Java programming Lab manual

1. Use eclipse or Netbean platform and acquaint with the various menus, create a test project, add a test class and run it see how you can use auto suggestions, auto fill. Try code formatter and code refactoring like renaming variables, methods and classes. Try debug step by step with a small program of about 10 to 15 lines which contains at least one if else condition and a for loop.

/\* Sample java program to check given number is prime or not \*/

//Importing packages

import java.lang.System;

import java.util.Scanner;

// Creating Class

class Sample\_Program {

// main method

public static void main(String args[]) {

int i,count=0,n;

// creating scanner object

Scanner sc=new Scanner(System.in);

// get input number from user

System.out.print("Enter Any Number : ");

n=sc.nextInt();

// logic to check prime or not

for(i=1;i<=n;i++) {

if(n%i==0) {

count++;

}

}

if(count==2)

System.out.println(n+" is prime");

else

System.out.println(n+" is not prime");

}

}

1. Write a Java program that works as a simple calculator. Use a grid layout to arrange buttons for the digits and for the , -,\*, % operations. Add a text field to display the result. Handle any possible exceptions like divided by zero.

/\* Program to create a Simple Calculator \*/

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class CalculatorApp extends JFrame {

private JTextField inputField;

private JButton[] numberButtons;

private JButton addButton, subtractButton, multiplyButton, divideButton, equalButton, clearButton;

private double firstNum, secondNum, result;

private char operator;

public CalculatorApp() {

initializeUI();

initializeButtons();

addComponentsToFrame();

attachButtonListeners();

}

private void initializeUI() {

setTitle("Simple Calculator");

setSize(300, 400);

setLocationRelativeTo(null);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new GridLayout(5, 4));

}

private void initializeButtons() {

inputField = new JTextField();

inputField.setEditable(false);

inputField.setHorizontalAlignment(SwingConstants.RIGHT);

numberButtons = new JButton[10];

for (int i = 0; i < 10; i++) {

numberButtons[i] = new JButton(String.valueOf(i));

}

addButton = new JButton("+");

subtractButton = new JButton("-");

multiplyButton = new JButton("\*");

divideButton = new JButton("/");

equalButton = new JButton("=");

clearButton = new JButton("C");

}

private void addComponentsToFrame() {

add(inputField);

for (int i = 1; i < 10; i++) {

add(numberButtons[i]);

}

add(numberButtons[0]);

add(addButton);

add(subtractButton);

add(multiplyButton);

add(divideButton);

add(equalButton);

add(clearButton);

}

private void attachButtonListeners() {

for (int i = 0; i < 10; i++) {

int num = i;

numberButtons[i].addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

inputField.setText(inputField.getText() + num);

}

});

}

addButton.addActionListener(e -> setOperator('+'));

subtractButton.addActionListener(e -> setOperator('-'));

multiplyButton.addActionListener(e -> setOperator('\*'));

divideButton.addActionListener(e -> setOperator('/'));

equalButton.addActionListener(e -> calculateResult());

clearButton.addActionListener(e -> {

inputField.setText("");

firstNum = secondNum = result = 0;

operator = '\0';

});

}

private void setOperator(char op) {

firstNum = Double.parseDouble(inputField.getText());

inputField.setText("");

operator = op;

}

private void calculateResult() {

secondNum = Double.parseDouble(inputField.getText());

try {

switch (operator) {

case '+':

result = firstNum + secondNum;

break;

case '-':

result = firstNum - secondNum;

break;

case '\*':

result = firstNum \* secondNum;

break;

case '/':

if (secondNum == 0) {

throw new ArithmeticException("Division by zero is not allowed.");

}

result = firstNum / secondNum;

break;

default:

throw new IllegalArgumentException("Invalid operator");

}

inputField.setText(String.valueOf(result));

} catch (NumberFormatException ex) {

inputField.setText("Error");

} catch (ArithmeticException ex) {

inputField.setText("Error: " + ex.getMessage());

} catch (IllegalArgumentException ex) {

inputField.setText("Error: " + ex.getMessage());

}

}

public static void main(String[] args) {

CalculatorApp calculator = new CalculatorApp();

calculator.setVisible(true);

}

}

1. a) Develop an applet in Java that displays a simple message.

