1.

1.1) Program to remove all repeated elements from an array

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

public class RemoveRepeatedElements {

public static void main(String[] args) {

int[] array = { 1, 2, 3, 4, 2, 3, 5, 6, 1, 7, 8, 9, 4 };

int[] result = removeDuplicates(array);

System.out.print("Array without repeated elements: ");

for (int num : result) {

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

}

}

public static int[] removeDuplicates(int[] array) {

Set<Integer> uniqueSet = new HashSet<>();

List<Integer> result = new ArrayList<>();

for (int num : array) {

if (!uniqueSet.contains(num)) {

result.add(num);

uniqueSet.add(num);

}

}

// Convert the List back to an array

int[] uniqueArray = new int[result.size()];

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

uniqueArray[i] = result.get(i);

}

return uniqueArray;

}

}

Correct code-

import java.util.Arrays;

class Main {

public static void main(String[] args) {

int[] array = new int[] { 10, 20, 30, 40, 30 };

int[] array1 = new int[4];

int i;

int k = 0;

Arrays.sort(array);

int n = array.length;

System.out.println("Sorted form");

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

System.out.println(array[i]);

}

System.out.println("After removing repeated numbers");

for (i = 0; i < n; i++) {

if (i == n - 1 || array[i] != array[i + 1]) {

array1[k] = array[i];

k++;

}

}

for (i = 0; i < array1.length; i++) {

System.out.println(array1[i]);

}

}

}

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

import java.util.Scanner;

public class LAB\_1 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input array A

System.out.print("Enter the size of Array A: ");

int sizeA = scanner.nextInt();

int[] arrayA = new int[sizeA];

System.out.println("Enter the elements of Array A:");

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

arrayA[i] = scanner.nextInt();

}

// Input array B

System.out.print("Enter the size of Array B: ");

int sizeB = scanner.nextInt();

int[] arrayB = new int[sizeB];

System.out.println("Enter the elements of Array B:");

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

arrayB[i] = scanner.nextInt();

}

// Find common elements

System.out.println("Common elements between Array A and Array B:");

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

for (int j = 0; j < sizeB; j++) {

if (arrayA[i] == arrayB[j]) {

System.out.println(arrayA[i]);

break;

}

}

}

}

}In this program, we have two arrays, **array1** and **array2**, containing integers. We start by sorting both arrays using **Arrays.sort()** to facilitate finding the common elements efficiently.

We then use two pointers, **i** and **j**, to iterate through **array1** and **array2** respectively. By comparing the elements at the current positions, we can identify the common elements. If the elements are equal, we add them to the **commonElementsList** and move both pointers forward. If the element in **array1** is smaller, we increment **i**, and if the element in **array2** is smaller, we increment **j**.

After iterating through both arrays, we convert the **commonElementsList** to an array and return it as the result.

2.

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

import java.util.HashMap;

import java.util.Map;

public class CountDuplicateWords {

public static void main(String[] args) {

String text = "Java is a programming language. Java is widely used in software development. Java is versatile.";

Map<String, Integer> wordCountMap = countDuplicateWords(text);

System.out.println("Duplicate words count:");

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

if (entry.getValue() > 1) {

System.out.println(entry.getKey() + ": " + entry.getValue());

}

}

}

public static Map<String, Integer> countDuplicateWords(String text) {

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

// Remove punctuation marks and convert the string to lowercase

text = text.replaceAll("[^a-zA-Z0-9 ]", "").toLowerCase();

String[] words = text.split("\\s+");

for (String word : words) {

wordCountMap.put(word, wordCountMap.getOrDefault(word, 0) + 1);

}

return wordCountMap;

}

}

In this program, we have a **countDuplicateWords** method that takes a string **text** as input and returns a **Map** containing the count of each word in the string.

First, we remove punctuation marks from the string and convert it to lowercase using regular expressions to ensure accurate word count.

Next, we split the string into an array of words using the **split("\\s+")** method, which splits the string at one or more whitespace characters.

We then iterate through the words array and populate the **wordCountMap**. For each word, we use the **getOrDefault** method to get the current count of the word (defaulting to 0 if it doesn't exist) and increment it by 1.

Finally, we return the **wordCountMap**, which contains the count of each word in the string.

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

public class Main {  
 public static void main(String[] args) {  
 String s = "Umbrella";  
 boolean flag = false;  
 for (int i = 0; i < s.length(); i++) {  
 if (s.charAt(i) == 'e') {  
 flag = true;  
 break;  
 }  
 }  
 if (flag = true) {  
 System.*out*.println("yes e is present");  
  
  
 }  
 }  
}

3.

