**1.**

**1.1) Program to remove all repeated elements from an array**

//Program to remove all repeated elements from an array  
import java.util.\***;**public class prblm1 {  
 public static <**T**> ArrayList<**T**> removeDuplicates(ArrayList<**T**> list) {  
 ArrayList<**T**> newList = new ArrayList<**T**>()**;** for (**T** element : list) {  
 if (!newList.contains(element)) {  
 newList.add(element)**;** }  
 }  
 return newList**;** }  
  
 public static void main(String args[]) {  
 Scanner scanner = new Scanner(System.*in*)**;** ArrayList<Integer> list = new ArrayList<>()**;** System.*out*.print("Enter the number of elements in the array: ")**;** int length = scanner.nextInt()**;** System.*out*.println("Enter the elements of the array:")**;** for (int i = **0;** i < length**;** i++) {  
 int element = scanner.nextInt()**;** list.add(element)**;** }  
 scanner.close()**;** System.*out*.println("ArrayList with duplicates: " + list)**;** ArrayList<Integer> newList = *removeDuplicates*(list)**;** System.*out*.println("ArrayList with duplicates removed: " + newList)**;** }  
}

**1.2) Write a Java program to find the common elements between two arrays of integers.**

//Write a Java program to find the common elements between two arrays of integers.  
import java.util.\***;**public class prblm2 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*)**;** System.*out*.print("Enter the number of elements in the first array: ")**;** int length1 = scanner.nextInt()**;** int[] array1 = new int[length1]**;** System.*out*.println("Enter the elements of the first array:")**;** for (int i = **0;** i < length1**;** i++) {  
 array1[i] = scanner.nextInt()**;** }  
  
 System.*out*.print("Enter the number of elements in the second array: ")**;** int length2 = scanner.nextInt()**;** int[] array2 = new int[length2]**;** System.*out*.println("Enter the elements of the second array:")**;** for (int i = **0;** i < length2**;** i++) {  
 array2[i] = scanner.nextInt()**;** }  
  
 scanner.close()**;** List<Integer> commonElements = new ArrayList<>()**;** for (int num : array1) {  
 if (Arrays.*binarySearch*(array2**,** num) >= **0**) {  
 commonElements.add(num)**;** }  
 }  
  
 System.*out*.println("Common Elements: " + commonElements)**;** }  
}

**2.**

**2. 1) Java Program to Count Number of Duplicate Words in String**

import java.util.Scanner**;**public class StringDuplicateWords {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*)**;** System.*out*.print("Enter a string: ")**;** String string = scanner.nextLine()**;** int count**;** // Converts the string into lowercase  
 string = string.toLowerCase()**;** // Split the string into words using built-in function  
 String words[] = string.split(" ")**;** System.*out*.println("Duplicate words in the given string: ")**;** for (int i = **0;** i < words.length**;** i++) {  
 count = **1;** for (int j = i + **1;** j < words.length**;** j++) {  
 if (words[i].equals(words[j])) {  
 count++**;** // Set words[j] to "0" to avoid printing visited word  
 words[j] = "0"**;** }  
 }  
  
 // Displays the duplicate word if count is greater than 1  
 if (count > **1** && !words[i].equals("0"))  
 System.*out*.println(words[i])**;** }  
  
 scanner.close()**;** }  
}

**2.2) How to Check if the String Contains 'e' in umbrella**

//How to Check if the String Contains 'e' in umbrella  
  
public class containsE {  
 public static void main(String[] args) {  
   
 String str = "umbrella"**;** // Check if the string contains 'e'  
 boolean containsE = str.contains("e")**;** if (containsE) {  
 System.*out*.println("The string contains 'e'.")**;** } else {  
 System.*out*.println("The string does not contain 'e'.")**;** }  
 }  
}

**3.**

**3.1)Java Program to Reverse a String.**

import java.util.Scanner**;**public class StringReverse {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*)**;** System.*out*.print("Enter a string: ")**;** String input = scanner.nextLine()**;** String reversed = *reverseString*(input)**;** System.*out*.println("Reversed string: " + reversed)**;** scanner.close()**;** }  
  
 public static String reverseString(String str) {  
 StringBuilder reversed = new StringBuilder()**;** for (int i = str.length() - **1;** i >= **0;** i--) {  
 reversed.append(str.charAt(i))**;** }  
 return reversed.toString()**;** }  
}