/\* Develop a applet to display the simple message \*/

import java.awt.\*;

import java.applet.\*;

/\*<applet code="FirstApplet" width=400 height=300></applet>\*/

public class FirstApplet extends Applet {

public void paint(Graphics g) {

g.setColor(Color.blue);

Font font = new Font("Arial", Font.BOLD, 16);

g.setFont(font);

g.drawString("This is My First Applet",60,110);

}

}

b) Develop an applet in Java that receives an integer in one text field, and computes its factorial Value and returns it in another text field, when the button named “Compute” is clicked.

/\*\* Develop applet to find factorial of the given number \*/

import java.awt.\*;

import java.applet.\*;

import java.awt.event.\*;

/\*<applet code="FactorialApplet" width=500 height=250>

</applet>\*/

public class FactorialApplet extends Applet implements ActionListener {

Label L1,L2;

TextField T1,T2;

Button B1;

public void init() {

L1=new Label("Enter any Number : ");

add(L1);

T1=new TextField(10);

add(T1);

L2=new Label("Factorial of Num : ");

add(L2);

T2=new TextField(10);

add(T2);

B1=new Button("Compute");

add(B1);

B1.addActionListener(this);

}

public void actionPerformed(ActionEvent e) {

if(e.getSource()==B1)

{

int value=Integer.parseInt(T1.getText());

int fact=factorial(value);

T2.setText(String.valueOf(fact));

}

}

int factorial(int n) {

if(n==0)

return 1;

else

return n\*factorial(n-1);

}

}

1. Write a Java program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num 2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a Number Format Exception. If Num2 were Zero, the program would throw an Arithmetic Exception. Display the exception in a message dialog box.

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class IntegerDivisionApp extends JFrame {

private JTextField num1Field;

private JTextField num2Field;

private JTextField resultField;

private JButton divideButton;

public IntegerDivisionApp() {

initializeUI();

addComponentsToFrame();

attachButtonListener();

}

private void initializeUI() {

setTitle("Integer Division Calculator");

setSize(300, 200);

setLocationRelativeTo(null);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setLayout(new GridLayout(4, 2));

}

private void addComponentsToFrame() {

JLabel num1Label = new JLabel("Num1:");

num1Field = new JTextField();

JLabel num2Label = new JLabel("Num2:");

num2Field = new JTextField();

JLabel resultLabel = new JLabel("Result:");

resultField = new JTextField();

resultField.setEditable(false);

divideButton = new JButton("Divide");

add(num1Label);

add(num1Field);

add(num2Label);

add(num2Field);

add(resultLabel);

add(resultField);

add(new JLabel()); // Empty label for formatting

add(divideButton);

}

private void attachButtonListener() {

divideButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

int num1 = Integer.parseInt(num1Field.getText());

int num2 = Integer.parseInt(num2Field.getText());

if (num2 == 0) {

throw new ArithmeticException("Cannot divide by zero!");

}

int result = num1 / num2;

resultField.setText(String.valueOf(result));

} catch (NumberFormatException ex) {

showMessageDialog("Error: Please enter valid integers for Num1 and Num2.");

} catch (ArithmeticException ex) {

showMessageDialog("Error: " + ex.getMessage());

}

}

});

}

private void showMessageDialog(String message) {

JOptionPane.showMessageDialog(this, message, "Error", JOptionPane.ERROR\_MESSAGE);

}

public static void main(String[] args) {

IntegerDivisionApp app = new IntegerDivisionApp();

app.setVisible(true);

}

}

}

1. Write a Java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

import java.util.\*;

// class for Even Number

class EvenNum implements Runnable {

public int a;

public EvenNum(int a) {

this.a = a;

}

public void run() {

System.out.println("The Thread "+ a +" is EVEN and Square of " + a + " is : " + a \* a);

}

} // class for Odd Number

class OddNum implements Runnable {

public int a;

public OddNum(int a) {

this.a = a;