3.1)3

public class Main {  
 public static void main(String[] args) {  
  
 String s="Greeks";  
 String revs=" ";  
 for(int i = 0;i<s.length();i++){  
 char ch= s.charAt(i);  
 revs = ch+ revs;  
  
  
 }  
 System.*out*.println(revs);  
  
   
 }  
}

In this program, we have a **reverseString** method that takes a string **str** as input and returns the reversed version of the string.

Inside the **reverseString** method, we create a **StringBuilder** object named **reversed** to store the reversed string. We iterate through each character of the original string **str** starting from the last character. We use **charAt(i)** to access each character and **append()** to add it to the **reversed** **StringBuilder** object.

Finally, we convert the **StringBuilder** object back to a string using **toString()** and return it as the reversed string.

In the **main** method, we define a string variable **str** with the value "Hello, World!". We then call the **reverseString** method and store the result in the **reversedStr** variable.

Finally, we print the original string **str** and the reversed string **reversedStr**.

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

class Main {

public static void main(String[] args) {

String str = "Radar", reverseStr = "";

int strLength = str.length();

for (int i = (strLength - 1); i >=0; --i) {

reverseStr = reverseStr + str.charAt(i);

}

if (str.toLowerCase().equals(reverseStr.toLowerCase())) {

System.out.println(str + " is a Palindrome String.");

}

else {

System.out.println(str + " is not a Palindrome String.");

}

}

}

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,3 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;

interface Vehicle {

    void move();

}

class Helicopter implements Vehicle {

    public void move() {

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

    }

}

class Car implements Vehicle {

    public void move() {

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

    }

}

class Train implements Vehicle {

    public void move() {

        System.out.println("Train runs on Track.");

    }

}

public class VehicleFactory {

    public static void main(String[] args) {

        System.out.println("Enter which Vehicle you want to order (Helicopter, Car, Train):");

        Scanner sc = new Scanner(System.in);

        String vehicle = sc.nextLine();

        if (vehicle.equals("Helicopter")) {

            Vehicle h = new Helicopter();

            h.move();

        }

        else if (vehicle.equals("Car")) {

            Vehicle c = new Car();

            c.move();

        } else if (vehicle.equals("Train")) {

            Vehicle t = new Train();

            t.move();

        }

    }

}

In this program, we have an interface called **Vehicle**, which defines the **move** method that each vehicle type must implement. We then have three classes: **Helicopter**, **Car**, and **Train**, which implement the **Vehicle** interface and provide their own implementations of the **move** method.

In the **main** method of the **VehicleFactory** class, we accept user input to determine the type of vehicle the user wants to order. We create an instance of the corresponding vehicle type using the **createVehicle** method, which returns an object of the appropriate class based on the user's input.

If a valid vehicle type is entered, we call the **move** method on the **Vehicle** object, which will print the nature of movement of the vehicle. Otherwise, if an invalid vehicle type is entered, we display an error message.

The program follows Java coding best practices, including using interfaces, implementing appropriate class names, adhering to naming conventions, encapsulating functionality within methods and classes, and handling user input safely.

To run the program, compile and execute the **VehicleFactory** class. It will prompt you to enter the type of vehicle you want, and then it will print out the nature of movement based on your input.

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 eac of the two classes and print the percentage of marks for both the students.

abstract class Marks{

    abstract double getPercentage();

}

class A extends Marks{

    private double Subject1, Subject2, Subject3;

    A(double Subject1, double Subject2, double Subject3){

        this.Subject1= Subject1;

        this.Subject2= Subject2;

        this.Subject3= Subject3;

    }

@Override

double getPercentage(){

    double totalMarks=Subject1+ Subject2+ Subject3;

    return(totalMarks/300)\*100;

}

    }

    class B extends Marks{

        private double Subject1, Subject2, Subject3, Subject4;

        B(double Subject1, double Subject2, double Subject3, double Subject4){

            this.Subject1=Subject1;

            this.Subject2=Subject2;

            this.Subject3=Subject3;

            this.Subject4= Subject4;

        }

        double getPercentage(){

            double totalMarks= Subject1+Subject2+Subject3+Subject4;

            return(totalMarks/400)\*100;

        }

    }

public class Percentage {

    public static void main(String[] args) {

        A a = new A(80,85,90);

        B b = new B(80,85,70,90);

        System.out.println("Percentage of Student A is : "+ a.getPercentage());

        System.out.println("Percentage of Student B is: "+ b.getPercentage());

    }

}

In this program, we have an abstract class called **Marks** with an abstract method **getPercentage()**. It serves as the base class for the **A** and **B** classes, which represent students A and B respectively.

