**A picture containing text, clipart

Description automatically generated**

**TERM WORK**

**ON**

**JAVA PROGRAMING**

**LANGUAGE**

**(PCS 308)**

**2021-22**

**SUBMITTED TO: SUBMITTED BY:**

**DR. PRATEEK SRIVASTAVA DEVANSH RAUTELA**

**ASSOCIATE PROFESSOR UNIVERSITY ROLL NO. : 2018314**

**GEHU, D.DUN SECTION : C**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GRAPHIC ERA HILL UNIVERSITY, DEHRADUN**

**A picture containing text, clipart

Description automatically generated**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Program No.** | **Program Name** | **Page No.** |
| 1 | Create a class “Student” having following instance variables and methods.  **Instance variables:** ID, Name, Branch and university  **Method:** setDetails() and showDetails(). |  |
| 2 | Write a Java Program to demonstrate the working of a banking-system  **Instance variables:** name, account\_no, amount  **Instance methods:**  deposit(), withdraw(), checkBalance(), insert() and display(). |  |
| 3 | Write a program to sum two numbers. Here inputs are provided through command line argument. |  |
| 4 | Create class Employee with following attributes and methods  ID, name, department and salary.  The setDetails() method sets the values of ID, name, department and salary.  And showDetails() method shows the value of each field. |  |
| 5 | Re-write program 1 with better memory management approach.  **Note:** use of static keyword |  |
| 6 | Apply functions on the String "Java". |  |
| 7 | Apply functions on StringBuffer object "HELLO" |  |
| 8 | Create a class “Student” having following instance variables and methods.  Instance variables: ID, Name, Branch, city and university  While creating constructors with one, two, three, four and five arguments reuse the constructors by **construction chaining** |  |
| 9 | Create two dimensional integer array and insert, search and traverse this array.  **Note:** Use Scanner class to insert data |  |
| 10 | Create a jagged array having three rows. Where 1st row contains 3 columns, 2nd row contains 4 columns and 3rd row contains 2 columns. Insert and traverse it. |  |
| 11 | Create a class “Shape” having area() method to calculate area. Overload the area() method for shapes like triangle, rectangle and circle. |  |
| 12 | Create a class “Bank” having method getRateOfInterest(). Create child classes as HDFC, SBI and PNB and override getRateOfInterest() and return interest rates as 4.0, 4.5 and 5% correspondingly. |  |
| 13 | Create a package **pack1** having one class **C1** and one interface **I1**. Class **C1** has two methods **int sum(int, int)** and **int sub(int, int)**. The I1 has one method **int division(int, int)**. Create another package **pack2** having class **C2**. Reuse **C1** and **I1** in **C2** and show the results. |  |
| 14 | Write a program to divide two numbers with proper exception handlers. |  |
| 15 | Create LowBalanceException that occurs when user tries to withdraw some amount that is greater than his current bank balance. To withdraw you have to write a **void withdrawal(int amount)** method |  |
| 16 | Write a program that reads from a text file byte by byte and writes in some another file. Write this program in an efficient way. |  |
| 17 | Write a program that reads from a text file char by char and writes in some another file. Write this program in an efficient way. |  |
| 18 | Write a program that reads from a text file line by line and writes on console. |  |
| 19 | Write a program that take your name from keyboard and writes in some text file. |  |
| 20 | Write a multithreaded program where three threads are there and printing the numbers from 1 to 10 concurrently. |  |
| 21 | Write a program to set and get the name of threads also set and get the priority of threads. |  |
| 22 | Write a class Display having void wish(String name) methods that wishes hello to given string name. Between printing hello and provided string name apply delay of 500 milliseconds. |  |
| 23 | Write a class Display having **synchronized void wish(String)** methods that wishes hello to given string name. Between printing hello and provided string name apply delay of 500 milliseconds. |  |
| 24 | Write a class Customer having **balance** as a property and **void withdrawal(int amount)**, and **void deposit(int amount)** as instance methods. There are two threads, the first thread wants to withdrawal some amount and second thread wants to deposit some amount. |  |
| 25 | Create a GUI for student’s information system. A GUI that asks all the relevant information’s related to a student. |  |
| 26 | Create a canvas having smiley face |  |
| 27 | . Write a JFrame having three textfields. The first two textfields refers to two numbers on which sum or subtraction will happen. The third textfield will show the result. There are two buttons “SUM” and “SUBTRACTION”. |  |
| 28 | Write a Java program that interacts with database. It enables to-  (a) Inserts the student name and roll number to database.  (b) Fetch records from table  (c) Modify the records  (d) Delete the records |  |

**A picture containing text, clipart

Description automatically generated**

**DEPARTMENT OF CSE**

|  |
| --- |
| **Photograph**  **Passport Size** |

**STUDENT LAB REPORT SHEET**

**Name of Student .................................................... Mob. No ......................................**

**Address Permanent .....................................................................................................**

**Father’s Name ........................... Occupation ...................... Mob. No .........................**

**Mother’s Name ........................... Occupation ...................... Mob. No .......................**

**Section ............ Branch ............ Semester ............ Class Roll No ............ Grade A B C**

**Local Address ................................... Email ............................................ Marks 5 3 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.N**  **o.** | **Practical** | **D.O.P.** | **Date of Submiss**  **ion** | **Grade (Viva)** | **Grade**  **(Report**  **File)** | **Total**  **Marks**  **(out of**  **10)** | **Student’s Signature** | **Teacher’s Signatur**  **e** |
| **1** |  |  |  |  |  |  |  |  |
| **2** |  |  |  |  |  |  |  |  |
| **3** |  |  |  |  |  |  |  |  |
| **4** |  |  |  |  |  |  |  |  |
| **5** |  |  |  |  |  |  |  |  |
| **6** |  |  |  |  |  |  |  |  |
| **7** |  |  |  |  |  |  |  |  |
| **8** |  |  |  |  |  |  |  |  |
| **9** |  |  |  |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |
| **11** |  |  |  |  |  |  |  |  |
| **12** |  |  |  |  |  |  |  |  |
| **13** |  |  |  |  |  |  |  |  |
| **14** |  |  |  |  |  |  |  |  |
| **15** |  |  |  |  |  |  |  |  |
| **16** |  |  |  |  |  |  |  |  |
| **17** |  |  |  |  |  |  |  |  |
| **18** |  |  |  |  |  |  |  |  |
| **19** |  |  |  |  |  |  |  |  |
| **20** |  |  |  |  |  |  |  |  |
| **21** |  |  |  |  |  |  |  |  |
| **22** |  |  |  |  |  |  |  |  |
| **23** |  |  |  |  |  |  |  |  |
| **24** |  |  |  |  |  |  |  |  |
| **25** |  |  |  |  |  |  |  |  |
| **26** |  |  |  |  |  |  |  |  |
| **27** |  |  |  |  |  |  |  |  |
| **28** |  |  |  |  |  |  |  |  |

