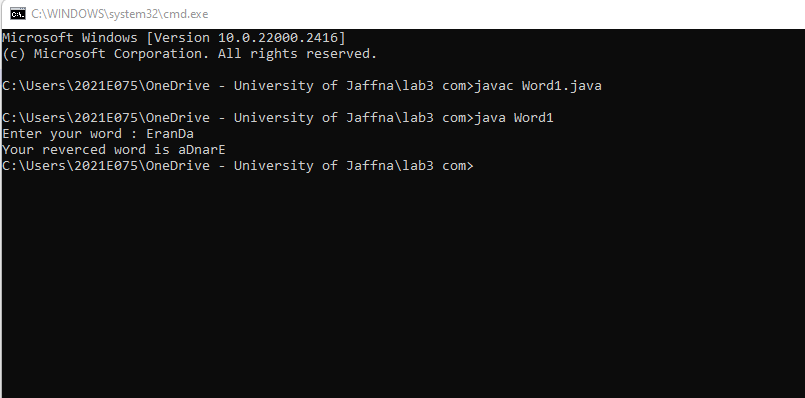
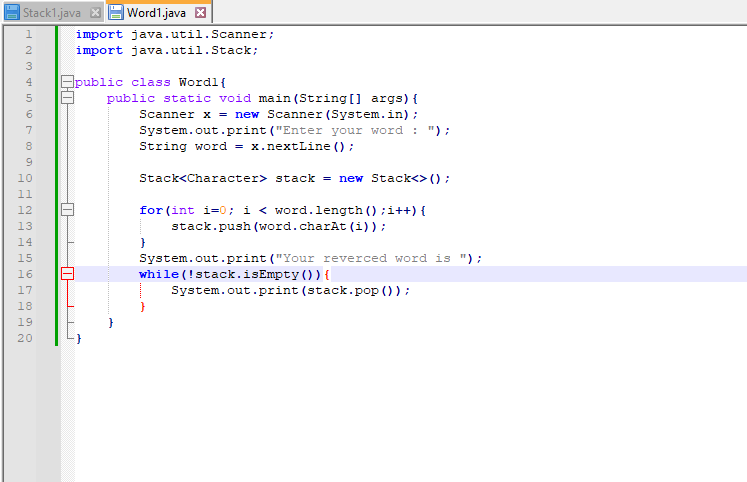
while(!stack.isEmpty()){

System.out.print(stack.pop());

}

}

}



Q1.b2.

import java.util.Scanner;

import java.util.Stack;

public class Delim {

public static void main(String[] args) {

Scanner x = new Scanner(System.in);

System.out.print("Enter delimiter word: ");

String str = x.nextLine();

boolean isMatched = checkDM(str);

if (isMatched) {

System.out.println("Delimiters matched.");

} else {

System.out.println("Delimiters are not matched.");

}

}

public static boolean checkDM(String str) {

Stack<Character> stack = new Stack<>();

for (int i = 0; i < str.length(); i++) {

char c = str.charAt(i);

if (c == '(' || c == '[' || c == '{') {

stack.push(c);

}

else if (c == ')' || c == ']' || c == '}') {

if (stack.isEmpty()) {

return false;

}

char open = stack.pop();

if (!isMatchingPair(open, c)) {

return false;

}

}

}

return stack.isEmpty();