**3.2) Write a Java program to check that String is palindrome or not.**

import java.util.Scanner**;**public class PalindromeString{  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*)**;** System.*out*.print("Enter a string: ")**;** String input = scanner.nextLine()**;** boolean isPalindrome = *checkPalindrome*(input)**;** if (isPalindrome) {  
 System.*out*.println("The string is a palindrome.")**;** } else {  
 System.*out*.println("The string is not a palindrome.")**;** }  
  
  
 scanner.close()**;** }  
  
 public static boolean checkPalindrome(String str) {  
 int left = **0;** int right = str.length() - **1;** while (left < right) {  
 if (str.charAt(left) != str.charAt(right)) {  
 return false**;** // Characters don't match, not a palindrome  
 }  
 left++**;** right--**;** }  
  
 return true**;** // All characters matched, it is a palindrome  
 }  
}

**4. A Company manufactures Vehicles, which could be a Helicopter, a Car, or a Train depending on the customer’s demand. Each Vehicle instance has a method called move, which prints on the console the nature of movement of the vehicle. For example, the Helicopter Flies in Air, the Car Drives on Road and the Train Runs on Track. Write a program that accepts input from the user on the kind of vehicle the user wants to order, and the system should print out nature of movement. Implement all Java coding best practices to implement this program.**

//A Company manufactures Vehicles, which could be a Helicopter, a Car, or a Train depending on the customer’s demand. Each Vehicle instance has a method called move, which prints on the console the nature of movement of the vehicle. For example, the Helicopter Flies in Air, the Car Drives on Road and the Train Runs on Track. Write a program that accepts input from the user on the kind of vehicle the user wants to order, and the system should print out nature of movement. Implement all Java coding best practices to implement this program.  
  
import java.util.Scanner**;**// Vehicle class (parent class)  
abstract class Vehicle {  
 public abstract void move()**;**}  
  
// Helicopter class (subclass of Vehicle)  
class Helicopter extends Vehicle {  
 @Override  
 public void move() {  
 System.*out*.println("The Helicopter flies in the air.")**;** }  
}  
  
// Car class (subclass of Vehicle)  
class Car extends Vehicle {  
 @Override  
 public void move() {  
 System.*out*.println("The Car drives on the road.")**;** }  
}  
  
// Train class (subclass of Vehicle)  
class Train extends Vehicle {  
 @Override  
 public void move() {  
 System.*out*.println("The Train runs on the track.")**;** }  
}  
  
public class VehicleType {  
 public static void main(String[] args) {  
 // Accept input from the user  
 Scanner scanner = new Scanner(System.*in*)**;** System.*out*.print("Enter the type of vehicle (Helicopter/Car/Train): ")**;** String vehicleType = scanner.nextLine()**;** scanner.close()**;** // Create the corresponding vehicle object based on user input  
 Vehicle vehicle**;** switch (vehicleType.toLowerCase()) {  
 case "helicopter":  
 vehicle = new Helicopter()**;** break**;** case "car":  
 vehicle = new Car()**;** break**;** case "train":  
 vehicle = new Train()**;** break**;** default:  
 System.*out*.println("Invalid vehicle type.")**;** return**;** // Exit the program if an invalid vehicle type is entered  
 }  
  
 // Print the nature of movement of the selected vehicle  
 vehicle.move()**;** }  
}

import java.util.Scanner;

// Define the Vehicle interface

interface Vehicle {

void move();

}

// Implement the Helicopter class

class Helicopter implements Vehicle {

@Override

public void move() {

System.out.println("The Helicopter flies in the air.");

}

}

// Implement the Car class

class Car implements Vehicle {

@Override

public void move() {

System.out.println("The Car drives on the road.");

}

}

// Implement the Train class

class Train implements Vehicle {

@Override

public void move() {

System.out.println("The Train runs on the track.");

}

}

// Main class

class VehicleFactory {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Welcome to the Vehicle Factory!");

while (true) {

System.out.println("What kind of vehicle would you like to order?");

System.out.println("1. Helicopter");

System.out.println("2. Car");

System.out.println("3. Train");

System.out.println("4. Exit");

int choice = scanner.nextInt();

if (choice == 4) {

System.out.println("Thank you for using the Vehicle Factory. Goodbye!");

break;

}

Vehicle vehicle = createVehicle(choice);

if (vehicle != null) {

vehicle.move();

} else {

System.out.println("Invalid choice. Please try again.");

}

}

scanner.close();

}

// Create the appropriate vehicle based on user choice

private static Vehicle createVehicle(int choice) {

switch (choice) {

case 1:

return new Helicopter();

case 2:

return new Car();

case 3:

return new Train();

default:

return null;

}

}

}

**5. We have to calculate the percentage of marks obtained in three subjects (each out of 100) by student A and in four subjects (each out of 100) by student B. Create an abstract class 'Marks' with an abstract method 'getPercentage'. It is inherited by two other classes 'A' and 'B' each having a method with the same name which returns the percentage of the students. The constructor of student A takes the marks in three subjects as its parameters and the marks in four subjects as its parameters for student B. Create an object for each of the two classes and print the percentage of marks for both the students.**