**PRACTICAL 1**

**Create a class “Student” having following instance variables and methods.**

**Instance variables : ID, Name, Branch and University.**

**Method: setDetails() and showDetails().**

**The setDetails() method sets the values of ID, Name, Branch and University.**

**And showDetails() method shows the values of each field.**

**CODE :**

**class Student{**

**int ID;**

**String name;**

**String Branch;**

**String University;**

**void setDetails(int r, String n, String br, String un)**

**{**

**name = n;**

**ID = r;**

**Branch = br;**

**University = un;**

**}**

**void showDetails() {**

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

**System.out.println("ID : "+ID);**

**System.out.println("Branch : "+Branch);**

**System.out.println("University : "+University);**

**}**

**public static void main(String[] args)**

**{**

**Student Details = new Student();**

**int r=Integer.parseInt(args[0]);**

**String n=args[1];**

**String br=args[2];**

**String un=args[3];**

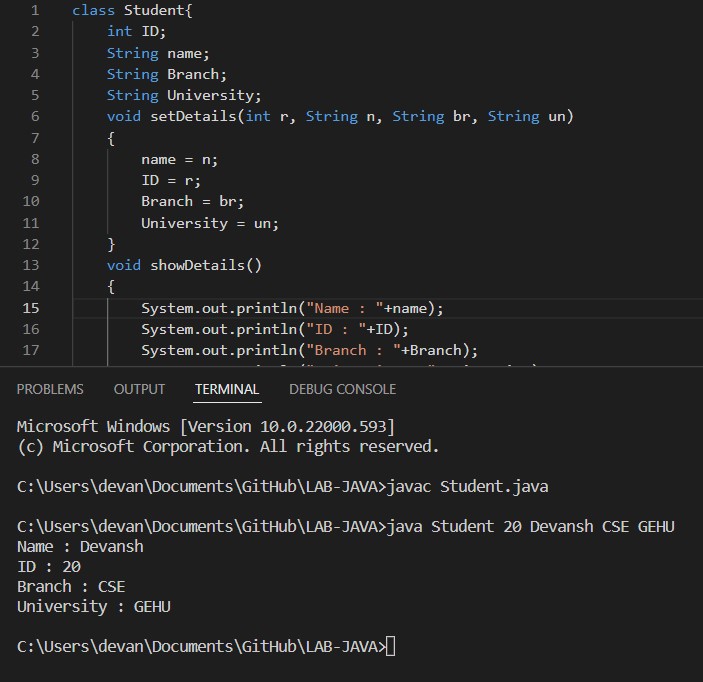
**Details.setDetails(r,n,br,un);**

**Details.showDetails();**

**}**

**}**

**OUTPUT :**

****

**PRACTICAL 2**

**Write a JAVA Program to demonstrate the working of a banking – system.**

**Instance variables : name, account\_no, amount.**

**Instance methods : deposit(), withdraw(), checkBalance(), insert(), and display().**

**Here we can deposit and withdraw amount from our account using deposit() and withdraw() methods respectively.**

**The insert() methods to initialize state and display() method is to display state values.**

**CODE :**

**public class Bank {**

**int account\_no;**

**String name;**

**int amount;**

**void insert(int account\_no, String name, int amount)**

**{**

**this.account\_no = account\_no;**

**this.name = name;**

**this.amount = amount;**

**}**

**void deposit(int amount)**

**{**

**this.amount += amount;**

**}**

**void withdraw(int amount)**

**{**

**if(this.amount >= amount)**

**{**

**System.out.println("Money withdrawn : "+amount);**

**this.amount -= amount;**

**System.out.println("Amount remaining :"+this.amount);**

**}**

**else{**

**System.out.println("Low Balance");**

**}**

**}**

**void display()**

**{**

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

**System.out.println("Account Number : "+this.account\_no);**

**System.out.println("Account Balance : "+this.amount);**

**}**

**public static void main(String args[])**

**{**

**Bank obj = new Bank();**

**obj.insert(20000,"Devansh Rautela",100000);**

**obj.display();**

**obj.withdraw(200000);**

**obj.display();**

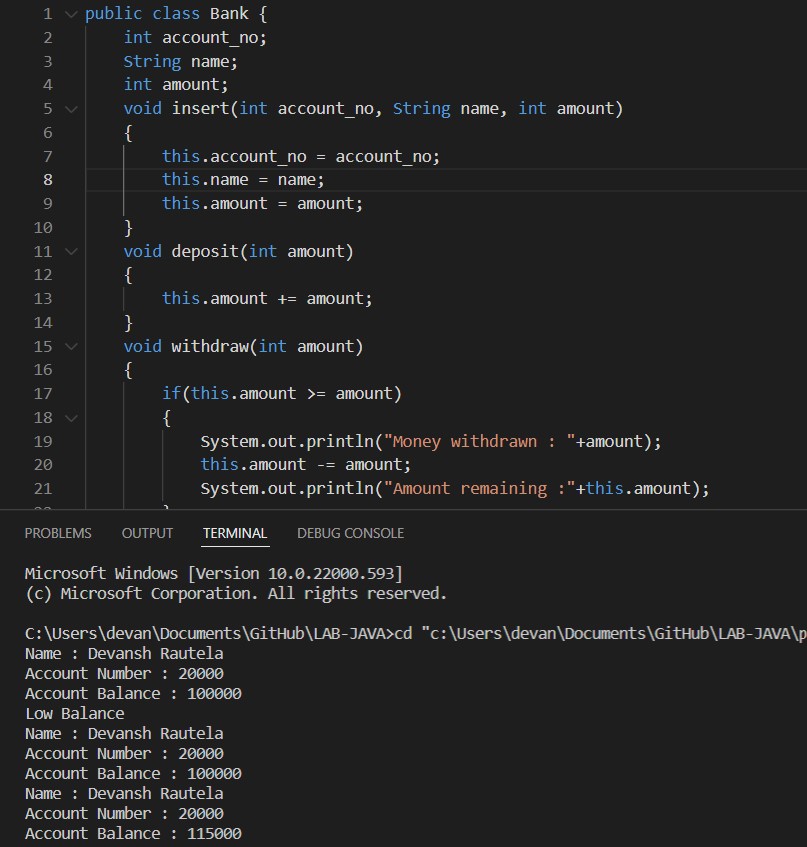
**obj.deposit(15000);**

**obj.display();**

**}**

**}**

**OUTPUT :**

****

**PRACTICAL 3**

**Write a Program to sum two numbers. Here inputs are provided through command line arguments.**

**CODE :**

**public class sum**

**{**

**public static void main(String[] args)**

**{**

**int n1 = Integer.parseInt(args[0]);**

**int n2 = Integer.parseInt(args[1]);**

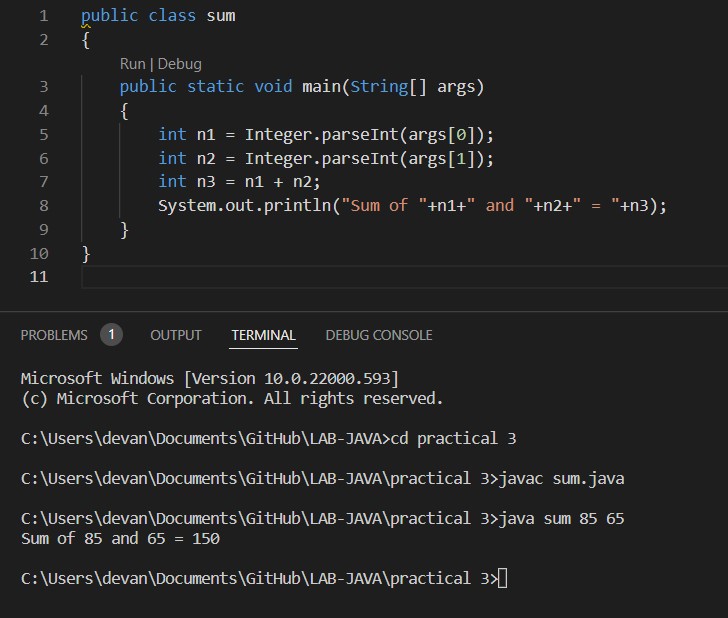
**int n3 = n1 + n2;**

**System.out.println("Sum of "+n1+" and "+n2+" = "+n3);**

**}**

**}**

**OUTPUT :**

****

**PRACTICAL 4**

**Create class Employee with following attributes and methods ID, name, department and salary.**

**This setDetails() method sets the values of ID, name, department, and salary.**

**And showDetails() method shows the value of each field.**

**Note :**

1. **Values must be entered through Scanner class.**
2. **Use proper constructor.**
3. **Use “this” reference variables to avoid ambiguity.**