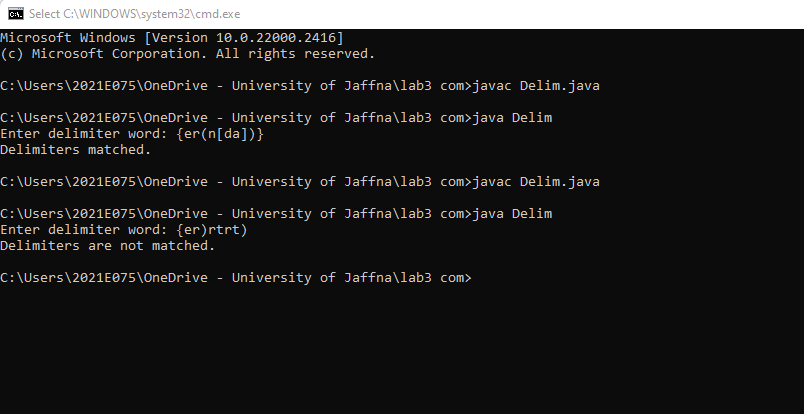
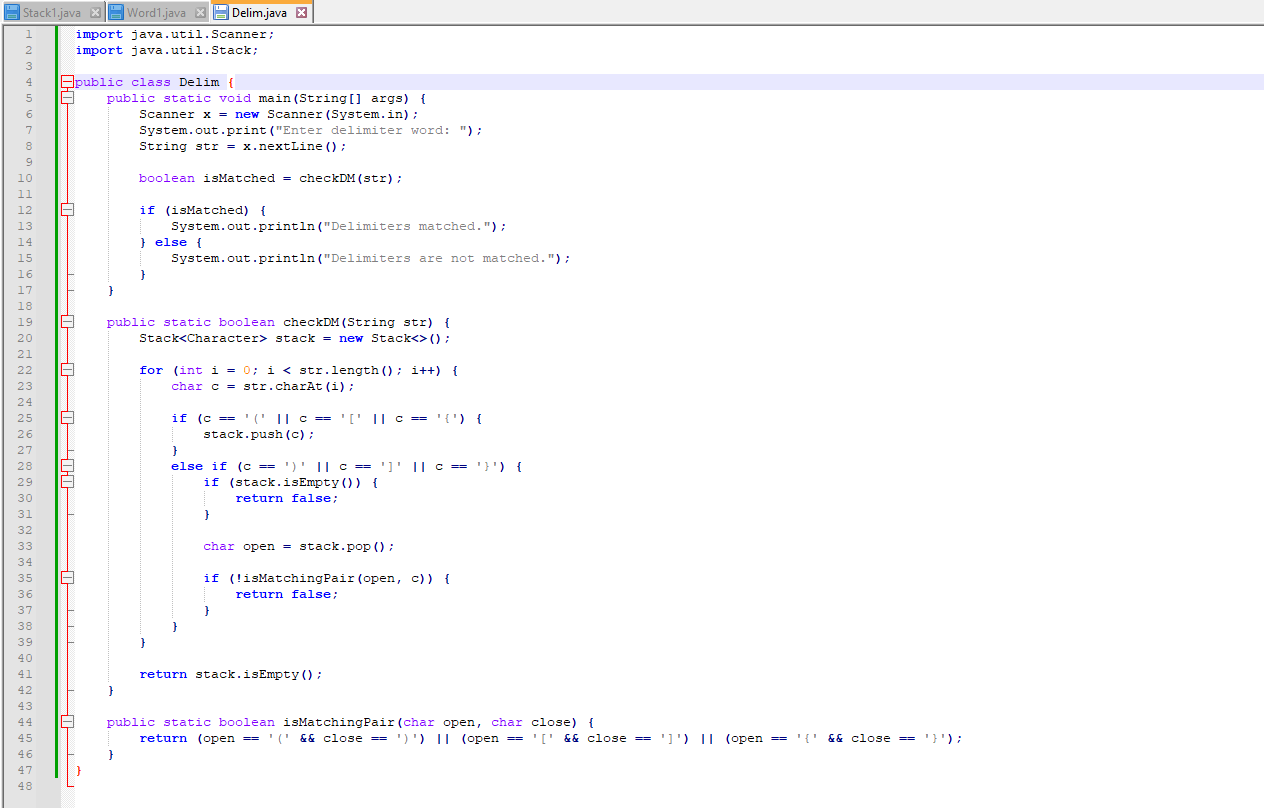
}

public static boolean isMatchingPair(char open, char close) {

return (open == '(' && close == ')') || (open == '[' && close == ']') || (open == '{' && close == '}');

}

}



Q2.a.

import java.util.LinkedList;

import java.util.Queue;

public class Queue1 {

public static void main(String[] args) {

Queue<Integer> queue = new LinkedList<>();

for (int i = 1; i <= 5; i++) {

queue.offer(i);