}

public void run() {

System.out.println("The Thread "+ a +" is ODD and Cube of " + a + " is: " + a \* a \* a);

}

}

// class to generate random number

class RandomNumGenerator extends Thread {

public void run() {

int n = 0;

Random rand = new Random();

try {

for (int i = 0; i < 10; i++) {

n = rand.nextInt(20);

System.out.println("Generated Number is " + n);

// check if random number is even or odd

if (n % 2 == 0) {

Thread thread1 = new Thread(new EvenNum(n));

thread1.start();

}

else {

Thread thread2 = new Thread(new OddNum(n));

thread2.start();

}

// thread wait for 1 second

Thread.sleep(1000);

System.out.println("------------------------------------");

}

}

catch (Exception ex) {

System.out.println(ex.getMessage());

}

}

}

// Driver class

public class MultiThreadRandOddEven {

public static void main(String[] args) {

RandomNumGenerator rand\_num = new RandomNumGenerator();

rand\_num.start();

}

}

1. Write a Java program for the following:i) Create a doubly linked list of elements.ii) Delete a given element from the above list.iii)Display the contents of the list after deletion.

import java.util.\*;

public class DoublyLinkListDemo {

public static void main(String[] args) {

int i,ch,element,position;

LinkedList<Integer> dblList = new LinkedList<Integer>();

System.out.println("1.Insert element at begining");

System.out.println("2.Insert element at end");

System.out.println("3.Insert element at position");

System.out.println("4.Delete a given element");

System.out.println("5.Display elements in the list");

System.out.println("6.Exit");

Scanner sc=new Scanner(System.in);

do {

System.out.print("Choose your choice(1 - 6) :");

ch=sc.nextInt();

switch(ch) {

case 1: // To read element form the user

System.out.print("Enter an element to insert at begining : ");

element=sc.nextInt();

// to add element to doubly linked list at begining

dblList.addFirst(element);

System.out.println("Successfully Inserted");

break;

case 2: // To read element form the user

System.out.print("Enter an element to insert at end : ");

element=sc.nextInt();

// to add element to doubly linked list at end

dblList.addLast(element);

System.out.println("Successfully Inserted");

break;

case 3: // To read position form the user

System.out.print("Enter position to insert element : ");

position=sc.nextInt();

// checks if the position is lessthan or equal to list size.

if(position<=dblList.size()) {

// To read element

System.out.print("Enter element : ");

element=sc.nextInt();

// to add element to doubly linked list at given position

dblList.add(position,element);

System.out.println("Successfully Inserted");

}

else {

System.out.println("Enter the size between 0 to"+dblList.size());

}

break;

case 4: // To read element form the user to remove

System.out.print("Enter element to remove : ");

Integer ele\_rm;

ele\_rm=sc.nextInt();

if (dblList.contains(ele\_rm)){

dblList.remove(ele\_rm);

System.out.println("Successfully Deleted");

Iterator itr=dblList.iterator();

System.out.println("Elements after deleting :"+ele\_rm);

while(itr.hasNext()) {

System.out.print(itr.next()+"<->");

}

System.out.println("NULL");

}

else {

System.out.println("Element not found");

}

break;

case 5: // To Display elements in the list

Iterator itr=dblList.iterator();

System.out.println("Elements in the list :");

while(itr.hasNext()) {

System.out.print(itr.next()+"<->");

}

System.out.println("NULL");

break;

case 6: System.out.println("Program terminated");

break;

default:System.out.println("Invalid choice");

}

}

while(ch!=6);

}

1. Write a Java program that simulates a traffic light. The program lets the user selects one of three lights: red, yellow, or green with radio buttons. On selecting a button, an appropriate message with Stop, Ready, or Go should appear above the buttons in the selected color. Initially, there is no message shown

import javax.swing.\*;

import javax.swing.event.\*;

import java.awt.\*;

import java.awt.event.\*;

class TrafficLightSimulator extends JFrame implements ItemListener {

JLabel lbl1, lbl2;