**CODE :**

**public class Employee {**

**int ID;**

**String name;**

**String department;**

**int salary;**

**Employee(int ID, String name, String department, int salary)**

**{**

**this.ID = ID;**

**this.name = name;**

**this.department = department;**

**this.salary = salary;**

**}**

**void setDetails(int r, String na, String br, int un)**

**{**

**name = na;**

**ID = i;**

**department = dp;**

**salary = s;**

**}**

**void display()**

**{**

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

**System.out.println("Roll no. : "+ID);**

**System.out.println("Department : "+department);**

**System.out.println("Salary : "+salary);**

**}**

**public static void main(String args[])**

**{**

**Employee xyz = new Employee(0, "", "", 0);**

**int i = Integer.parseInt(args[0]);**

**String na = args[1];**

**String dp = args[2];**

**int s = Integer.parseInt(args[3]);**

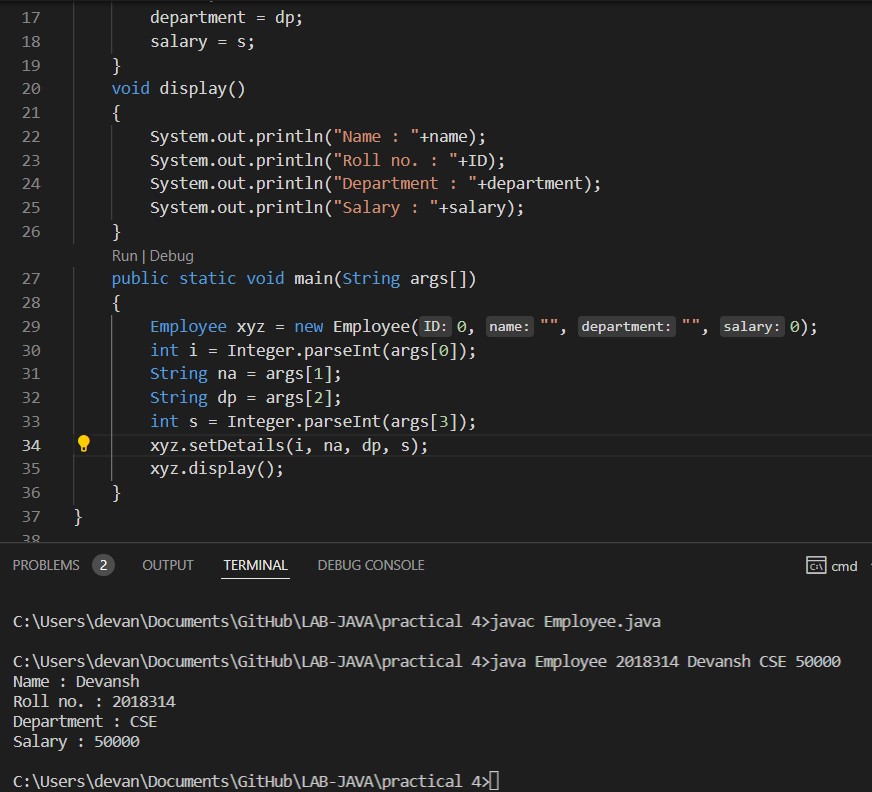
**xyz.setDetails(i, na, dp, s);**

**xyz.display();**

**}**

**}**

**OUTPUT :**

****

**PRACTICAL 5**

**Re-write program 1 with better memory management approach.**

**Note: use of static keyword**

**CODE :**

**public class memory {**

**int roll;**

**String name;**

**String branch;**

**static String univ="gehu";**

**void set(int r, String n, String b)**

**{**

**name = n;**

**roll = r;**

**branch = b;**

**}**

**void display()**

**{**

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

**System.out.println("Roll no. : "+roll);**

**System.out.println("Branch : "+branch);**

**System.out.println("University : "+univ);**

**}**

**public static void main(String args[])**

**{**

**memory detail = new memory();**

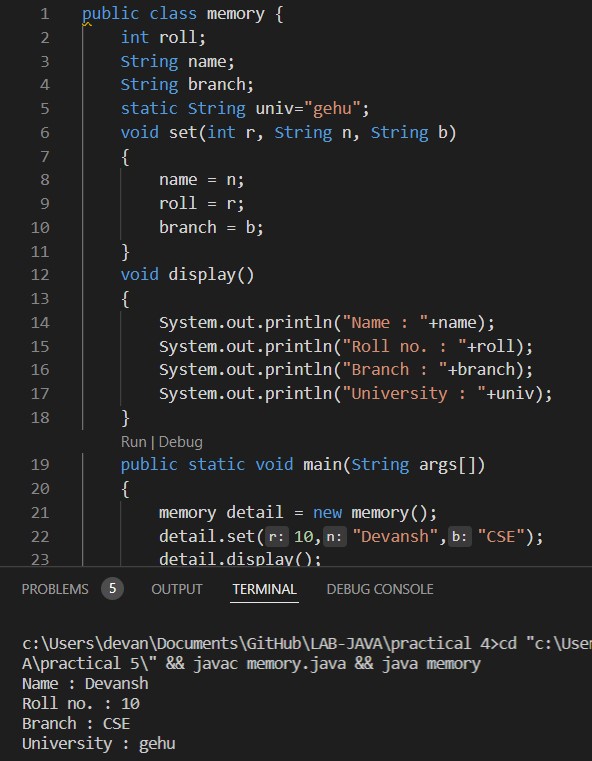
