**1.**

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

public class RemoveDuplicateInArrayExample{

public static int removeDuplicateElements(int arr[], int n){

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

return n;

}

int[] temp = new int[n];

int j = 0;

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

if (arr[i] != arr[i+1]){

temp[j++] = arr[i];

}

}

temp[j++] = arr[n-1];

// Changing original array

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

arr[i] = temp[i];

}

return j;

}

public static void main (String[] args) {

int arr[] = {10,20,20,30,30,40,50,50};

int length = arr.length;

length = removeDuplicateElements(arr, length);

//printing array elements

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

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

}

}

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

import java.io.\*;

import java.util.\*;

class GFG {

private static void FindCommonElemet(String[] arr1,

String[] arr2)

{

Set<String> set = new HashSet<>();

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

for (int j = 0; j < arr2.length; j++) {

if (arr1[i] == arr2[j]) {

// add common elements

set.add(arr1[i]);

break;

}

}

}

for (String i : set) {

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

}

}

// main method

public static void main(String[] args)

{

// create Array 1

String[] arr1

= { "Article", "in", "Geeks", "for", "Geeks" };

// create Array 2

String[] arr2 = { "Geeks", "for", "Geeks" };

// print Array 1

System.out.println("Array 1: "

+ Arrays.toString(arr1));

// print Array 2

System.out.println("Array 2: "

+ Arrays.toString(arr2));

System.out.print("Common Elements: ");

// Find the common elements

FindCommonElemet(arr1, arr2);

}

}

**2.**

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

package String;

public class CountWords

{

public static void main(String[] args)

{

String input="Welcome to Java Session Session Session"; //Input String

String[] words=input.split(" "); //Split the word from String

int wrc=1; //Variable for getting Repeated word count

for(int i=0;i<words.length;i++) //Outer loop for Comparison

{

for(int j=i+1;j<words.length;j++) //Inner loop for Comparison

{

if(words[i].equals(words[j])) //Checking for both strings are equal

{

wrc=wrc+1; //if equal increment the count

words[j]="0"; //Replace repeated words by zero

}

}

if(words[i]!="0")

System.out.println(words[i]+"--"+wrc); //Printing the word along with count

wrc=1;

}

}

}

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

class Check\_Letter

{

public static void main(String[] args)

{

String str = "String Exercises";

boolean pre = false;

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

{

if(str.charAt(i) == 'x')

{

pre=true;

break;

}

}

System.out.println(pre);

}

}

**3.**

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

public class StringFormatter {

public static String reverseString(String str){

char ch[]=str.toCharArray();

String rev="";

for(int i=ch.length-1;i>=0;i--){

rev+=ch[i];

}

return rev;

}

}

File: TestStringFormatter.java

public class TestStringFormatter {

public static void main(String[] args) {

System.out.println(StringFormatter.reverseString("my name is khan"));

System.out.println(StringFormatter.reverseString("I am sonoo jaiswal"));

}

}

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

import java.util.\*;

class PalindromeExample2

{

public static void main(String args[])

{

String original, reverse = ""; // Objects of String class

Scanner in = new Scanner(System.in);

System.out.println("Enter a string/number to check if it is a palindrome");

original = in.nextLine();

int length = original.length();

for ( int i = length - 1; i >= 0; i-- )

reverse = reverse + original.charAt(i);

if (original.equals(reverse))

System.out.println("Entered string/number is a palindrome.");

else

System.out.println("Entered string/number isn't 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.**

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.**

abstract class Marks {

public abstract float getPercentage();

}

class A extends Marks {

private float marks1, marks2, marks3;

public A(float m1, float m2, float m3) {

marks1 = m1;

marks2 = m2;

marks3 = m3;

}

public float getPercentage() {

return ((marks1 + marks2 + marks3) / 300) \* 100;

}

}

class B extends Marks {

private float marks1, marks2, marks3, marks4;

public B(float m1, float m2, float m3, float m4) {

marks1 = m1;

marks2 = m2;

marks3 = m3;

marks4 = m4;

}

public float getPercentage() {

return ((marks1 + marks2 + marks3 + marks4) / 400) \* 100;

}

}

public class Main {

public static void main(String[] args) {

A studentA = new A(80, 85, 90);

B studentB = new B(75, 80, 85, 90);

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**

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");

}

}

**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 {

/\*

\* Create the method long power(int, int) here.