// We have to calculate the percentage of marks obtained in three subjects  
// (each out of 100) by student A and in four subjects (each out of 100) by student B.  
// Create an abstract class 'Marks' with an abstract method 'getPercentage'.  
// It is inherited by two other classes 'A' and 'B' each having a method with the  
// same name which returns the percentage of the students. The constructor of student A  
// takes the marks in three subjects as its parameters and the marks in four subjects as  
// its parameters for student B. Create an object for each of the two classes and print  
// the percentage of marks for both the students.  
  
import java.util.\***;**abstract class Marks {  
 public abstract double getPercentage()**;**}  
  
class A extends Marks {  
  
 private double subject1Marks**;** private double subject2Marks**;** private double subject3Marks**;** public A(double subject1Marks**,** double subject2Marks**,** double subject3Marks) {  
 this.subject1Marks = subject1Marks**;** this.subject2Marks = subject2Marks**;** this.subject3Marks = subject3Marks**;** }  
  
 @Override  
 public double getPercentage() {  
 double totalMarks = subject1Marks + subject2Marks + subject3Marks**;** return (totalMarks / **300**) \* **100;** }  
}  
  
class B extends Marks {  
 private double subject1Marks**;** private double subject2Marks**;** private double subject3Marks**;** private double subject4Marks**;** public B(double subject1Marks**,** double subject2Marks**,** double subject3Marks**,** double subject4Marks) {  
 this.subject1Marks = subject1Marks**;** this.subject2Marks = subject2Marks**;** this.subject3Marks = subject3Marks**;** this.subject4Marks = subject4Marks**;** }  
  
 @Override  
 public double getPercentage() {  
 double totalMarks = subject1Marks + subject2Marks + subject3Marks + subject4Marks**;** return (totalMarks / **400**) \* **100;** }  
}  
  
public class CalculatePercentage {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*)**;** System.*out*.println("Enter Marks for A: ")**;** int a = scanner.nextInt()**;** int b = scanner.nextInt()**;** int c = scanner.nextInt()**;** System.*out*.println("Enter Marks for B :")**;** int d = scanner.nextInt()**;** int e = scanner.nextInt()**;** int f = scanner.nextInt()**;** int g = scanner.nextInt()**;** A studentA = new A(a**,** b**,** c)**;** B studentB = new B(d**,** e**,** f**,** g)**;** System.*out*.println("Percentage of marks for Student A: " + studentA.getPercentage())**;** System.*out*.println("Percentage of marks for Student B: " + studentB.getPercentage())**;** }  
}

**6. Write the following code in your editor below:  
A class named Arithmetic with a method named add that takes 2 integers as parameters and returns an integer denoting their sum.  
A class named Adder that inherits from a superclass named Arithmetic. The main method in the Tester class should print the following: SAMPLE O/P:My superclass is: Arithmetic  
42 13 20**

//A class named Arithmetic with a method named add that takes 2 integers as parameters and returns an integer denoting their sum.

// A

// class named

// Adder that

// inherits from

// a superclass

// named Arithmetic.

// The main

// method in

// the Tester

// class should

// print the following:

// SAMPLE O/P:

// My superclass is:Arithmetic 42 13 20

class Arithmetic {

    public int add(int a, int b) {

        return a + b;

    }

}

class Adder extends Arithmetic {

}

public class ClassSuperclass {

    public static void main(String[] args) {

        Adder adder = new Adder();

        // Print superclass name

        System.out.println("My superclass is: " + adder.getClass().getSuperclass().getName());

        // Perform addition using the add() method

        int sum = adder.add(42, 13);

        System.out.println("42 + 13 = " + sum);

        // Perform addition using the add() method from the superclass

        int sum2 = adder.add(20, 0);

        System.out.println("20 + 0 = " + sum2);

    }

}

**7. You are required to compute the power of a number by implementing a calculator. Create a class My Calculator which consists of a single method long power (int, int). This method takes two integers n and p, as parameters and finds (n)p. If either or is negative, then the method must throw an exception which says " n or p should not be negative”. Also, if both and are zero, then the method must throw an exception which says "n or p should not be negative”.**

class MyCalculator {

    public long power(int n, int p) throws Exception {

        if (n < 0 || p < 0) {

            throw new Exception("n or p should not be negative");

        } else if (n == 0 && p == 0) {

            throw new Exception("n and p should not be zero");

        } else {

            return (long) Math.pow(n, p);

        }

    }

}

public class CalculatorException {

    public static void main(String[] args) {

        MyCalculator calculator = new MyCalculator();

        try {

            long result1 = calculator.power(2, 3);

            System.out.println("Result 1: " + result1); // 2^3 = 8

            long result2 = calculator.power(0, 0);

            System.out.println("Result 2: " + result2);

        } catch (Exception e) {

            System.out.println("Exception: " + e.getMessage());