**detail.set(10,"Devansh","CSE");**

**detail.display();**

**}**

**}**

**OUTPUT:**

****

**PRACTICAL 6**

Apply following functions on the String "Java".

(i) Try to concat "Welcome" and write down your observation.

(ii) Find character at index 1

(iii) Find index of first 'a'.

(iv) Find index of second 'a'

(v) Compare "Java" to "JAVA"

(vi) Compare "Java" to "JAVA" ignoring the case

(vii) Find the index of first 'a' from last

**CODE :**

public class handlers {

public static void main(String [] args)

{

String s = "java";

System.out.println(s.concat(" WELCOME")); //doesn't changes original string

System.out.println(s.charAt(1));

System.out.println(s.indexOf('a'));

System.out.println(s.indexOf('a',s.indexOf('a')+1));

System.out.println(s.equals("JAVA"));

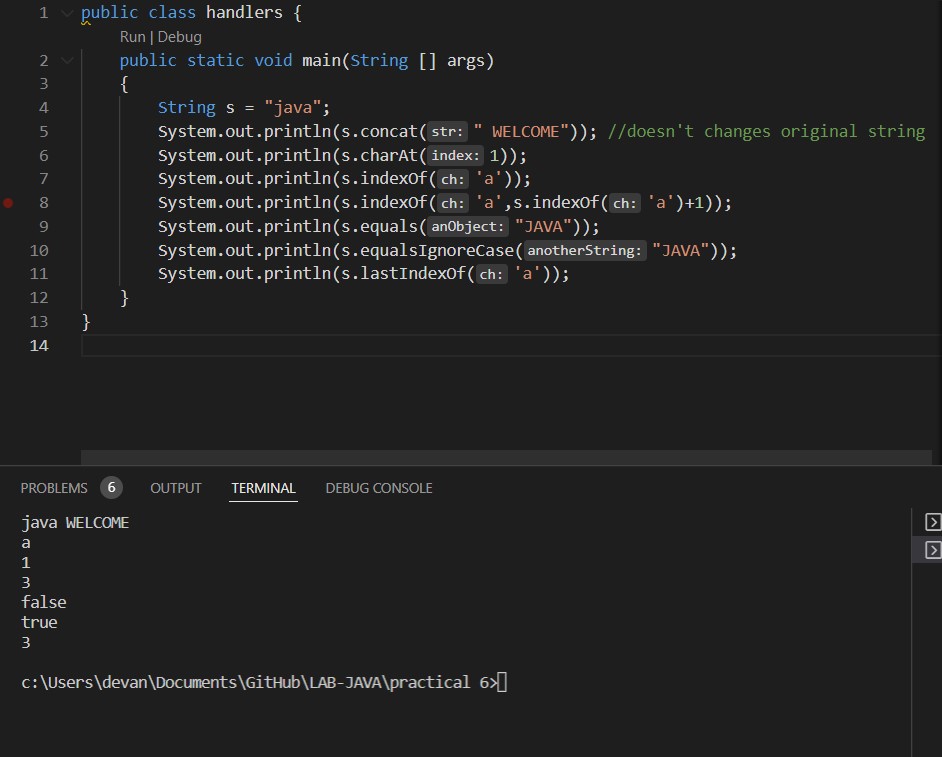
System.out.println(s.equalsIgnoreCase("JAVA"));

System.out.println(s.lastIndexOf('a'));

}

}

**OUTPUT :**

****

**PRACTICAL 7**

Apply following functions on StringBuffer object "HELLO"

(i) Append "Java"

(ii) Insert "Java" at index 1

(iii) Replace with "Java" with characters between index 1 to 2

(iv) Delete characters between index 1 and 2

(v) Reverse the string "HELLO"

**CODE :**

public class handlers {

public static void main(String[] args)

{

StringBuffer s = new StringBuffer("HELLO");

System.out.println(s.append("java"));

System.out.println(s.insert(1,"java"));

System.out.println(s.replace(1, 2, "java"));

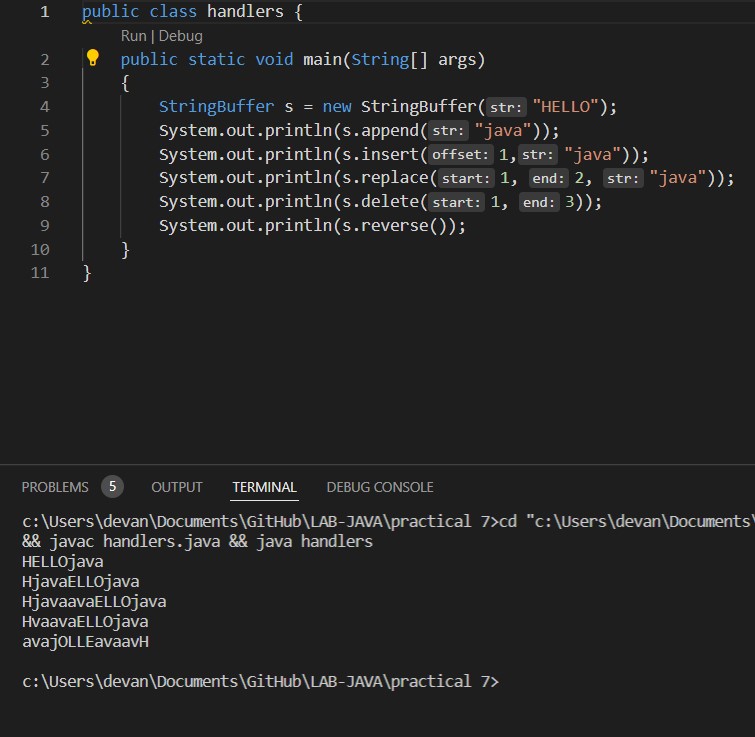
System.out.println(s.delete(1, 3));

System.out.println(s.reverse());

}

}

**OUTPUT :**

****

**PRACTICAL 8**

Create a class “Student” having following instance variables and methods.

