5(a)

import java.util.ArrayList;

class BagOfNumbers {

private ArrayList<Integer> bag;

public BagOfNumbers() {

bag = new ArrayList<>();

}

// Add a number to the bag

public void add(int x) {

bag.add(x);

}

// Remove the first occurrence of a number

public void remove(int x) {

if (bag.contains(x)) {

bag.remove(Integer.valueOf(x));

} else {

System.out.println("Number " + x + " not found in the bag.");

}

}

// Count occurrences of a specific number

public int countOccurrences(int x) {

int count = 0;

for (int num : bag) {

if (num == x) {

count++;

}

}

return count;

}

// Check if the bag is empty

public boolean isEmpty() {

return bag.isEmpty();

}

// Get the size of the bag

public int size() {

return bag.size();

}

// Display the bag contents

public void display() {

System.out.println("Bag contents: " + bag);

}

}

public class Main {

public static void main(String[] args) {

BagOfNumbers bag = new BagOfNumbers();

System.out.println("Adding numbers: 5, 10, 5, 20");

bag.add(5);

bag.add(10);

bag.add(5);

bag.add(20);

bag.display();

System.out.println("Count occurrences of 5: " + bag.countOccurrences(5));

System.out.println("Removing 5 from the bag...");

bag.remove(5);

bag.display();

System.out.println("Bag size: " + bag.size());

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

System.out.println("Removing all numbers...");

bag.remove(10);

bag.remove(5);

bag.remove(20);

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

}

}

5(b)

import java.util.Queue;

import java.util.LinkedList;

public class StackUsingTwoQueues {

private Queue<Integer> q1;

private Queue<Integer> q2;

public StackUsingTwoQueues() {

q1 = new LinkedList<>();

q2 = new LinkedList<>();

}

// Push element x onto stack

public void push(int x) {

System.out.println("Pushing element: " + x);

q2.add(x);

while (!q1.isEmpty()) {

q2.add(q1.poll());

}

Queue<Integer> temp = q1;

q1 = q2;

q2 = temp;

}

// Removes the element on top of the stack

public void pop() {

if (q1.isEmpty()) {

System.out.println("Stack is empty. Cannot pop.");

return;

}

int topElement = q1.poll();

System.out.println("Popping element: " + topElement);

}

// Get the top element

public void top() {

if (q1.isEmpty()) {

System.out.println("Stack is empty. No top element.");

return;

}

int topElement = q1.peek();

System.out.println("Top element: " + topElement);

}

// Check if the stack is empty

public void isEmpty() {

boolean empty = q1.isEmpty();

System.out.println("Is stack empty? " + empty);

}

public static void main(String[] args) {

StackUsingTwoQueues stack = new StackUsingTwoQueues();

stack.push(10);

stack.push(20);

stack.push(30);

stack.top();

stack.pop();

stack.top();

stack.isEmpty();

stack.pop();

stack.pop();

stack.isEmpty();

}

}

5©

import java.util.\*;

class RemoveDuplicates {

Node head;

void push(int d) {

Node n = new Node(d);

n.next = head;

head = n;

}

void removeDup() {

HashSet<Integer> set = new HashSet<>();

Node curr = head, prev = null;

while (curr != null) {

if (set.contains(curr.data)) {

prev.next = curr.next;

} else {

set.add(curr.data);

prev = curr;

}

curr = curr.next;

}

}

void printList() {

Node t = head;

while (t != null) {

public static void main(String[] args) {

RemoveDuplicates list = new RemoveDuplicates();

list.push(5); list.push(4); list.push(4); list.push(3);

list.push(2); list.push(2); list.push(1);

System.out.print("Original List: ");

list.printList();

list.removeDup();

System.out.print("\nAfter removing duplicates: ");

list.printList();

}

}