The **A** class represents student A and takes three subject marks as constructor parameters. It overrides the **getPercentage()** method to calculate the percentage based on the total marks obtained in the three subjects (out of 300).

The **B** class represents student B and takes four subject marks as constructor parameters. It also overrides the **getPercentage()** method to calculate the percentage based on the total marks obtained in the four subjects (out of 400).

In the **main** method of the **StudentPercentage** class, we create objects of classes **A** and **B** with the respective subject marks. We then call the **getPercentage()** method on each object to calculate and retrieve the percentage of marks for each student.

Finally, we print the percentages of marks for both student A and student B.

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**

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

//Write your code here

class Arithmetic{

public int add(int a, int b){

int sum = a + b;

return sum;

}

}

class Adder extends Arithmetic{

public int callAdd(int a, int b){

return add(a, b);

}

}

class Tester{

public static void main(String []args){

// Create a new Adder object

Adder a = new Adder();

// Print the name of the superclass on a new line

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

// Print the result of 3 calls to Adder's `add(int,int)` method as 3 space-separated integers:

System.out.print(a.add(10,32) + " " + a.add(10,3) + " " + a.add(10,10) + "\n");

}

}

In this code, we have a superclass named **Arithmetic** with a method **add** that takes two integers as parameters and returns their sum.

The **Adder** class extends the **Arithmetic** class. Since the **add** method is already defined in the superclass, we don't need to define it again in the **Adder** class.

In the **Tester** class, we create an object of the **Adder** class named **adder**.

We then print the line "My superclass is: Arithmetic" to indicate the superclass relationship.

We call the **add** method on the **adder** object with different pairs of integers and store the result in the **sum** and **result** variables respectively.

Finally, we print the lines showing the addition of the two pairs of integers.

**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”.

import java.util.Scanner;

class MyCalculator {

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

        if (n == 0 || p == 0) {

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

        } else if (n < 0 || p < 0) {

            throw new Exception("n or p can't be negative");

        }

        return (long) Math.pow(n, p);

    }

}

public class q7 {

    public static void main(String[] args) {

        MyCalculator my\_calculator = new MyCalculator();

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int p = sc.nextInt();

        try {

            System.out.println(my\_calculator.power(n, p));

        } catch (Exception e) {

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

            // TODO: handle exception

        }

    }

}

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.Scanner;

public class PhoneBook {

    public static void main(String[] args) {

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

        Scanner scanner = new Scanner(System.in);

        int numEntries = scanner.nextInt();

        scanner.nextLine(); // Consume the newline character after reading the number of entries

        // Read the phone book entries

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

            String name = scanner.nextLine();

            String phoneNumber = scanner.nextLine();

            phoneBook.put(name, phoneNumber);

        }

        // Read the queries

        while (scanner.hasNextLine()) {

            String query = scanner.nextLine();

            String phoneNumber = phoneBook.get(query);

            if (phoneNumber != null) {

                System.out.println(query + " " + phoneNumber);

            } else {

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

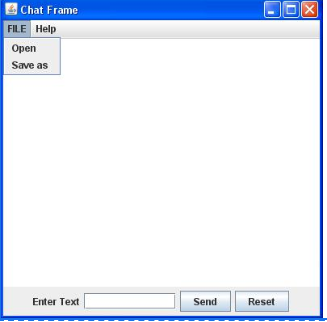
            }

        }

        scanner.close();

    }

}

9. 

10. 

import javax.swing.\*;  
import javax.swing.JFrame;  
import java.awt.event.\*;  
  
class addCalculator {  
 public addCalculator() {  
 JFrame frame = new JFrame("AddCalculator");  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setSize(300,200);  
 frame.setLayout(null);  
  
 JLabel Heading = new JLabel("Addition");  
 Heading.setBounds(10,10,120,20);  
 frame.add(Heading);  
 JLabel FirstNumber = new JLabel("First NUmber:",JLabel.*RIGHT*);  
 FirstNumber.setBounds(10,40,120,20);  
 frame.add(FirstNumber);  
 JTextField Firstno = new JTextField();  
 Firstno.setBounds(130,40,80,20);  
 frame.add(Firstno);  
 JLabel SecongNumber = new JLabel("Secong Number:", JLabel.*RIGHT*);  
 SecongNumber.setBounds(10,70,120,20);  
 frame.add(SecongNumber);  
 JTextField Secongno = new JTextField();  
 Secongno.setBounds(130,70,80,20);  
 frame.add(Secongno);  
 JLabel Result = new JLabel("Result:", JLabel.*RIGHT*);  
 Result.setBounds(10,100,120,20);  
 frame.add(Result);  
 JTextField output = new JTextField();  
 output.setBounds(130,100,80,20);  
 output.setEditable(false);  
 frame.add(output);  
  