Instance variables: ID, Name, Branch, city and university

While creating constructors with one, two, three, four and five arguments reuse the constructors by **construction chaining**

**CODE :**

public class Student {

int roll;

String name;

String branch;

String city;

String univ;

Student(int roll)

{

this.roll = roll;

}

Student(int roll, String name)

{

this(roll);

this.roll = roll;

this.name = name;

}

Student(int roll, String name, String branch)

{

this(roll,name);

this.roll = roll;

this.name = name;

this.branch = branch;

}

Student(int roll, String name, String branch, String city)

{

this(roll, name, branch);

this.roll = roll;

this.name = name;

this.branch = branch;

this.city = city;

}

Student(int roll, String name, String branch, String city,String univ)

{

this(roll, name, branch, city);

this.roll = roll;

this.name = name;

this.branch = branch;

this.city = city;

this.univ = univ;

}

void display()

{

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

System.out.println("Rollno. : "+roll);

System.out.println("Branch : "+branch);

System.out.println("City : "+city);

System.out.println("University : "+univ);

}

public static void main(String[] args)

{

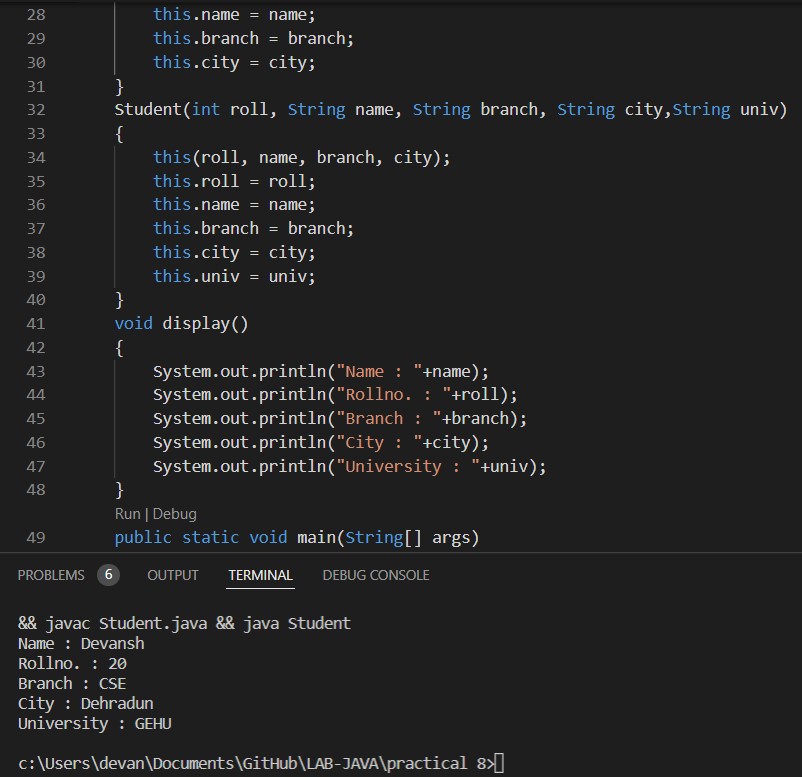
Student detail = new Student(20, "Devansh", "CSE","Dehradun","GEHU");

detail.display();

}

}

**OUTPUT :**

****

**PRACTICAL 9**

Create two dimensional integer array and insert, search and traverse this array.

**Note:** Use Scanner class to insert data.

**CODE :**

import java.io.InputStream;

import java.util.\*;

public class array {

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

int arr[][] = new int[3][3]; //input

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

{

for(int j=0; j<arr[i].length; j++)

{

arr[i][j]=sc.nextInt();

}

}

//display

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

{

for(int j=0; j<arr[i].length; j++)

{

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

}

System.out.println("");

}

//search

int key = sc.nextInt();

int f=0;

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

{

for(int j=0; j<arr[i].length; j++)

{

if(arr[i][j]==key)

{

System.out.println(key+" is present");

f=1;

break;

}

}

}

if(f==0)

{

System.out.println(key+" not found");

}

}

}

**PARCTICAL 10**

**Create a jagged array having three rows. Where 1st row contains 3 columns, 2nd row contains 4 columns and 3rd row contains 2 columns. Insert and traverse it.**

**CODE :**

**import java.io.InputStream;**

**import java.util.\*;**

**public class jagged {**

**public static void main(String[] args)**

**{**

**int arr1[][] = new int[3][];**

**arr1[0]=new int[3];**

**arr1[1]=new int[4];**

**arr1[2]=new int[2];**

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

**for(int i=0; i<arr1.length; i++) //input**

**{**

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

**{**

**arr1[i][j]=sc.nextInt();**

**}**

**}**

**for(int i=0; i<arr1.length; i++) //display**

**{**

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

**{**

**System.out.print(arr1[i][j]+" ");**

**}**

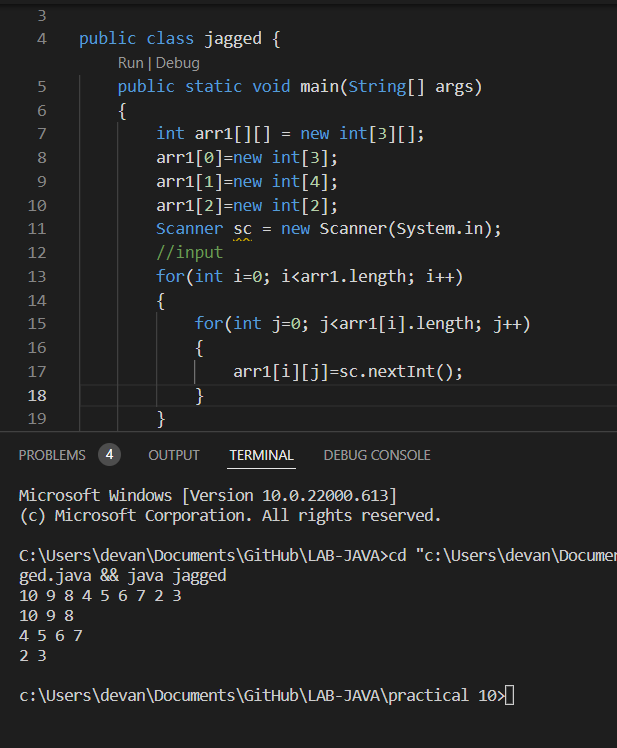
**System.out.println("");**

**}**

**}**

**}**

**OUTPUT :**

****

**PRACTICAL 11**

**Create a class “Shape” having area() method to calculate area. Overload the area() method for shapes like triangle, rectangle and circle.**