        }

    }

}

**8. You are given a phone book that consists of people's names and their phone number. After that you will be given some person's name as query. For each query, print the phone number of that person. Use HashMap to implement it.The first line will have an integer denoting the number of entries in the phone book. Each entry consists of two lines: a name and the corresponding phone number.  
After these, there will be some queries. Each query will contain a person's name. Read the queries until end-of-file.  
Constraints:  
A person's name consists of only lower-case English letters and it may be in the format 'first-name last-name' or in the format 'first-name'. Each phone number has exactly 8 digits without any leading zeros.For each case, print "Not found" if the person has no entry in the phone book. Otherwise, print the person's name and phone number.**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class Solution {

/\* input Scanner for reading input for the program \*/

private static Scanner inputScanner;

public static void main(String[] args) {

/\* Create a phone book Map \*/

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

/\* Create the input Scanner instance \*/

inputScanner = new Scanner(System.in);

/\* Read the number of input phone numbers \*/

int numFriends = inputScanner.nextInt();

/\* Go to the next line of input \*/

inputScanner.nextLine();

/\* Loop thru the number of friends, reading the name and phone number, and adding to the phoneBook \*/

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

/\* Read the name of the friend \*/

String name = inputScanner.nextLine();

/\* Read the phone number \*/

String phone = inputScanner.nextLine();

/\* Put the name and phone number into the phoneBook \*/

phoneBook.put(name, phone);

}

/\* Loop while there is still more input data \*/

while (inputScanner.hasNext()) {

/\* Read the person to search for a phone number \*/

String inputName = inputScanner.nextLine();

/\* Does the name exist in the phone Book \*/

if (phoneBook.containsKey(inputName)) {

/\* Yes, print out the name and phone number \*/

System.out.println(inputName + "=" + phoneBook.get(inputName));

} else {

/\* Print out error message if not found \*/

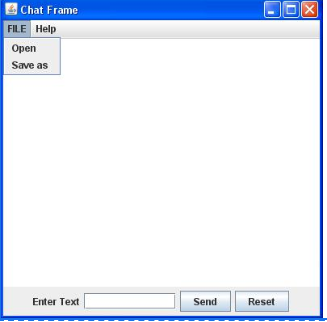
System.out.println("Not found");

}

}

}

}

9. 

import javax.swing.\***;**import java.awt.\***;**class ChatFrame {  
 public static void main(String args[]) {  
  
 // Creating the Frame  
 JFrame frame = new JFrame("Chat Frame")**;** frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*)**;** frame.setSize(**400, 400**)**;** // Creating the MenuBar and adding components  
 JMenuBar mb = new JMenuBar()**;** JMenu m1 = new JMenu("File")**;** JMenu m2 = new JMenu("Help")**;** mb.add(m1)**;** mb.add(m2)**;** JMenuItem m11 = new JMenuItem("Open")**;** JMenuItem m22 = new JMenuItem("Save as")**;** m1.add(m11)**;** m1.add(m22)**;** // Creating the panel at bottom and adding components  
 JPanel panel = new JPanel()**;** // the panel is not visible in output  
 JLabel label = new JLabel("Enter Text")**;** JTextField tf = new JTextField(**10**)**;** // accepts upto 10 characters  
 JButton send = new JButton("Send")**;** JButton reset = new JButton("Reset")**;** panel.add(label)**;** // Components Added using Flow Layout  
 panel.add(tf)**;** panel.add(send)**;** panel.add(reset)**;** // Text Area at the Center  
 JTextArea ta = new JTextArea()**;** // Adding Components to the frame.  
 frame.getContentPane().add(BorderLayout.*SOUTH***,** panel)**;** frame.getContentPane().add(BorderLayout.*NORTH***,** mb)**;** frame.getContentPane().add(BorderLayout.*CENTER***,** ta)**;** frame.setVisible(true)**;** }  
}

10. 

import java.awt.EventQueue;

import javax.swing.\*;

import java.awt.\*;

public class NumberAddition {

private JFrame frame;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

NumberAddition window = new NumberAddition();

window.frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the application.

\*/

public NumberAddition() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

frame = new JFrame();

frame.setBounds(100, 100, 450, 300);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.getContentPane().setLayout(null);

JLabel label1=new JLabel("Number Addition");

label1.setForeground(Color.BLUE);

label1.setBounds(20, 10, 100, 30);

frame.getContentPane().add(label1);

//for number 1

JLabel first=new JLabel("First Number: ");

first.setBounds(10, 40, 120, 20);

frame.getContentPane().add(first);

JTextField f1=new JTextField();

f1.setBounds(140, 40, 140, 20);

frame.getContentPane().add(f1);

//for number 2

JLabel second=new JLabel("Second Number:");

second.setBounds(10, 65, 120, 20);

frame.getContentPane().add(second);

JTextField s1=new JTextField();

s1.setBounds(140, 65, 140, 20);

frame.getContentPane().add(s1);