JPanel nPanel, cPanel;

CheckboxGroup cbg;

public TrafficLightSimulator() {

setTitle("Traffic Light Simulator");

setSize(600,400);

setLayout(new GridLayout(2, 1));

nPanel = new JPanel(new FlowLayout());

cPanel = new JPanel(new FlowLayout());

lbl1 = new JLabel();

Font font = new Font("Verdana", Font.BOLD, 70);

lbl1.setFont(font);

nPanel.add(lbl1);

add(nPanel);

Font fontR = new Font("Verdana", Font.BOLD, 20);

lbl2 = new JLabel("Select Lights");

lbl2.setFont(fontR);

cPanel.add(lbl2);

cbg = new CheckboxGroup();

Checkbox rbn1 = new Checkbox("Red Light", cbg, false);

rbn1.setBackground(Color.RED);

rbn1.setFont(fontR);

cPanel.add(rbn1);

rbn1.addItemListener(this);

Checkbox rbn2 = new Checkbox("Orange Light", cbg, false);

rbn2.setBackground(Color.ORANGE);

rbn2.setFont(fontR);

cPanel.add(rbn2);

rbn2.addItemListener(this);

Checkbox rbn3 = new Checkbox("Green Light", cbg, false);

rbn3.setBackground(Color.GREEN);

rbn3.setFont(fontR);

cPanel.add(rbn3);

rbn3.addItemListener(this);

add(cPanel);

setVisible(true);

// to close the main window

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

// To read selected item

public void itemStateChanged(ItemEvent i) {

Checkbox chk = cbg.getSelectedCheckbox();

String str=chk.getLabel();

char choice=str.charAt(0);

switch (choice) {

case 'R':lbl1.setText("STOP");

lbl1.setForeground(Color.RED);

break;

case 'O':lbl1.setText("READY");

lbl1.setForeground(Color.ORANGE);

break;

case 'G':lbl1.setText("GO");

lbl1.setForeground(Color.GREEN);

break;

}

}

// main method

public static void main(String[] args) {

new TrafficLightSimulator();

}

}

## Write a Java program to create an abstract class named Shape that contains two integers and an empty method named print Area (). Provide three classes named Rectangle, Triangle, and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method print Area () that prints the area of the given shape.

import java.util.\*;

abstract class Shape {

public int x,y;

public abstract void printArea();

}

class Rectangle1 extends Shape {

public void printArea() {

float area;

area= x \* y;

System.out.println("Area of Rectangle is " +area);

}

}

class Triangle extends Shape {

public void printArea() {

float area;

area= (x \* y) / 2.0f;

System.out.println("Area of Triangle is " + area);

}

}

class Circle extends Shape {

public void printArea() {

float area;

area=(22 \* x \* x) / 7.0f;

System.out.println("Area of Circle is " + area);

}

}

public class AreaOfShapes {

public static void main(String[] args) {

int choice;

Scanner sc=new Scanner(System.in);

System.out.println("Menu \n 1.Area of Rectangle \n 2.Area of Traingle \n 3.Area of Circle ");

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

choice=sc.nextInt();

switch(choice) {

case 1: System.out.println("Enter length and breadth for area of rectangle : ");

Rectangle1 r = new Rectangle1();

r.x=sc.nextInt();

r.y=sc.nextInt();

r.printArea();

break;

case 2: System.out.println("Enter bredth and height for area of traingle : ");

Triangle t = new Triangle();

t.x=sc.nextInt();

t.y=sc.nextInt();

t.printArea();

break;

case 3: System.out.println("Enter radius for area of circle : ");

Circle c = new Circle();

c.x = sc.nextInt();

c.printArea();

break;

default:System.out.println("Enter correct choice");

}

}

}

1. Suppose that a table named Table.txt is stored in a text file. The first line in the file is the header, and the remaining lines correspond to rows in the table. The elements are separated by commas. Write a java program to display the table using Labels in Grid Layout.

import java.io.\*;

import java.util.\*;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import javax.swing.event.\*;

class Text\_To\_Table extends JFrame

{

public void convertTexttotable()