**CODE :**

**public class Shape {**

**int area(int a)**

**{**

**return a\*a;**

**}**

**int area(int i, int b)**

**{**

**return i\*b;**

**}**

**int area(float I, float b)**

**{**

**return (int)((I\*b)/2);**

**}**

**int area(int r, double p)**

**{**

**return(int)(p\*r\*r);**

**}**

**public static void main(String[] args)**

**{**

**Shape d = new Shape();**

**System.out.println("Area of square : "+d.area(10));**

**System.out.println("Area of rectangle : "+d.area(10,5));**

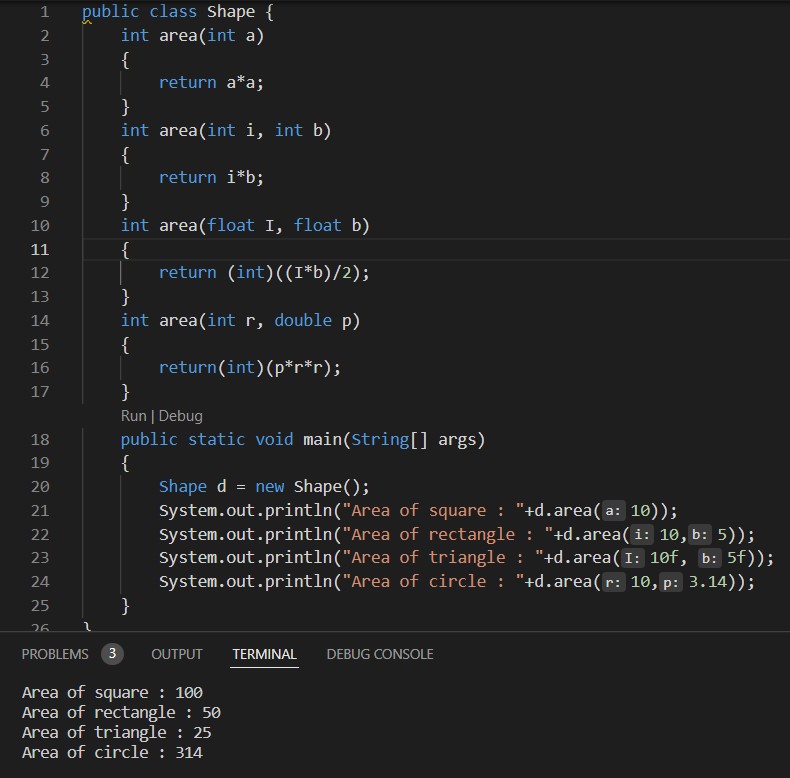
**System.out.println("Area of triangle : "+d.area(10f, 5f));**

**System.out.println("Area of circle : "+d.area(10,3.14));**

**}**

**}**

**OUTPUT :**

****

**PRACTICAL 12**

**Create a class “Bank” having method getRateOfInterest(). Create child classes as HDFC, SBI and PNB and override getRateOfInterest() and return interest rates as 4.0, 4.5 and 5% correspondingly.**

**Use concept of Upcasting to implement this scenario.**

**CODE :**

**class Bank{**

**void getRoi()**

**{**

**System.out.println("5.5%");**

**}**

**}**

**class hdfc extends bank**

**{**

**void getRoi()**

**{**

**System.out.println("hdfc 4%");**

**}**

**}**

**class pnb extends bank**

**{**

**void getRoi()**

**{**

**System.out.println("pnb 4.5%");**

**}**

**}**

**class sbi extends bank**

**{**

**void getRoi()**

**{**

**System.out.println("sbi 5%");**

**}**

**}**

**class call**

**{**

**public static void main(String[] args) {**

**bank r;**

**b=new bank();**

**b.getRoi();**

**b=new pnb();**

**b.getRoi();**

**b=new hdfc();**

**b.getRoi();**

**b=new sbi();**

**b.getRoi();**

**}**

**}**

**OUTPUT :**

**Text

Description automatically generated**

**PRACTICAL 13**

**QUESTION:** Create a package **pack1** having one class **C1** and one interface **I1**. Class **C1** has two methods **int sum(int, int)** and **int sub(int, int)**. The I1 has one method **int division(int, int)**. Create another package **pack2** having class **C2**. Reuse **C1** and **I1** in **C2** and show the results.

**SOURCE CODE:**

***C1.java -->***

package pack1;

public class c1{

public int sum(int a,int b){

return a+b;

}

public int sub(int a,int b){

return a-b;

}

}

***I1.java 🡪***

package pack1;

public interface i1 {

public int divide(int a,int b);

}

***C2.java 🡪***

package pack2;

import pack1.\*;

public class c2 extends c1 implements i1{

public int divide(int a,int b){

return a/b;

}

}

class run{

public static void main(String[] args) {

c2 c=new c2();

System.out.println("Sum is "+c.sum(12,1));

System.out.println("Sum is "+c.sub(12,1));

System.out.println("Sum is "+c.divide(12,2));

}

}

**OUTPUT :**

Text

Description automatically generated

**PRACTICAL 14**

Write a program to divide two numbers with proper exception handlers.

**CODE:**

import java.util.Scanner;

class run{

public static void main(String[] args) {

Scanner s=new Scanner(System.in);

System.out.println("\nEnter two number to divide ");

int a=s.nextInt();

int b=s.nextInt();

int res;

try {

res=a/b;

System.out.println("The division of two number are "+res);

} catch (Exception e) {

System.out.println(e);

}

System.out.println("End of code ");

}

}

**OUTPUT:**

Text

Description automatically generated

Text

Description automatically generated

**PRACTICAL 15**

Create LowBalanceException that occurs when user tries to withdraw some amount that is greater than his current bank balance. To withdraw you have to write a **void withdrawal(int amount)** method.

**CODE:**

import java.util.\*;

class Bank{

long acc\_no,salary;

String name;

void insert(long acc\_no,long salary,String name){

this.name=name;

this.salary=salary;

this.acc\_no=acc\_no;

}

void deposite(long salary){

this.salary+=salary;

}

void withdraw(long salary){

try{

if(this.salary>salary)

this.salary-=salary;

else

throw new Exception(" ");

}

catch(Exception e){

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

System.out.println("Low Balance");

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

}

}

void showDetails(){

System.out.println("\nName -> "+this.name);

System.out.println("Account\_Number -> "+this.acc\_no);

System.out.println("Account\_Balance -> "+this.salary);

}

public static void main(String[] args) {

Scanner s=new Scanner(System.in);

System.out.println("\nEnter name,account number,amount");

String name=s.next();

long acc\_no=s.nextLong();

long salary=s.nextLong();

Bank b=new Bank();

b.insert(acc\_no, salary, name);

b.deposite(1000);

b.showDetails();

b.withdraw(1500);

b.showDetails();

b.withdraw(1000);

}

}

**OUTPUT:**Text

Description automatically generated

**PRACTICAL 16**

Write a program that reads from a text file byte by byte and writes in some another file. Write this program in an efficient way.