\*/

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 should not be negative.");

else

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

}

}

class Solution {

public static final MyCalculator my\_calculator = new MyCalculator();

public static final Scanner in = new Scanner(System.in);

public static void main(String[] args) {

while (in .hasNextInt()) {

int n = in .nextInt();

int p = in .nextInt();

try {

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

} catch (Exception e) {

System.out.println(e);

}

}

}

}

**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 \*/

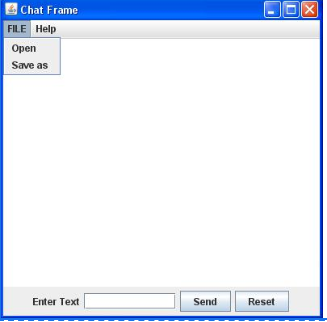
System.out.println("Not found");

}

}

}

}

9. 

import javax.swing.\*;

import java.awt.\*;

class gui {

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();

}

}

26. Implement Student class using following Concepts

1. All types of Constructors
2. Static variables and instance variables
3. Static blocks and instance blocks
4. Static methods and instance methods

**CODE –**

1) All types of Constructors

//Java Program to demonstrate the use of the parameterized constructor.  
class Student4{  
 int id;  
 String name;  
 //creating a parameterized constructor  
 Student4(int i,String n){  
 id = i;  
 name = n;  
 }  
 //method to display the values6  
 void display()  
 {  
 System.*out*.println(id+" "+name);  
 }  
  
 public static void main(String args[]){  
 //creating objects and passing values  
 Student4 s1 = new Student4(111,"Ram");  
 Student4 s2 = new Student4(222,"Raj");  
 //calling method to display the values of object  
 s1.display();  
 s2.display();  
 }  
}

**CODE-**

class Student6{  
 int id;  
 String name;  
 //constructor to initialize integer and string  
 Student6(int i,String n){  
 id = i;  
 name = n;  
 }  
 //constructor to initialize another object  
 Student6(Student6 s){  
 id = s.id;  
 name =s.name;  
 }  
 void display(){System.*out*.println(id+" "+name);}  
  
 public static void main(String args[]){  
 Student6 s1 = new Student6(111,"Raj");  
 Student6 s2 = new Student6(s1);  
 s1.display();  
 s2.display();  
 }  
}

**CODE-**

class Car {  
 Car() {  
 System.*out*.println("Car is created");  
 }  
 Car(int a) {  
 System.*out*.print("Car with parameter is created");  
 }  
 public static void main(String args[]){  
  
 Car b = new Car();  
  
 Car a = new Car(10);  
 }  
}

**CODE-** 2) Static variables and instance variables

class Student{  
 int rollno;//instance variable  
 String name;  
 static String *college*;//static variable  
 //constructor  
 Student(int a, String n){  
 rollno = a;  
 name = n;  
 }  
 //method to display the values  
 void display (){  
 System.*out*.println(rollno+" "+name+" "+*college*);}  
}  
//Test class to show the values of objects  
class TestStaticVariable1{  
 public static void main(String args[]){  
  
 Student s1 = new Student(111,"Rohan");  
 Student s2 = new Student(222,"Arya");  
 //we can change the college of all objects by the single line of code  
 Student.*college*="VIT";  
 s1.display();  
 s2.display();  
 }  
}

**CODE-** 3) Static blocks and instance blocks

class Test {  
  
 // Case 1: Static variable  
 static int *i*;  
  