//for result

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

result.setBounds(10, 90, 120, 20);

frame.getContentPane().add(result);

JTextField r1=new JTextField();

r1.setBounds(140, 90, 140, 20);

frame.getContentPane().add(r1);

//add button

JButton add=new JButton("ADD");

add.setBounds(90,120, 80,20);

frame.getContentPane().add(add);

//clear button

JButton clear=new JButton("CLEAR");

clear.setBounds(200,120, 80,20);

frame.getContentPane().add(clear);

//exit button

JButton exit=new JButton("Exit");

exit.setBounds(250, 200, 80, 20);

frame.add(exit);

}

}

**11.**

**11.1) Write a Java program that takes a number as input and prints its multiplication table up to 10. Test Data:  
Input a number: 8  
Expected Output :  
8 x 1 = 8  
8 x 2 = 16  
8 x 3 = 24  
...  
8 x 10 = 80**

import java.util.Scanner;

public class TableExample

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

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

//reading a number whose table is to be print

int num=sc.nextInt();

//loop start execution form and execute until the condition i<=10 becomes false

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

{

//prints table of the entered number

System.out.println(num+" \* "+i+" = "+num\*i);

}

}

}

**11.2) Write a java program to check that given number is prime or not.**

public class PrimeExample{

public static void main(String args[]){

int i,m=0,flag=0;

int n=3;//it is the number to be checked

m=n/2;

if(n==0||n==1) {

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

}

else {

for(i=2;i<=m;i++){

if(n%i==0){

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

flag=1;

break;

}

}

if(flag==0) { System.out.println(n+" is prime number"); }

}//end of else

}

}

**12. Write a Java program to display the pattern like a diamond.  
Input number of rows (half of the diamond) :7 Expected Output :  
  
  
\*   
\*\*\*   
\*\*\*\*\*   
\*\*\*\*\*\*\*   
\*\*\*\*\*\*\*\*\*   
\*\*\*\*\*\*\*\*\*\*\*   
\*\*\*\*\*\*\*\*\*\*\*\*\*   
\*\*\*\*\*\*\*\*\*\*\*   
\*\*\*\*\*\*\*\*\*   
\*\*\*\*\*\*\*   
\*\*\*\*\*   
\*\*\*   
\***

import java.util.Scanner;

public class Exercise21 {

public static void main(String[] args)

{

int i,j,r;

System.out.print("Input number of rows (half of the diamond) : ");

Scanner in = new Scanner(System.in);

r = in.nextInt();

for(i=0;i<=r;i++)

{

for(j=1;j<=r-i;j++)

System.out.print(" ");

for(j=1;j<=2\*i-1;j++)

System.out.print("\*");

System.out.print("\n");

}

for(i=r-1;i>=1;i--)

{

for(j=1;j<=r-i;j++)

System.out.print(" ");

for(j=1;j<=2\*i-1;j++)

System.out.print("\*");

System.out.print("\n");

}

}

}

**13.**

**13.1) Write Java Program to find the transpose of a given matrix .**

import java.util.Scanner;

public class MatrixTransposeExample2

{

public static void main(String args[])

{

int i, j;

System.out.println("Enter total rows and columns: ");

Scanner s = new Scanner(System.in);

int row = s.nextInt();

int column = s.nextInt();

int array[][] = new int[row][column];

System.out.println("Enter matrix:");

for(i = 0; i < row; i++)

{

for(j = 0; j < column; j++)

{

array[i][j] = s.nextInt();

System.out.print(" ");

}

}

System.out.println("The above matrix before Transpose is ");

for(i = 0; i < row; i++)

{

for(j = 0; j < column; j++)

{

System.out.print(array[i][j]+" ");

}

System.out.println(" ");

}

System.out.println("The above matrix after Transpose is ");

for(i = 0; i < column; i++)

{

for(j = 0; j < row; j++)

{

System.out.print(array[j][i]+" ");

}

System.out.println(" ");

}

}

}

**13.2) Write Java Program to find the number of the words in the given text file.**

import java.io.BufferedReader;

import java.io.FileReader;

public class CountWordFile

{

public static void main(String[] args) throws Exception {

String line;

int count = 0;

//Opens a file in read mode

FileReader file = new FileReader("data.txt ");

BufferedReader br = new BufferedReader(file);

//Gets each line till end of file is reached

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

//Splits each line into words

String words[] = line.split("");

//Counts each word

count = count + words.length;

}

System.out.println("Number of words present in given file: " + count);

br.close();

}

}

14. 

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*; // Using Swing's components and containers

// A Swing application extends from javax.swing.JFrame

public class SwingCalculator extends JFrame {

private JTextField tfDisplay;

private int result = 0; // the result so far

private String numberInStr = ""; // the number entered as String

private char previousOpr = ' '; // the previous operator

private char currentOpr = ' '; // the current operator

// Constructor to setup the UI components and event handlers

public SwingCalculator() {

// TODO: Setup the UI

// ......