**CODE:**

import java.io.\*;

class file\_io{

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

FileInputStream fis=new FileInputStream("C:\\Users\\HP\\OneDrive\\Desktop\\LAB FILE\\Java\_lab\\file\\end\_term\\Q16\\hello.txt");

BufferedInputStream bis=new BufferedInputStream(fis);

FileOutputStream fos=new FileOutputStream("C:\\Users\\HP\\OneDrive\\Desktop\\LAB FILE\\Java\_lab\\file\\end\_term\\Q16\\hello1.txt");

BufferedOutputStream bos=new BufferedOutputStream(fos);

int x=0;

while((x=bis.read())!=-1){

bos.write(x);

}

bos.close();

fos.close();

bis.close();

fis.close();

}

}

**OUTPUT:**

A screenshot of a car dashboard

Description automatically generated with medium confidence

**PRACTICAL 17**

Write a program that reads from a text file char by char and writes in some another file. Write this program in an efficient way.

**SOURCE CODE:**

import java.io.\*;

class file\_io{

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

FileReader fis=new FileReader("C:\\Users\\HP\\OneDrive\\Desktop\\LAB FILE\\Java\_lab\\file\\end\_term\\Q17\\hello.txt");

BufferedReader bis=new BufferedReader(fis);

FileWriter fos=new FileWriter("C:\\Users\\HP\\OneDrive\\Desktop\\LAB FILE\\Java\_lab\\file\\end\_term\\Q17\\hello1.txt");

BufferedWriter bos=new BufferedWriter(fos);

int x=0;

while((x=bis.read())!=-1){

bos.write(x);

}

bos.close();

fos.close();

bis.close();

fis.close();

}

}

**OUTPUT:**

Text

Description automatically generated

**PRACTICAL 18**

Write a program that reads from a text file line by line and writes on console.

**CODE:**

import java.io.\*;

public class line\_console{

public static void main(String args[]) {

try{

FileReader fr=new FileReader("C:\\Users\\HP\\OneDrive\\Desktop\\LAB FILE\\Java\_lab\\file\\end\_term\\Q18\\file.txt");

BufferedReader br=new BufferedReader(fr);

StringBuffer sb=new StringBuffer();

String line;

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

sb.append(line);

sb.append("\n");

}

fr.close();

System.out.println("Contents of File: ");

System.out.println(sb.toString());

}

catch(IOException e){

e.printStackTrace();

}

}

}

**OUTPUT:**Text

Description automatically generated

**PRACTICAL 19**

Write a program that take your name from keyboard and writes in some text file.

**CODE:**

import java.io.\*;

import java.util.Scanner;

class con\_file{

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

File file=new File("C:\\Users\\HP\\OneDrive\\Desktop\\LAB FILE\\Java\_lab\\file\\end\_term\\Q19\\file.txt");

FileOutputStream fos=new FileOutputStream(file);

BufferedOutputStream bos=new BufferedOutputStream(fos);

Scanner s=new Scanner(System.in);

System.out.println("Enter name ");

String st=s.next();

byte[]b=st.getBytes();

bos.write(b);

bos.close();

fos.close();

s.close();

}

}

**OUTPUT:**

Graphical user interface, text

Description automatically generated

**PRACTICAL 20**

Write a multithreaded program where three threads are there and printing the numbers from 1 to 10 concurrently.

**CODE:**

import java.lang.Thread;

class A extends Thread{

public void run(){

int i;

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

System.out.println("From A :"+i);

}

}

class B extends Thread{

public void run(){

int j;

for(j=1;j<=10;j++)

System.out.println("From B :"+j);

}

}

class C extends Thread{

public void run(){

int k;

for(k=1;k<=10;k++)

System.out.println("From C :"+k);

}

}

class abcd{

public static void main(String[] args) {

A a=new A();

B b=new B();

C c=new C();

a.start();

b.start();

c.start();

}

}

**OUTPUT:**

Text

Description automatically generated

**PRACTICAL 21**

Write a program to set and get the name of threads also set and get the priority of threads.

**CODE:**

class A implements Runnable{

public void run(){

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

System.out.println("A :"+i);

}

}

class B implements Runnable{

public void run(){

for(int j=11;j<=21;j++)

System.out.println("B :"+j);

}

}

class Abc{

public static void main(String[] args) {

A a=new A();

B b=new B();

Thread t=new Thread(a);

t.start();

try {

t.join();

} catch (Exception e) {

System.out.println(e);

}

Thread t1=new Thread(b);

t1.setPriority(9);

t1.start();

try {

t1.join();

} catch (Exception e) {

System.out.println(e);

}

System.out.println(t.getName());

t.setName("MY Thread");

System.out.println(t.getName());

System.out.println(t.getPriority());

System.out.println(t1.getPriority());

}

}

**OUTPUT:**

Text

Description automatically generated

**PRACTICAL 22**

Write a class Display having void wish(String name) methods that wishes hello to given string name. Between printing hello and provided string name apply delay of 500 milliseconds. Suppose multiple threads are there and they are trying to access this wish() method concurrently on **same object** then irregular output will be there. Write this application in such a way so that output becomes regular.

**CODE:**

class display extends Thread

{

String name;

display(String name)

{

this.name=name;

}

public void run()

{

System.out.print("hello ");

try

{

Thread.sleep(500);

wish(name);

}

catch(Exception e)

{

System.out.println(e);

}

}

void wish(String name)

{

System.out.println(name);

}

}

class A

{

public static void main(String args[])

{

display a = new display("arun");

display b = new display("ram");

display c = new display("sayam");

a.start();

try

{

a.join();

}

catch(Exception e)

{

System.out.println(e);

}

b.start();

try

{

b.join();

}

catch(Exception e)

{

System.out.println(e);

}

c.start();

}

}

**OUTPUT:**

A screenshot of a computer

Description automatically generated

**PRACTICAL 23**

Write a class Display having **synchronized void wish(String)** methods that wishes hello to given string name. Between printing hello and provided string name apply delay of 500 milliseconds. Suppose multiple threads are there and they are trying to access this wish() method concurrently on **different** **objects** then irregular output will be there. Write this application in such a way so that output becomes regular.