 // Case 2: non-static variables  
 // Case 3: Static block  
 // Start of static block  
 static  
 {  
 *i* = 5;  
 System.*out*.println("static block called ");  
 }  
 // End of static block  
}  
class ABC {  
 public static void main(String args[])  
 {  
 System.*out*.println(Test.*i*);  
 }  
}

**CODE-** 4) Static methods and instance methods

public class Methods {  
  
 static int *a* = 8;  
  
  
 int b = 2;  
  
 void simpleDisplay()  
 {  
  
 System.*out*.println(*a*);  
 System.*out*.println(b);  
 }  
  
  
 static void staticDisplay()  
 {  
 System.*out*.println(*a*);  
 }  
  
 public static void main(String[] args)  
 {  
 Methods obj = new Methods();  
 obj.simpleDisplay();  
  
 *staticDisplay*();  
 }  
}

27. There is a class Adder which has two data members of type 1D int array and int variable. It has two functions: getdata and numsum. Function getdata accepts non-empty array of distinct integers from user in 1D int array data member and a targetsum in another data member. The function numsum adds any two elements from an input array which is equal to targetsum and return an array of resulting two elements, in any order. If no two numbers sum up to the target sum, the function should return an empty array. Note that the target sum is to be obtained by summing two different integers in the array; you can’t add a single integer to itself in order to obtain the target sum. You can assume that there will be at most one pair of numbers summing up to the target sum. Use constructor. Use extra variables if needed.

**Sample Input and Output**

**Test Case 1**

|  |  |  |
| --- | --- | --- |
| **Input Parameters** | **Values** | **Expected Output** |
| 1D Array | [3,5,-4,8,11,1,-1,7] | [8,7] |
| targetsum | 15 |

**Test Case 2**

|  |  |  |
| --- | --- | --- |
| **Input Parameters** | **Values** | **Expected Output** |
| 1D Array | [3,5,-4,8,11,1,-1,6] | [ ] |
| targetsum | 15 |

**CODE-**

package com.company;

import java.util.Arrays;

import java.util.Scanner;

public class Adder{

static Scanner sc = new Scanner(System.in);

public int [] arr = new int[n];

public static int n;

public static int targetsum;

Adder(int n)

{

this.n = n;

}

public void getdata(int n){

System.out.println("Enter Your "+ n +" inputs:");

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

{

arr[i] = sc.nextInt();

}

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

targetsum = sc.nextInt();

}

public int[] numsum() {

int[] arr2 = new int[2];

int flag = 0;

for (int j = 0; j < n && flag == 0; j++) {

for (int k = 0; k < n; k++) {

if (arr[j] + arr[k] == targetsum && arr[j] != arr[k]) {

arr2[0] = arr[j];

arr2[1] = arr[k];

flag = 1;

}

}

}

return arr2;

}

public static void main(String[] args) {

System.out.println("Enter the size of array:");

Adder.n = sc.nextInt();

Adder a1 = new Adder(Adder.n);

a1.getdata(Adder.n);

System.out.print("Resultant array is : " + Arrays.toString(a1.numsum()));

}

}

28. Calculate area of triangle, square & circle using function overloading. Function parameter accept from user. Create Base Class **Shape** and Derived Classes **Triangle, Square, Circle** respectively. Implement **getInputs()** Method for accepting inputs, and Overload **setArea()** method for calculating area of respective shapes.

Use Class **Tester** for creating objects.