}

displayQueue(queue);

int removedItem = queue.poll();

System.out.println("Dequeued item: " + removedItem);

displayQueue(queue);

int frontItem = queue.peek();

System.out.println("Front item: " + frontItem);

boolean isEmpty = queue.isEmpty();

System.out.println("Is the queue empty? " + isEmpty);

}

public static void displayQueue(Queue<Integer> queue) {

System.out.print("Queue: ");

for (int item : queue) {

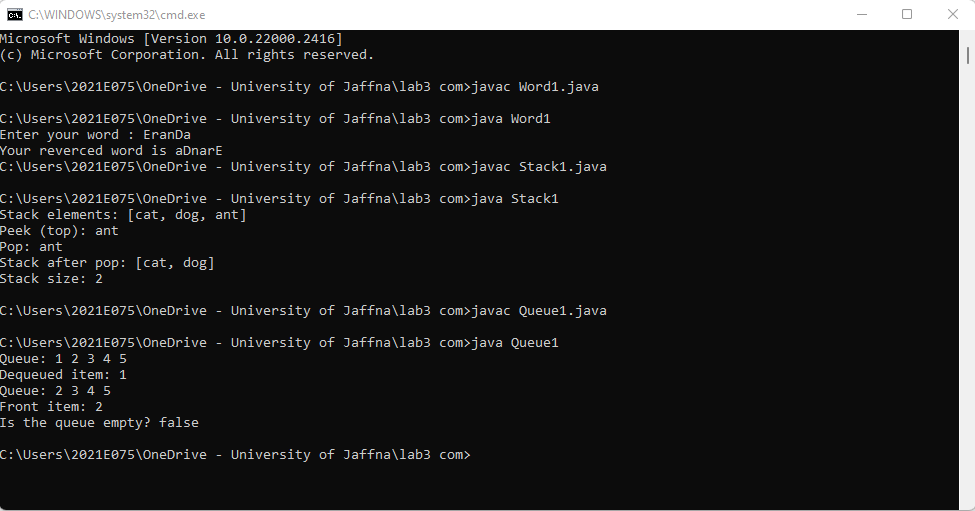
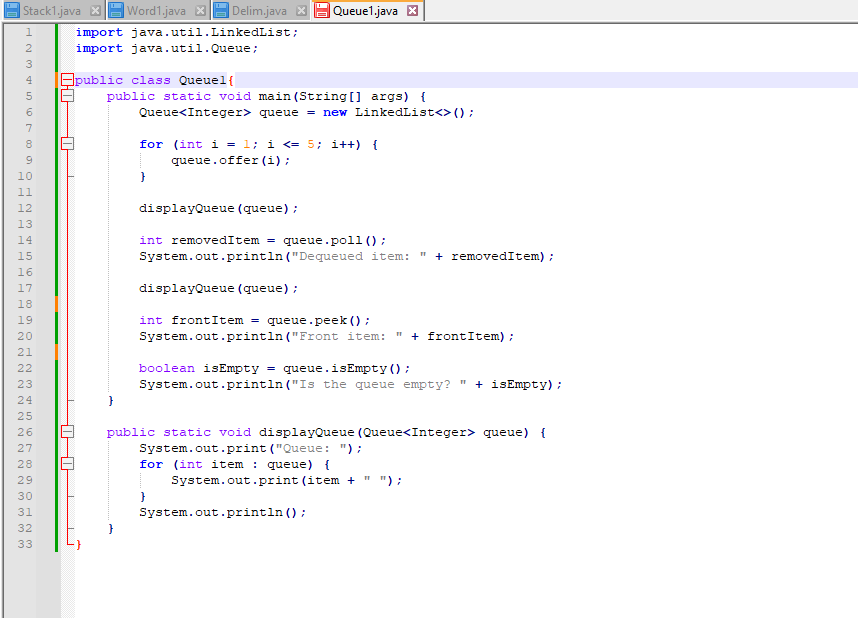
System.out.print(item + " ");

}

System.out.println();