// Number buttons listener (inner class)

class NumberBtnListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent evt) {

numberInStr += evt.getActionCommand();

tfDisplay.setText(numberInStr);

}

}

// Operator buttons listener (inner class)

class OprBtnListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent evt) {

previousOpr = currentOpr; // save

currentOpr = evt.getActionCommand().charAt(0);

// TODO: Processing logic

// ......

}

}

}

**15. Write a Java Program to iterate ArrayList using for-loop, iterator, and advance for-loop. Insert 3 Array List. Input 20 30 40 Output:  
  
  
iterator Loop:  
20  
30  
40  
Advanced For Loop:  
20  
30  
40  
For Loop:  
20  
30  
40**

import java.util.ArrayList;

import java.util.Iterator;

class lab\_exam10 {

public static void main(String[] args) {

ArrayList<Integer> a1 = new ArrayList<Integer>();

a1.add(20);

a1.add(30);

a1.add(40);

System.out.println("Using For Loop:");

for(int i=0;i< a1.size();i++){

System.out.print(a1.get(i)+" ");

}

System.out.println("\nUsing Iterator:");

Iterator it = a1.iterator();

while (it.hasNext())

System.out.print(it.next() + " ");

System.out.println("\nUsing Advanced For Loop:");

for (Integer i : a1){

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

}

}

}

**16. Write a Java Program to count the number of words in a string using HashMap.Output:  
Input :Enter String: "This this is is done by Saket Saket";  
{Saket=2, by=1, this=1, This=1, is=2, done=1}**

import java.util.HashMap;

import java.util.Map;

public class lab\_exam8 {

public static void main(String[] args) {

String str = "This this is is done by Saket Saket";

Map<String, Integer> hashMap = new HashMap<>();

String[] words = str.split(" ");

for (String word : words) {

Integer integer = hashMap.get(word);

if (integer == null)

hashMap.put(word, 1);

else {

hashMap.put(word, integer + 1);

}

}

System.out.println(hashMap);

}

}

**17. Write a program to read 10 string from console and then print the sorted strings on console (Use String Class). 2) combine two string 3)reverse first string and display it .**

import java.util.Scanner;

public class lab\_exam {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String str;String arr1[]=new String[10];

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

System.out .println("Enter String: "+i);

str=sc.next();

arr1[i]=str;

char arr[] = str.toCharArray();

char temp;

for(int a=0;a<arr.length;a++){

for (int b=a+1;b<arr.length;b++){

if (arr[b] < arr[a]) {

temp = arr[a];

arr[a] = arr[b];

arr[b] = temp;

}

}

}

System.out .println(arr);

}

System.out.println(arr1[0]+" "+arr1[1]);

String nstr="";

char ch;

for (int i=0; i<arr1[0].length(); i++){

ch= arr1[0].charAt(i);

nstr= ch+nstr;

}

System.out .println("Reversed First String:"+ nstr);

}

}

**18. Write a program to implement following inheritance. Accept data for 5 persons and display the name of employee having salary greater than 5000.  
  
Class Name: Person  
Member variables:  
Name, age  
  
Class Name: Employee  
Member variables:  
Designation, salary**

class Person{

String name;

int age;

Person(int age, String name) {

this.name = name;

this.age = age;

}

}

class Employee extends Person{

String designation;

int salary;

Employee(String designation, String name, int age, int salary) {

super(age, name);

this.designation = designation;

this.salary = salary;

if (salary>5000){

System.out.println(name);

}

}

}

public class employeeSalary{

public static void main (String [] args){

Employee emp = new Employee("Developer","Naman",19,45000);

Employee emp1 = new Employee("Manager","Karan",25,4000);

Employee emp2 = new Employee("Accountant","Akash",26,5000);

Employee emp3 = new Employee("Developer","Yash",22,6000);

Employee emp4 = new Employee("Data Scientist","Sahil",23,2000);

}

}

**19. Implementing “Multiple Inheritance”. Create a two interfaces Account containing methods set() and display() And interface Person containing methods store() and disp(). Derive a class Customer from Person and Account. Accept the name, account number, balance and display all the information related to account along with the interest.**

import java.util.Scanner;

// Account interface

interface Account {

void set(String accountNumber, double balance);

void display();

}

// Person interface

interface Person {

void store(String name);

void disp();

}

// Customer class implementing Account and Person interfaces

class Customer implements Account, Person {

private String name;

private String accountNumber;

private double balance;

@Override

public void set(String accountNumber, double balance) {

this.accountNumber = accountNumber;

this.balance = balance;

}

@Override

public void display() {

System.out.println("Account Number: " + accountNumber);

System.out.println("Balance: $" + balance);