**Sample Input and Output**

|  |  |  |
| --- | --- | --- |
| **Sample Input/Parameter for Triangle** | **Values** | **Expected Output** |
| Height (H) | 50 | 2500 |
| Base (B) | 100 |

|  |  |  |
| --- | --- | --- |
| **Sample Input/Parameter for Circle** | **Values** | **Expected Output** |
| π (Pie) | 3.14 | 7853.98 |
| Radius (R) | 50 |

|  |  |  |
| --- | --- | --- |
| **Sample Input/Parameter for Square** | **Values** | **Expected Output** |
| Side (S) | 15 | 225 |

**CODE-**

|  |
| --- |
| **public class Tester {**  **public static void main(String[] args) {**  **triangle t = new triangle();**  **square s = new square();**  **circle c = new circle();**  **shape shape = new shape();**  **shape.getInput();**  **System.out.println("Area of Square is: "+ s.setArea(shape.side));**  **System.out.println("Area of Triangle: " + t.setArea(shape.h, shape.b));**  **System.out.println("Area of circle: "+ c.setArea(shape.radius));**  **}**  **}**  **-----------------------------------------------------------------------------------------------------------------------**  **import java.util.Scanner;**  **public class shape {**  **float A, side;**  **double area;**  **double area2, radius;**  **double PI = 3.142;**  **float h,b;**  **public void setArea(){**  **}**  **public void getInput(){**  **Scanner sc = new Scanner(System.in);**  **System.out.println("Enter length for square:");**  **side = sc.nextFloat();**  **System.out.println("Enter length for height and base:");**  **h = sc.nextFloat();**  **b = sc.nextFloat();**  **System.out.println("Enter radius:");**  **radius = sc.nextFloat();**  **}**  **---------------------------------------------------------------------------------------------------------------------**  **public class square extends shape {**  **public double setArea(float a){**  **A = a\*a;**  **return A;**  **}**  **}**  **public class triangle extends shape{**  **public double setArea(float h, float b){**  **area = 0.5\*h\*b;**  **return area;**  **}**  **}**  **public class circle extends shape{**  **public double setArea(double r){**  **area2 = PI \*r\*r;**  **return area2;**  **}**  **}** |

**29.** Write a program for following exception, develop a suitable scenario in which the following exceptions occur:

* 1. divide by zero
  2. Array index out of bounds exception
  3. Null pointer Exception

**Develop a menu driven program for handle the above listed exception.**

**Sample Input and Output**

|  |  |
| --- | --- |
| **Sample Input/Parameter** | **Expected Output** |
| Try to divide a number by zero | You shouldn’t divide a number by zero. |
| Try to access the array index which does not exist. | OOPs!!!Array Index 7 out of bounds for length 6. |
| Try to find the length of String in method (pass parameter string as null) | Null Pointer Exception arises!! |

**CODE-**

|  |
| --- |
| **import java.util.Scanner;**  **public class Main {**  **static int num = 1;**  **static Scanner sc = new Scanner(System.in);**  **public static void main(String[] args) {**  **while (num != 0){**  **System.out.println("\n Enter \n" +**  **"1 for Divide by zero exception\n" +**  **"2 for Array index out of bounds exception\n" +**  **"3 for Null pointer Exception\n" +**  **"0 to exit\n");**  **num = sc.nextInt();**  **switch (num){**  **case 1:**  **int num1 , num2 , r;**  **System.out.println("Enter first number: ");**  **num1 = sc.nextInt();**  **System.out.println("Enter second number: ");**  **num2 = sc.nextInt();**  **try {**  **r = num1 / num2;**  **System.out.println("The result is: " + r);**  **} catch (ArithmeticException exception) {**  **System.out.println("You should not divide a number by zero.\n" + exception +"\n");**  **}**  **break;**  **case 2:**  **int arr[] = { 1, 2, 3, 4, 5 , 6};**  **int p;**  **try {**  **System.out.println("enter position :");**  **p = sc.nextInt();**  **System.out.println(arr[p]);**  **}catch (ArrayIndexOutOfBoundsException exception){**  **System.out.println("OOPs!!!Array Index 7 out of bounds for length 6\n" + exception +"\n");**  **}**  **break;**  **case 3:**  **String str1 = null;**  **try{**  **if(str1.equals("java")){**  **System.out.println("same");**  **}**  **}catch (NullPointerException exception){**  **System.out.println("Null Pointer Exception arises\n" + exception +"\n");**  **}**  **break;**  **default:**  **if (num == 0){**  **}else {**  **System.out.println("\nEnter appropriate number\n");**  **}**  **break;**  **}**  **}**  **System.out.println("\nThank you");**  **}**  **}** |

**30.** A fashion E-commerce company keeps a track of all the orders using an **ArrayList** and a class Order. Implement class **Order** and retrieve and return the list of items present in all the orders. Implement the logic inside **getItems()** method.