{

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(400,300);

GridLayout g = new GridLayout(0, 4);

setLayout(g);

try

{

FileInputStream fis = new FileInputStream("./Table.txt");

Scanner sc = new Scanner(fis);

String[] arrayList;

String str;

while (sc.hasNextLine())

{

str = sc.nextLine();

arrayList = str.split(",");

for (String i : arrayList)

{

add(new Label(i));

}

}

}

catch (Exception ex) {

ex.printStackTrace();

}

setVisible(true);

setTitle("Display Data in Table");

}

}

public class TableText

{

public static void main(String[] args)

{

Text\_To\_Table tt = new Text\_To\_Table();

tt.convertTexttotable();

}

}

1. Write a java program that handles all mouse events and shows the event name at the center of the window when a mouse event is fired (Use Adapter classes).

import javax.swing.\*;

import java.awt.\*;

import javax.swing.event.\*;

import java.awt.event.\*;

class MouseEventPerformer extends JFrame implements MouseListener

{

JLabel l1;

public MouseEventPerformer()

{

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(300,300);

setLayout(new FlowLayout(FlowLayout.CENTER));

l1 = new JLabel();

Font f = new Font("Verdana", Font.BOLD, 20);

l1.setFont(f);

l1.setForeground(Color.BLUE);

add(l1);

addMouseListener(this);

setVisible(true);

}

public void mouseExited(MouseEvent m)

{

l1.setText("Mouse Exited");

}

public void mouseEntered(MouseEvent m)

{

l1.setText("Mouse Entered");

}

public void mouseReleased(MouseEvent m)

{

l1.setText("Mouse Released");

}

public void mousePressed(MouseEvent m)

{

l1.setText("Mouse Pressed");

}

public void mouseClicked(MouseEvent m)

{

l1.setText("Mouse Clicked");

}

public static void main(String[] args) {

MouseEventPerformer mep = new MouseEventPerformer();

}

}

1. Write a java program that loads names and phone numbers from a text file where the data is organized as one line per record and each field ina record are separated by tab(\t). it takes a name or phone number as input and prints the corresponding othe value from the hash table (hint: use hash tables)

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class PhoneBookApp {

private static final String FILE\_NAME = "contacts.txt";

private static final String DELIMITER = "\t";

private static Map<String, String> phoneBook = new HashMap<>();

public static void main(String[] args) {

loadContactsFromFile();

searchContact();

}

private static void loadContactsFromFile() {

try (BufferedReader reader = new BufferedReader(new FileReader(FILE\_NAME))) {

String line;

while ((line = reader.readLine()) != null) {

String[] parts = line.split(DELIMITER);

if (parts.length == 2) {

String name = parts[0].trim();

String phoneNumber = parts[1].trim();

phoneBook.put(name, phoneNumber);

}

}

} catch (IOException e) {

e.printStackTrace();

}

}

private static void searchContact() {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a name or phone number to search: ");

String searchTerm = scanner.nextLine().trim();

String result = phoneBook.get(searchTerm);

if (result != null) {

if (phoneBook.containsKey(searchTerm)) {

System.out.println("Name: " + searchTerm + ", Phone Number: " + result);

} else {

for (Map.Entry<String, String> entry : phoneBook.entrySet()) {

if (entry.getValue().equals(searchTerm)) {

System.out.println("Name: " + entry.getKey() + ", Phone Number: " + searchTerm);

return;

}

}

}

} else {

System.out.println("Contact not found!");

}

}

}

1. Write a java program that correctly implements the producer –consumer problem using the concept of interthread communication.

import java.util.LinkedList;

import java.util.Queue;

class Buffer {

private final int maxSize;

private final Queue<Integer> buffer;

public Buffer(int maxSize) {

this.maxSize = maxSize;

this.buffer = new LinkedList<>();

}

public synchronized void produce(int value) throws InterruptedException {

while (buffer.size() >= maxSize) {

wait(); // Buffer is full, wait for consumer to consume data

}

buffer.offer(value);