}

}



Q2.b

import java.util.Queue;

import java.util.PriorityQueue;

import java.util.Collections;

public class priority{

public static void main(String[] args){

Queue<String> queue=new PriorityQueue<>();

queue.offer("dog");

queue.offer("cat");

queue.offer("elephant");

queue.offer("bird");

queue.offer("cow");

System.out.println(queue);

Queue<Integer> queue2 = new PriorityQueue<>(Collections.reverseOrder());

queue2.offer(2);

queue2.offer(9);

queue2.offer(101);

queue2.offer(89);

queue2.offer(5);

System.out.println(queue2);

queue.poll();

queue.poll();

System.out.println(queue.poll());

System.out.println(queue.peek());

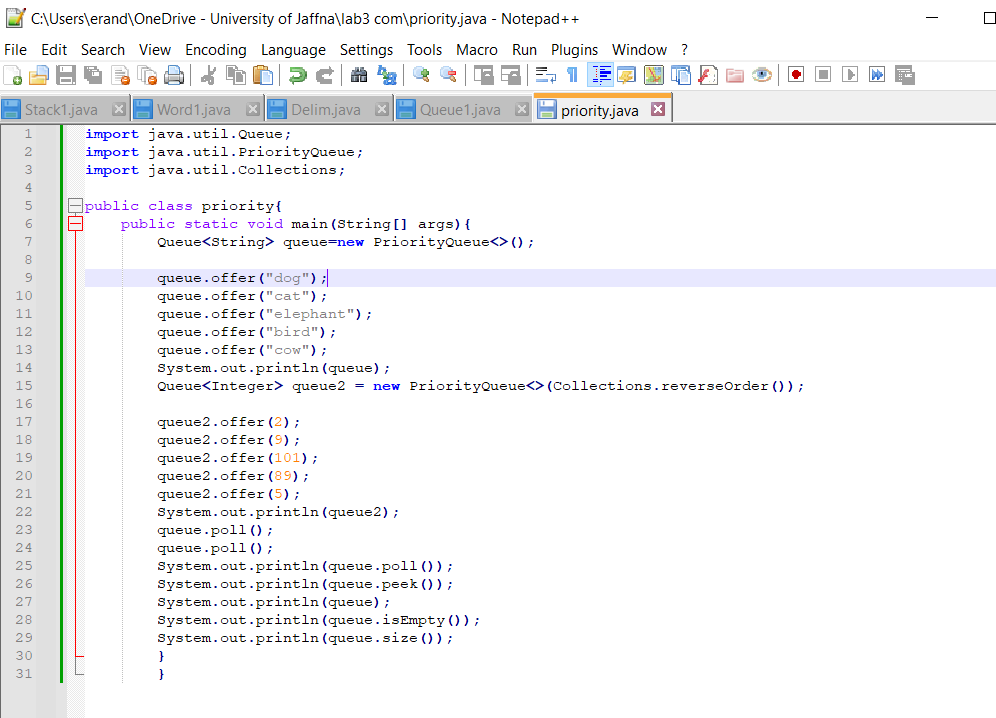
System.out.println(queue);

System.out.println(queue.isEmpty());

System.out.println(queue.size());

}

}



Q3.

import java.util.LinkedList;

public class List1 {

public static void main(String[] args) {

LinkedList<String> list = new LinkedList<>();

list.push("dog");

list.push("cat");

list.push("elephant");

list.offer("cow");

list.push("bird");

list.poll();

list.pop();

System.out.println("Current List: " + list);

list.add(3, "monkey");

list.remove("dog");

System.out.println("Index of 'dog': " + list.indexOf("dog"));

System.out.println("Current List: " + list);

System.out.println("Peek First: " + list.peekFirst());

System.out.println("Peek Last: " + list.peekLast());

list.addFirst("car");

list.addLast("van");

System.out.println("Updated List: " + list);

String first = list.removeFirst();

String last = list.removeLast();

System.out.println("Final List: " + list);

System.out.println("Removed First: " + first);

System.out.println("Removed Last: " + last);

}

}