Test the functionalities using the **main()** method of the **Tester** class.

**Sample Input and Output**

|  |  |
| --- | --- |
| **Sample Input** | **Expected Output** |
| orders=[Order(101,itemNames=[Jeans, Shirt, Belt],true), Order(102,itemNames=[Tie,Shirt],true),Order(103,itemNames=[Tshirt,Socks,Tie],true) | [Jeans,Shirt,Belt,Tie,Shirt,Tshirt,Socks,Tie] |
| orders=[Order(311,itemNames=[Sportswear, Dumbbell],true), Order(102,itemNames=[, Jeans],true),Order(103,itemNames=[Smartwatch,Fitnessband,Joggers],true) | Sportswear,Dumbbell,Smartwatch,Fitnessband,Joggers,Jeans] |

**CODE-**

|  |
| --- |
| **import java.util.List;**  **import java.util.ArrayList;**  **public class Order {**  **private int orderId;**  **private List<String> itemNames;**  **private boolean cashOnDelivery;**  **public Order(int orderId, List<String> itemNames, boolean cashOnDelivery) {**  **this.orderId = orderId;**  **this.itemNames = itemNames;**  **this.cashOnDelivery = cashOnDelivery;**  **}**  **public int getOrderId() {**  **return orderId;**  **}**  **public void setOrderId(int orderId) {**  **this.orderId = orderId;**  **}**  **public List<String> getItemNames() {**  **return itemNames;**  **}**  **public void setItemNames(List<String> itemNames) {**  **this.itemNames = itemNames;**  **}**  **public boolean isCashOnDelivery() {**  **return cashOnDelivery;**  **}**  **public void setCashOnDelivery(boolean cashOnDelivery) {**  **this.cashOnDelivery = cashOnDelivery;**  **}**  **@Override**  **public String toString() {**  **return "Order Id: "+getOrderId()+", Item names: "+getItemNames()+", Cash on delivery: "+isCashOnDelivery();**  **}**  **}**  **class Tester\_Order {**  **public static List<String> getItems(List<Order> orders) {**  **List<String> ls = new ArrayList<>();**  **for (Order o : orders) {**  **ls.addAll(o.getItemNames());**  **}**  **return ls;**  **}**  **public static void main(String[] args) {**  **List<Order> orders = new ArrayList<Order>();**  **List<String> items1 = new ArrayList<String>();**  **items1.add("Jeans");**  **items1.add("Shirt");**  **items1.add("Belt");**  **orders.add(new Order(101, items1, true));**  **List<String> items2 = new ArrayList<String>();**  **items2.add("Tie");**  **items2.add("Shirt");**  **orders.add(new Order(102, items2, true));**  **List<String> items3 = new ArrayList<String>();**  **items3.add("Tshirt");**  **items3.add("Socks");**  **items3.add("Tie");**  **orders.add(new Order(103, items3, true));**  **List<String> items = getItems(orders);**  **System.out.println("----------------------------");**  **System.out.println("list 1: " + items);**  **List<Order> orders\_2 = new ArrayList<Order>();**  **List<String> items\_4 = new ArrayList<String>();**  **items\_4.add("Sportswear");**  **items\_4.add("Dumbbell");**  **orders\_2.add(new Order(311, items\_4, true));**  **List<String> items\_5 = new ArrayList<String>();**  **items\_5.add("");**  **items\_5.add("Jeans");**  **orders\_2.add(new Order(312, items\_5, true));**  **List<String> items\_6 = new ArrayList<String>();**  **items\_6.add("Smartwatch");**  **items\_6.add("Fitnessband");**  **items\_6.add("Joggers");**  **orders\_2.add(new Order(313, items\_6, true));**  **System.out.println("----------------------------");**  **List<String> items\_7 = getItems(orders\_2);**  **System.out.println("");**  **System.out.println("list 2: " + items\_7);**  **}**  **}** |
|  |

**31.** Implement various operations using JDBC Connectivity. Consider any suitable application and implement JDBC code for it.