}

@Override

public void store(String name) {

this.name = name;

}

@Override

public void disp() {

System.out.println("Name: " + name);

}

public void calculateInterest(double interestRate) {

double interest = balance \* interestRate / 100;

balance += interest;

System.out.println("Interest: $" + interest);

System.out.println("Updated Balance: $" + balance);

}

}

// Main class

class MultipleInheritanceExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Customer customer = new Customer();

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

String name = scanner.nextLine();

customer.store(name);

System.out.print("Enter account number: ");

String accountNumber = scanner.nextLine();

System.out.print("Enter balance: $");

double balance = scanner.nextDouble();

scanner.nextLine();

customer.set(accountNumber, balance);

System.out.println("\nCustomer Information:");

customer.disp();

customer.display();

System.out.print("\nEnter interest rate (%): ");

double interestRate = scanner.nextDouble();

System.out.println("\nAccount Information with Interest:");

customer.calculateInterest(interestRate);

scanner.close();

}

}

**20. "Write a program, to implement the following hierarchy. Displays information of each class the rectangle represents the classes. The classes Movie and MusicVideo inherits all the members of the class VideoTape.**  
"



class VideoTape {

private String title;

private int length;

public VideoTape(String title, int length) {

this.title = title;

this.length = length;

}

public String getTitle() {

return title;

}

public int getLength() {

return length;

}

@Override

public String toString() {

return "VideoTape: " + title + " (" + length + " minutes)";

}

}

class Movie extends VideoTape {

private String rating;

public Movie(String title, int length, String rating) {

super(title, length);

this.rating = rating;

}

public String getRating() {

return rating;

}

@Override

public String toString() {

return "Movie: " + getTitle() + " (" + getLength() + " minutes, rated " + rating + ")";

}

}

class MusicVideo extends VideoTape {

private String artist;

public MusicVideo(String title, int length, String artist) {

super(title, length);

this.artist = artist;

}

public String getArtist() {

return artist;

}

@Override

public String toString() {

return "MusicVideo: " + getTitle() + " (" + getLength() + " minutes, by " + artist + ")";

}

}

// Test the classes

VideoTape tape1 = new VideoTape("The Secret Life of Pets", 90);

System.out.println(tape1); // prints "VideoTape: The Secret Life of Pets (90 minutes)"

Movie movie1 = new Movie("Jurassic Park", 127, "PG-13");

System.out.println(movie1); // prints "Movie: Jurassic Park (127 minutes, rated PG-13)"

MusicVideo musicVideo1 = new MusicVideo("Roar", 3, "Katy Perry");

System.out.println(musicVideo1); // prints "MusicVideo: Roar (3 minutes, by KatyPerry)"

**21. Write a Java program to create a class called "Student" with a name, grade, and courses attributes, and methods to add and remove courses.**

import java.util.ArrayList;

import java.util.List;

public class Student {

private String name;

private int grade;

private List<String> courses;

public Student(String name, int grade) {

this.name = name;

this.grade = grade;

this.courses = new ArrayList<>();

}

public String getName() {

return name;

}

public int getGrade() {

return grade;

}

public void addCourse(String course) {

courses.add(course);

}

public void removeCourse(String course) {

courses.remove(course);

}

public List<String> getCourses() {

return courses;

}

public static void main(String[] args) {

// Creating a student object

Student student = new Student("John Doe", 10);

// Adding courses

student.addCourse("Math");

student.addCourse("Science");

student.addCourse("English");

// Removing a course

student.removeCourse("Science");

// Getting the student's name, grade, and courses

System.out.println("Student Name: " + student.getName());

System.out.println("Grade: " + student.getGrade());

System.out.println("Courses: " + student.getCourses());

}

}

**22. Write a Java program to create a class known as Person with methods called getFirstName() and getLastName(). Create a subclass called Employee that adds a new method named getEmployeeId() and overrides the getLastName() method to include the employee's job title.**

public class Person {

private String firstName;

private String lastName;

public Person(String firstName, String lastName) {

this.firstName = firstName;

this.lastName = lastName;

}

public String getFirstName() {

return firstName;

}

public String getLastName() {

return lastName;

}

}

// Employee.java

// Child class Employee

public class Employee extends Person {

private int employeeId;

private String jobTitle;

public Employee(String firstName, String lastName, int employeeId, String jobTitle) {

super(firstName, lastName);

this.employeeId = employeeId;

this.jobTitle = jobTitle;

}

public int getEmployeeId() {

return employeeId;

}

@Override

public String getLastName() {

return super.getLastName() + ", " + jobTitle;

}

}

// Main.java

// Main class

public class Main {

public static void main(String[] args) {

Employee employee1 = new Employee("Kortney", "Rosalee", 4451, "HR Manager");