 JButton Add = new JButton("Add");  
 Add.setBounds(10,130,80,20);  
 Add.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int FirstNumber = Integer.*parseInt*(Firstno.getText());  
  
 int secondnumber = Integer.*parseInt*(Secongno.getText());  
 int result = FirstNumber+secondnumber;  
 output.setText(Integer.*toString*(result));  
  
  
  
  
  
 }  
 });  
  
 frame.add(Add);  
 JButton clear = new JButton("Clear");  
 clear.setBounds(100,130,80,20);  
 clear.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 Firstno.setText(" ");  
 Secongno.setText("");  
 output.setText("");  
  
 }  
 });  
  
  
 frame.add(clear);  
  
 JButton exitButton = new JButton("Exit");  
 exitButton.setBounds(210,140,80,20);  
 exitButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 System.*exit*(0);  
 }  
 });  
 frame.add(exitButton);  
 frame.setVisible(true);  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 }  
  
 public static void main(String[] args) {  
 new addCalculator();  
 }  
}

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 MultiplicationTable {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Input a number: ");

int number = scanner.nextInt();

System.out.println("Multiplication Table for " + number + ":");

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

int result = number \* i;

System.out.println(number + " x " + i + " = " + result);

}

scanner.close();

}

}

In this program, we first import the **Scanner** class to read input from the user.

Then we create a **Scanner** object named **scanner**.

We prompt the user to enter a number by displaying the message "Input a number: ".

We read the input number from the user using the **nextInt()** method of the **Scanner** object and store it in the **number** variable.

We then display the message "Multiplication Table for {number}:" to indicate the number for which we are generating the multiplication table.

Using a **for** loop, we iterate from 1 to 10, and for each iteration, we calculate the result of multiplying the **number** by the current iteration value **i**.

Finally, we print each multiplication statement using **System.out.println()**.

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

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 Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc = new Scanner(System.in);

int num = sc.nextInt();

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

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

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

}

System.out.println();

}

for(int i = num-1 ; i>0;i--) {

for(int j=2\*i-1; j>=1;j--) {

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

}

System.out.println();

}

}

}

.

13.

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

import java.util.Scanner;

public class MatrixTranspose {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of rows in the matrix: ");

int rows = scanner.nextInt();

System.out.print("Enter the number of columns in the matrix: ");

int columns = scanner.nextInt();

int[][] matrix = new int[rows][columns];

System.out.println("Enter the elements of the matrix:");

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

for (int j = 0; j < columns; j++) {

matrix[i][j] = scanner.nextInt();

}

}

int[][] transpose = findTranspose(matrix);

System.out.println("Transpose of the matrix:");

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

for (int j = 0; j < rows; j++) {

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

}

System.out.println();

}

scanner.close();

}

public static int[][] findTranspose(int[][] matrix) {

int rows = matrix.length;

int columns = matrix[0].length;

int[][] transpose = new int[columns][rows];

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

for (int j = 0; j < columns; j++) {

transpose[j][i] = matrix[i][j];

}

}

return transpose;

}

}

In this program, we first prompt the user to enter the number of rows and columns in the matrix using the **Scanner** class.

We then create a 2D array **matrix** with the specified number of rows and columns.

Next, we ask the user to enter the elements of the matrix using nested **for** loops. The elements are stored in the **matrix** array.

We then call the **findTranspose()** method and pass the **matrix** array as an argument. This method calculates the transpose of the matrix and returns it as a new 2D array.

Inside the **findTranspose()** method, we determine the number of rows and columns in the original matrix.

We create a new 2D array **transpose** with the number of rows equal to the number of columns in the original matrix, and the number of columns equal to the number of rows in the original matrix.

We use nested loops to iterate through the elements of the original matrix and assign them to the corresponding positions in the **transpose** array.

Finally, we print the transpose of the matrix by iterating through the **transpose** array and displaying the elements.