**CODE-**

|  |
| --- |
| **import java.sql.\*;**  **import java.util.\*;**  **import java.sql.Statement;**  **public class jdbc {**  **static Scanner sc = new Scanner(System.in);**  **public static void main(String[] args) {**  **int num = 1;**  **String url="jdbc:mysql://localhost:3306/student\_DB";**  **String user="root";**  **String pwd="New@pwd@123";**  **while (num != 0) {**  **System.out.println("\n Enter \n" +**  **"1 Insert Entry\n" +**  **"2 Update Entry\n"+**  **"3 Delete Entry\n"+**  **"4 Show all Entry\n"+**  **"5 Exit\n");**  **System.out.println("Please Enter Below:");**  **num = sc.nextInt();**  **switch (num) {**  **case 1:**  **System.out.println("Enter Name of the student:");**  **String sname = sc.next();**    **System.out.println("Enter roll no of the student:");**  **int sroll = sc.nextInt();**    **System.out.println("Enter Email of the Student:");**  **String semail = sc.next();**    **try {**  **Connection con = DriverManager.getConnection(url, user, pwd);**  **Statement statement = con.createStatement();**  **// Insert**  **String sqlInsert = "insert into vit\_students values('" + sname + "','" + sroll + "','"+semail+"')";**  **statement.executeUpdate(sqlInsert);**  **System.out.println("record inserted successfully");**  **} catch (Exception ex) {**  **System.out.println(ex);**  **}**  **break;**  **case 2:**  **try{**  **Connection con = DriverManager.getConnection(url, user, pwd);**  **Statement statement = con.createStatement();**  **System.out.println("Enter roll no:");**  **int roll = sc.nextInt();**  **System.out.println("Enter name to edit:");**  **String name = sc.next();**  **// Uppdate**  **String sqlUpdate = "update vit\_students set student\_name=('"+ name +"') where student\_roll\_no=('"+ roll +"')";**  **statement.executeUpdate(sqlUpdate);**  **System.out.println("Updated Successfully");**  **} catch (Exception ex){**  **System.out.println(ex);**  **}**  **break;**  **case 3:**  **try {**  **System.out.println("Enter roll no to delete the Entry:");**  **int roll = sc.nextInt();**  **Connection con = DriverManager.getConnection(url, user, pwd);**  **Statement statement = con.createStatement();**  **// Delete**  **String sqlDelete = "delete from vit\_students where student\_roll\_no =('"+ roll +"')";**  **statement.executeUpdate(sqlDelete);**  **System.out.println("Record Deleted Successfully");**  **}catch (Exception e){**  **System.out.println(e);**  **}**  **break;**  **case 4:**  **try{**  **Connection con = DriverManager.getConnection(url, user, pwd);**  **Statement statement = con.createStatement();**  **String sql = "select \* from vit\_students";**  **ResultSet rs = statement.executeQuery(sql);**  **while (rs.next()) {**  **System.out.print("\nstudent\_name: " + rs.getString(1));**  **System.out.print(" student\_roll\_no: " + rs.getInt(2));**  **System.out.print(" student\_email: " + rs.getString(3)+"\n");}**  **}catch(Exception e)**  **{**  **System.out.println(e);**  **}**  **break;**  **case 5:**  **num = 0;**  **break;**  **default:**  **if (num == 0){**  **}else {**  **System.out.println("\nEnter appropriate number\n");**  **}**  **break;**  **}**  **}**  **System.out.println("\nThank you");**  **}**  **}** |
|  |