System.out.println(employee1.getFirstName() + " " + employee1.getLastName() + " (" + employee1.getEmployeeId() + ")");

Employee employee2 = new Employee("Junior", "Philipa", 4452, "Software Manager");

System.out.println(employee2.getFirstName() + " " + employee2.getLastName() + " (" + employee2.getEmployeeId() + ")");

}

}

**23. Write a Java program to find the length of the longest consecutive elements sequence from an unsorted array of integers.  
Sample array: [49, 1, 3, 200, 2, 4, 70, 5]  
The longest consecutive elements sequence is [1, 2, 3, 4, 5], therefore the program will return its length 5.**

import java.io.\*;

import java.util.\*;

class GFG {

static int findLongestConseqSubseq(int arr[], int n)

{

// Sort the array

Arrays.sort(arr);

int ans = 0, count = 0;

ArrayList<Integer> v = new ArrayList<Integer>();

v.add(arr[0]);

// Insert repeated elements

// only once in the vector

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

if (arr[i] != arr[i - 1])

v.add(arr[i]);

}

// Find the maximum length

// by traversing the array

for (int i = 0; i < v.size(); i++) {

// Check if the current element is

// equal to previous element +1

if (i > 0 && v.get(i) == v.get(i - 1) + 1)

count++;

else

count = 1;

// Update the maximum

ans = Math.max(ans, count);

}

return ans;

}

// Driver code

public static void main(String[] args)

{

int arr[] = { 1, 9, 3, 10, 4, 20, 2 };

int n = arr.length;

System.out.println(

"Length of the Longest "

+ "contiguous subsequence is "

+ findLongestConseqSubseq(arr, n));

}

}

**24. Create a class Student with attributes roll no, name, age and course. Initialize values through parameterized constructor. If age of student is not in between 15 and 21 then generate user-defined exception "AgeNotWithinRangeException". If name contains numbers or special symbols raise exception "NameNotValidException". Define the two exception classes.**

package PS;

import java.io.\*;

import java.util.\*;

class AgeNotWithInRangeException extends Exception

{

public String validage()

{

return ("Age is not between 15 and 21 … Please ReEnter the Age");

}

}

class NameNotValidException extends Exception

{

public String validname()

{

return("Name is not Valid … Please ReEnter the Name");

}

}

class Student

{

int roll,age;

String name,course;

Student()

{

roll=0;

name=null;

age=0;

course=null;

}

Student(int r,String n,int a,String c)

{

roll=r;

course=c;

int l,temp=0;

l=n.length();

for(int i=0;i<l;i++)

{

char ch;

ch=n.charAt(i);

if(ch<'A' || ch>'Z' && ch<'a' || ch>'z')

temp=1;

}

/\*———-Checking Name——————–\*/

try

{

if(temp==1)

throw new NameNotValidException();

else

name=n;

}

catch(NameNotValidException e2)

{

System.out.println(e2);

}

/\*———-Checking Age——————–\*/

try

{

if(a>=15 && a<=21)

age=a;

else

throw new AgeNotWithInRangeException();

}

catch(AgeNotWithInRangeException e1)

{

System.out.println(e1);

}

}

void display()

{

System.out.println("roll Name Age Course");

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

System.out.println(roll+" "+name+" "+age+" "+course);

}

}

class StudentDemo

{

public static void main(String args[])throws IOException

{

Scanner sc = new Scanner(System.in);

int r,a;

String n,c;

System.out.println("Enter roll,name,age,course");

r=sc.nextInt();

sc.nextLine();

n=sc.nextLine();

a=sc.nextInt();

sc.nextLine();

c=sc.nextLine();

Student s=new Student(r,n,a,c);

s.display();

}

}

25. 

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

class Color\_Demo extends Frame

{

Label lbl1,lbl2,lbl3,lbl4,lbl5;

public Color\_Demo()

{

lbl1 = new Label("All The Best");

lbl1.setForeground(Color.red);

add(lbl1);

lbl2 = new Label("All The Best");

lbl2.setForeground(Color.magenta);

add(lbl2);

lbl3 = new Label("All The Best");

lbl3.setForeground(Color.blue);

add(lbl3);

lbl4 = new Label("All The Best");

lbl4.setForeground(Color.green);

add(lbl4);

lbl5 = new Label("All The Best");

lbl5.setForeground(Color.cyan);

add(lbl5);

setVisible(true);

setSize(400, 300);

setLayout(new FlowLayout());

setBackground(Color.gray);

}

public void paint(Graphics g)

{

g.setColor(Color.magenta);

g.drawString("All The Best",100,100);

g.setColor(Color.cyan);

g.drawString("All The Best",150,150);

g.setColor(Color.red);

g.drawString("All The Best",200,200);

g.setColor(Color.black);

g.drawString("All The Best",250,250);

}

public static void main(String[] args)

{

new Color\_Demo();

}

}