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

Code- 01

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class WordCount {

public static void main(String[] args) {

// Provide the path to the text file

String filePath = "path/to/your/text/file.txt";

try {

int wordCount = countWords(filePath);

System.out.println("Number of words in the file: " + wordCount);

} catch (FileNotFoundException e) {

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

e.printStackTrace();

}

}

public static int countWords(String filePath) throws FileNotFoundException {

File file = new File(filePath);

Scanner scanner = new Scanner(file);

int wordCount = 0;

while (scanner.hasNext()) {

// Read the next word and increment the word count

scanner.next();

wordCount++;

}

scanner.close();

return wordCount;

}

}

Code-2

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("\\Users\\sanika\\Documents\\OOP\_lab\_que.docx");

        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. 

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

import java.util.ArrayList;

import java.util.Iterator;

public class ArrayListIteration {

public static void main(String[] args) {

// Create an ArrayList

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

numbers.add(20);

numbers.add(30);

numbers.add(40);

// Iterate using for-loop

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

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

System.out.println(numbers.get(i));

}

// Iterate using iterator

System.out.println("\nIterator Loop:");

Iterator<Integer> iterator = numbers.iterator();

while (iterator.hasNext()) {

System.out.println(iterator.next());

}

// Iterate using advanced for-loop

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

for (int number : numbers) {

System.out.println(number);

}

}

}

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 q16 {

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 nd dispaly it .

import java.util.Scanner;

public class q16 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s;

System.out.println("Enter 10 Strings");

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

s = sc.nextLine();

System.out.println(s);

}

System.out.println("Enter two strings to combine");

String a1 = sc.nextLine();

String a2 = sc.nextLine();

System.out.println(a1+" " +a2);

String rev="";

char ch;

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

ch= a1.charAt(i);

rev = ch+rev;

}

System.out.println(rev);

}

}

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

CODE-1

class Person{  
 String name;  
 int age;  
 Person(String name, int age){  
 this.name= name;  
 this.age=age;  
 }  
  
}  
  
class Employee extends Person{  
 String Designation;  
 double Salary;  
 Employee(String name, int age, String Designation, double Salary){  
 super(name,age);  
 this.Designation=Designation;  
 this.Salary= Salary;  
  
   
  
  
if(Salary>5000){  
 System.*out*.println(name);  
 }  
}  
}  
class q18{  
 public static void main(String[] args) {  
  
 Employee emp1 = new Employee("Sarvesh",20,"Manager",5000);  
 Employee emp2 = new Employee("Bhushan",21,"Asst.Manager",4000);  
 Employee emp3 = new Employee("Omkar",22,"Co.Manager",6000);  
 Employee emp4 = new Employee("Mayuresh",20,"Asst.Manager",7000);  
 Employee emp5 = new Employee("Harshdeep",21,"Co.Manager",3000);  
  
 }  
}

import java.util.Scanner;

class Person {

protected String name;

protected int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

}

class Employee extends Person {

private String designation;

private double salary;

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

super(name, age);

this.designation = designation;

this.salary = salary;

}

public boolean isSalaryGreaterThan(double amount) {

return salary > amount;

}

}

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Employee[] employees = new Employee[5];

// Accept data for 5 persons

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

System.out.println("Enter details for Person " + (i + 1));

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

String name = scanner.nextLine();

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

int age = Integer.parseInt(scanner.nextLine());

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

String designation = scanner.nextLine();

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

double salary = Double.parseDouble(scanner.nextLine());

employees[i] = new Employee(name, age, designation, salary);

System.out.println();

}

// Display names of employees with salary greater than 5000

System.out.println("Employees with salary greater than 5000:");

for (Employee employee : employees) {

if (employee.isSalaryGreaterThan(5000)) {

System.out.println(employee.name);

}

}

}

}

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.

interface Account {

void set(String name, int accountNumber, double balance);

void display();

}

interface Person {

void store(String name);

void disp();

}

class Customer implements Account, Person {

private String name;

private int accountNumber;

private double balance;

@Override

public void set(String name, int accountNumber, double balance) {

this.name = name;

this.accountNumber = accountNumber;

this.balance = balance;

}

@Override

public void display() {

System.out.println("Account Information");

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

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

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

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

}

@Override

public void store(String name) {

this.name = name;

}

@Override

public void disp() {

System.out.println("Customer Information");

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

}

private double calculateInterest() {

return balance \* 0.05; // Assuming interest rate of 5%

}

}

public class Main {

public static void main(String[] args) {

Customer customer = new Customer();

// Accept customer information

customer.store("John Doe");

customer.set("John Doe", 123456, 5000.0);

// Display account and customer information

customer.display();

customer.disp();

}

}

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.  
"



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;

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 void addCourse(String course) {

courses.add(course);

}

public void removeCourse(String course) {

courses.remove(course);

}

public void displayCourses() {

System.out.println("Courses for " + name + ":");

for (String course : courses) {

System.out.println(course);

}

}

}

public class Main {

public static void main(String[] args) {

// Create a student

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

// Add courses

student.addCourse("Math");

student.addCourse("Science");

student.addCourse("English");

// Display courses

student.displayCourses();

// Remove a course

student.removeCourse("Science");

// Display courses after removal

student.displayCourses();

}

}

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.