**CODE:**

import java.lang.Thread;

class Display{

    public synchronized void wish(String name){

            System.out.print("Hello ");

            try {

                Thread.sleep(500);

            } catch (Exception e) {

                System.out.println(e);

            }

            System.out.println(name);

    }

}

class MyThread extends Thread{

    Display d;

    String name;

    MyThread(Display d,String name){

        this.d=d;

        this.name=name;

    }

    public void run(){

        d.wish(name);

    }

}

class Demo{

    public static void main(String[] args) {

        Display d=new Display();

        MyThread t1=new MyThread(d, "Deepanshu");

        t1.start();

        MyThread t2=new MyThread(d, "Ravi");

        t2.start();

        MyThread t3=new MyThread(d, "Ravi");

        t3.start();

    }

}

**OUTPUT:**

A screenshot of a computer

Description automatically generated

**PRACTICAL 24**

.Write a class Customer having **balance** as a property and **void withdrawal(int amount)**, and **void deposit(int amount)** as instance methods. There are two threads, the first thread wants to withdrawal some amount and second thread wants to deposit some amount. Apply inter thread communication where, if withdrawal amount is higher than current balance then first thread will wait for second thread to deposit then resume the withdrawal.

**CODE:**

class customer

{

int bal=1000;

public synchronized void withdraw(int amt)

{

System.out.println("going to withdraw");

if(this.bal<amt)

{

try

{

 wait();

}

catch(Exception e)

{

System.out.println(e);

}

}

this.bal -=amt;

System.out.println("final balance after withdrawal "+bal);

}

public synchronized void deposit(int amt)

{

System.out.println("going to deposit");

this.bal +=amt;

System.out.println("final balance after deposit "+bal);

notify();

}

}

class thread1 extends Thread

{

customer c;

thread1(customer c)

{

this.c=c;

}

public void run()

{

c.withdraw(1500);

}

}

class thread2 extends Thread

{

customer c;

thread2(customer c)

{

this.c=c;

}

public void run()

{

c.deposit(1000);

}

}

class demo

{

public static void main(String args[])

{

customer c = new customer();

thread1 t1 = new thread1(c);

t1.start();

thread2 t2 = new thread2(c);

t2.start();

}

}

**OUTPUT:**

A screenshot of a computer

Description automatically generated

**PRACTICAL 25**

Create a GUI for student’s information system. A GUI that asks all the relevant information’s related to a student.

**CODE:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

class  xyz extends JFrame implements ActionListener{

    JTextField t1,t2,t3,t4;

    xyz()

    {

        JLabel l1=new JLabel("Name");

        t1=new JTextField(20);

        JLabel l4=new JLabel("Course");

        t4=new JTextField(10);

        JLabel l2=new JLabel("Section");

        t2=new JTextField(3);

        JLabel l3=new JLabel("Id");

        t3=new JTextField(10);

        Button b=new Button("submit");

        setLayout(new FlowLayout());

        add(l1);

        add(t1);

        add(l4);

        add(t4);

        add(l2);

        add(t2);

        add(l3);

        add(t3);

        b.addActionListener(this);

        add(b);

    }

    public void actionPerformed(ActionEvent e)

    {

        String n=t1.getText();

        String b=t4.getText();

        String c=t2.getText();

        String i=t3.getText();

    System.out.println("name : "+n);

    System.out.println("course : "+b);

    System.out.println("Section : "+c);

    System.out.println("Id : "+Integer.parseInt(i));

    }

public static void main(String[] args) {

    xyz d=new xyz();

    d.setSize(300,400);

    d.setVisible(true);

}

}

**OUTPUT:**

Graphical user interface, text

Description automatically generated

**PRACTICAL 26**

Create a canvas having smiley face.

**CODE:**

import javax.swing.JFrame;

import java.awt.Color;

import java.awt.Graphics;

import javax.swing.JPanel;

class DrawSmiley extends JPanel

{

   public void paintComponent(Graphics g)

   {//from   w w w . j  a va 2s  .c om

      super.paintComponent(g);

      g.drawOval(80, 70, 150, 150);

      // Ovals for eyes

      // with black color filled

      g.setColor(Color.BLACK);

      g.fillOval(120, 120, 15, 15);

      g.fillOval(170, 120, 15, 15);

      // Arc for the smile

      g.drawArc(130, 180, 50, 20, 180, 180);

   }

   public static void main(String[] args)

   {

      DrawSmiley panel = new DrawSmiley();

      JFrame application = new JFrame();

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

      application.add(panel);

      application.setSize(230, 250);

      application.setVisible(true);

   }}

**OUTPUT:**

Graphical user interface

Description automatically generated with medium confidence

**PRACTICAL 27**

Write a JFrame having three textfields. The first two textfields refers to two numbers on which sum or subtraction will happen. The third textfield will show the result. There are two buttons “SUM” and “SUBTRACTION”. Once user will click on sum or subtraction buttons then the corresponding result will be displayed in result field

**CODE:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

class  xy extends JFrame implements ActionListener{

    JTextField t1,t2,t3;

    xy()

    {

        JLabel l1=new JLabel("number 1");

        t1=new JTextField(20);

        JLabel l2=new JLabel("number 2");

        t2=new JTextField(20);

        JLabel l3=new JLabel("result");

        t3=new JTextField(20);

        Button b1=new Button("add");

        Button b2=new Button("sub");

        setLayout(new FlowLayout());

        add(l1);

        add(t1);

        add(l2);

        add(t2);

        add(l3);

        add(t3);

        b2.addActionListener(this);

        b1.addActionListener(this);

        add(b1);

        add(b2);

    }

    public void actionPerformed(ActionEvent e)

    {

    String  s=e.getActionCommand();

    int result=0;

    if(s.equals("add"))

    {

        result=Integer.parseInt(t1.getText())+Integer.parseInt(t2.getText());

    }

    if(s.equals("sub"))

    {

        result=Integer.parseInt(t1.getText())-Integer.parseInt(t2.getText());

    }

    t3.setText(Integer.toString(result));

    }

public static void main(String[] args) {

    xy d=new xy();

    d.setSize(300,400);

    d.setVisible(true);

}

}

**OUTPUT:**A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**PRACTICAL 28**

Write a Java program that interacts with database. It enables to-

(a) Inserts the student name and roll number to database.

(b) Fetch records from table

(c) Modify the records

(d) Delete the records

**CODE:**

import java.sql.\*;

class jdbtest

{

public static void main(String args[])

{

Connection con=null;

try{

Class.forName("com.mysql.jdbc.Driver");

System.out.println("success");

}

catch(Exception e){

System.out.println(e);

}

try{

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/sys?autoReconnect=true&useSSL=false","root","gehu");

System.out.println("connection success");

}

catch(Exception e){

System.out.println(e);

}

try{

Statement st=con.createStatement();

st.executeUpdate("insert into emp1(id,name) values(18,'deepanshu')");

ResultSet rs=st.executeQuery("select \* from sys.emp1");

while(rs.next())

{

System.out.println(rs.getInt(1));

System.out.println(rs.getString(2));

}

}

catch(Exception e){

System.out.println(e);

}

}

}

**OUTPUT:**

Text

Description automatically generated