System.out.println("Produced: " + value);

notify(); // Notify consumer that data is available

}

public synchronized int consume() throws InterruptedException {

while (buffer.isEmpty()) {

wait(); // Buffer is empty, wait for producer to produce data

}

int value = buffer.poll();

System.out.println("Consumed: " + value);

notify(); // Notify producer that space is available in the buffer

return value;

}

}

class Producer extends Thread {

private final Buffer buffer;

public Producer(Buffer buffer) {

this.buffer = buffer;

}

@Override

public void run() {

try {

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

buffer.produce(i);

Thread.sleep(1000);

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

class Consumer extends Thread {

private final Buffer buffer;

public Consumer(Buffer buffer) {

this.buffer = buffer;

}

@Override

public void run() {

try {

for (int i = 0; i < 5; i++) {

buffer.consume();

Thread.sleep(1500);

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

public class ProducerConsumerApp {

public static void main(String[] args) {

Buffer buffer = new Buffer(3);

Producer producer = new Producer(buffer);

Consumer consumer = new Consumer(buffer);

producer.start();

consumer.start();

try {

producer.join();

consumer.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

1. Write a java program to list all the files in a directory including the files present in all its sub directory

import java.io.File;

public class ListFilesRecursively {

public static void main(String[] args) {

String directoryPath = "e:\\deepthi"; // Replace this with the path to your directory

listFilesRecursively(directoryPath);

}

private static void listFilesRecursively(String directoryPath) {

File directory = new File(directoryPath);

// Check if the provided path is a valid directory

if (!directory.isDirectory()) {

System.out.println("Invalid directory path: " + directoryPath);

return;

}

File[] files = directory.listFiles();

if (files != null) {

for (File file : files) {

if (file.isDirectory()) {

// If it's a directory, recursively list files in it

listFilesRecursively(file.getAbsolutePath());

} else {

// If it's a file, print its absolute path

System.out.println(file.getAbsolutePath());

}

}

}

}

}

1. Write a java program that implements Quick sort algorithm for sorting a list of names in ascending order

import java.util.Arrays;

public class QuickSortNames {

public static void main(String[] args) {

String[] names = {"Alice", "Bob", "David", "Eva", "Cathy", "Frank"};

System.out.println("Unsorted Names: " + Arrays.toString(names));

quickSort(names, 0, names.length - 1);

System.out.println("Sorted Names: " + Arrays.toString(names));

}

public static void quickSort(String[] arr, int low, int high) {

if (low < high) {

int partitionIndex = partition(arr, low, high);

quickSort(arr, low, partitionIndex - 1);

quickSort(arr, partitionIndex + 1, high);

}

}

private static int partition(String[] arr, int low, int high) {

String pivot = arr[high];

int i = low - 1;

for (int j = low; j < high; j++) {

if (arr[j].compareTo(pivot) <= 0) {

i++;

swap(arr, i, j);

}

}

swap(arr, i + 1, high);

return i + 1;

}

private static void swap(String[] arr, int i, int j) {

String temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

1. Write a java program that implements bubble sort algorithm for sorting in descending order and also shows the number of interchanges occurred for the given set of integers.

import java.util.Arrays;

public class BubbleSortDescending {

public static void main(String[] args) {

int[] numbers = {64, 34, 25, 12, 22, 11, 90};

System.out.println("Unsorted Numbers: " + Arrays.toString(numbers));

int interchanges = bubbleSortDescending(numbers);

System.out.println("Sorted Numbers in Descending Order: " + Arrays.toString(numbers));

System.out.println("Number of Interchanges: " + interchanges);

}

public static int bubbleSortDescending(int[] arr) {

int n = arr.length;

int interchanges = 0;

for (int i = 0; i < n - 1; i++) {

boolean swapped = false;

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] < arr[j + 1]) {

swap(arr, j, j + 1);

swapped = true;

interchanges++;

}

}

// If no two elements were swapped in the inner loop, the array is already sorted

if (!swapped) {

break;

}

}

return interchanges;

}

private static void swap(int[] arr, int i, int j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}