// Person class

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 class extending Person

class Employee extends Person {

private String employeeId;

private String jobTitle;

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

super(firstName, lastName);

this.employeeId = employeeId;

this.jobTitle = jobTitle;

}

public String getEmployeeId() {

return employeeId;

}

@Override

public String getLastName() {

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

}

}

// Main class

public class Main {

public static void main(String[] args) {

// Create a Person object

Person person = new Person("John", "Doe");

System.out.println("First Name: " + person.getFirstName());

System.out.println("Last Name: " + person.getLastName());

System.out.println();

// Create an Employee object

Employee employee = new Employee("Jane", "Smith", "E12345", "Manager");

System.out.println("First Name: " + employee.getFirstName());

System.out.println("Last Name: " + employee.getLastName());

System.out.println("Employee ID: " + employee.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.util.HashSet;

public class LongestConsecutiveSequence {

public static int findLongestConsecutiveSequence(int[] nums) {

// Create a HashSet to store all the numbers in the array

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

for (int num : nums) {

set.add(num);

}

int longestSequence = 0;

// Iterate through the array

for (int num : nums) {

// Check if the current number is the start of a sequence

if (!set.contains(num - 1)) {

int currentNum = num;

int currentSequence = 1;

// Find the length of the consecutive sequence

while (set.contains(currentNum + 1)) {

currentNum++;

currentSequence++;

}

// Update the longest sequence if necessary

longestSequence = Math.max(longestSequence, currentSequence);

}

}

return longestSequence;

}

public static void main(String[] args) {

int[] nums = {49, 1, 3, 200, 2, 4, 70, 5};

int longestSequence = findLongestConsecutiveSequence(nums);

System.out.println("The length of the longest consecutive sequence is: " + longestSequence);

}

}

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.

// Custom exception class for age not within range

class AgeNotWithinRangeException extends Exception {

    public AgeNotWithinRangeException(String message) {

        super(message);

    }

}

// Custom exception class for invalid name

class NameNotValidException extends Exception {

    public NameNotValidException(String message) {

        super(message);

    }

}

// Student class

class Student {

    private int rollNo;

    private String name;

    private int age;

    private String course;

    // Parameterized constructor

    public Student(int rollNo, String name, int age, String course)

            throws AgeNotWithinRangeException, NameNotValidException {

        // Validate age

        if (age < 15 || age > 21) {

            throw new AgeNotWithinRangeException("Age is not within the range of 15 to 21.");

        }

        // Validate name

        if (!name.matches("^[a-zA-Z ]+$")) {

            throw new NameNotValidException("Name contains numbers or special symbols.");

        }

        this.rollNo = rollNo;

        this.name = name;

        this.age = age;

        this.course = course;

    }

    // Getters

    public int getRollNo() {

        return rollNo;

    }

    public String getName() {

        return name;

    }

    public int getAge() {

        return age;

    }

    public String getCourse() {

        return course;

    }

}

public class StudentException {

    public static void main(String[] args) {

        try {

            // Create a valid student

            Student validStudent = new Student(1, "John Doe", 20, "Computer Science");

            System.out.println("Valid Student:");

            System.out.println("Roll No: " + validStudent.getRollNo());

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

            System.out.println("Age: " + validStudent.getAge());

            System.out.println("Course: " + validStudent.getCourse());

            // Create an invalid student with age not within range

            Student invalidAgeStudent = new Student(2, "Jane Smith", 14, "Mathematics");

            System.out.println("Invalid Age Student:");

            // Create an invalid student with name not valid

            Student invalidNameStudent = new Student(3, "1234", 18, "Physics");

            System.out.println("Invalid Name Student:");

        } catch (AgeNotWithinRangeException e) {

            System.out.println("Error creating student: " + e.getMessage());

        } catch (NameNotValidException e) {

            System.out.println("Error creating student: " + e.getMessage());

        }

    }

}

25. 