B.M.S COLLEGE OF ENGINEERING BENGALURU

Autonomous Institute, Affiliated to VTU

LAB REPORT

23CS3PCOOJ

Submitted in partial fulfilment of the requirements for Lab

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by:

NEHAL A K

1BM22CS176

Department of Computer Science and Engineering,

B.M.S College of Engineering,

Bull Temple Road, Basavanagudi, Bangalore, 560 019

2023-2024.

INDEX

SL.NO	Title	Date
1	LAB 1	12/12/2023
2	LAB 2	19/12/2023
3	LAB 3	26/12/2023
4	LAB 4	02/01/2024
5	LAB 5	09/01/2024
6	LAB 6	16/01/2024
7	LAB 7	23/01/2024
8	LAB 8	30/01/2024
9	LAB 9	06/02/2024
10	LAB 10	20/02/2024
11	Complete Scanned Observation Book	12/12/2023- 20/02/2024

Lab Programs

1. Develop a Java program that prints all real solutions to the quadratic equation ax 2 +bx+c = 0. Read in a, b,c and use the quadratic formula. If the discriminate b 2 -4ac is negative, display a message stating that there are no real solutions.

```
import java.util.Scanner;
class quad {
   public static void main(String args[]) {
      int a, b, c;
      double r1, r2, d;
      Scanner s = new Scanner(System.in);
      System.out.println("Nehal A K\n1BM22CS176");
      System.out.println("Enter the coefficients of a,b,c");
      a = s.nextInt();
      b = s.nextInt();
      c = s.nextInt();
      while (a == 0) {
         System.out.println("Not a quadratic equation");
         System.out.println("Enter a non zero value for a:");
         a = s.nextInt();
      }
      d = b * b - 4 * a * c;
      if (d == 0) {
         r1 = (-b) / (2 * a);
         System.out.println("Roots are real and equal");
         System.out.println("Root1 = Root2 = " + r1);
      } else if (d > 0) {
         r1 = ((-b) + (Math.sqrt(d))) / (double) (2 * a);
         r2 = ((-b) - (Math.sqrt(d))) / (double) (2 * a);
         System.out.println("Roots are real and distinct");
         System.out.println("Root1 = " + r1 + " Root2 = " + r2);
      } else if (d < 0) {
         System.out.println("Roots are imaginary");
         r1 = (-b) / (2 * a);
         r2 = Math.sqrt(-d) / (2 * a);
```

```
            System.out.println("Root1 = " + r1 + " + i" + r2);
            System.out.println("Root1 = " + r1 + " - i" + r2);
        }

    }
}
```

2. Develop a Java program to create a class Student with members usn, name, an array credits and an array marks. Include methods to accept and display details and a method to calculate SGPA of a student.

```
import java.util.Scanner;

class Subject {
    int subjectMarks;
    int credits;
    int grade;
}


class Student {
    String name;
    String usn;
    double SGPA;
    Subject[] subject;
    Scanner s;



    Student() {
        int i;
        subject = new Subject[9];
        for (i = 0; i < 9; i++)
```

```
            subject[i] = new Subject();
        s = new Scanner(System.in);
    }


    void getStudentDetails() {
        System.out.println("Enter student name");
        name = s.next();
        System.out.println("Enter student usn");
        usn = s.next();
    }


    void getMarks() {
        int i;
        for (i = 0; i < 9; i++) {
            System.out.println("Enter marks for subject " + (i + 1));
            subject[i].subjectMarks = s.nextInt();
            System.out.println("Enter credits for subject " + (i + 1));
            subject[i].credits = s.nextInt();

            if (subject[i].subjectMarks >= 90) {
                subject[i].grade = 10;
            } else if (subject[i].subjectMarks >= 80 && subject[i].subjectMarks
< 90) {
                subject[i].grade = 9;
```

```
        } else if (subject[i].subjectMarks >= 70 && subject[i].subjectMarks
< 80) {
            subject[i].grade = 8;
        } else if (subject[i].subjectMarks >= 60 && subject[i].subjectMarks
< 70) {
            subject[i].grade = 7;
        } else if (subject[i].subjectMarks >= 50 && subject[i].subjectMarks
< 60) {
            subject[i].grade = 6;
        } else if (subject[i].subjectMarks >= 40 && subject[i].subjectMarks
< 50) {
            subject[i].grade = 5;
        } else {
            System.out.println("Failed");
            System.exit(0);
        }
    }
  }


  void computeSGPA() {
    int totalCredits = 0;
    int creditsGained = 0;
    int i;

    for (i = 0; i < 9; i++) {
      totalCredits += subject[i].credits;
```

```
            creditsGained += subject[i].credits * subject[i].grade;
        }

        SGPA = (double) creditsGained / totalCredits;
    }


    void displayResult() {
        System.out.println("Name = " + name);
        System.out.println("Usn = " + usn);
        System.out.println("SGPA = " + SGPA);
    }
}

public class Main {
    public static void main(String args[]) {
        Student s1 = new Student();
        s1.getStudentDetails();
        s1.getMarks();
        s1.computeSGPA();
        s1.displayResult();
    }
}
```

3. Create a class Book which contains four members: name,author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString( ) method that could display the complete details of the book. Develop a Java program to create n book objects.

```
import java.util.Scanner;


class Book {
   String name;
   String author;
   int price;
   int numPages;

   Book(String name, String author, int price, int numPages) {
      this.name = name;
      this.author = author;
      this.price = price;
      this.numPages = numPages;
   }



   public String toString() {
      String bookDetails = "Book name: " + this.name + "\n" +
            "Author name: " + this.author + "\n" +
            "Price: " + this.price + "\n" +
            "Number of pages: " + this.numPages + "\n";
      return bookDetails;
```

```
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        System.out.println("Enter the number of books: ");
        int n = s.nextInt();


        Book[] books = new Book[n];


        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for Book " + (i + 1) + ":");
            System.out.print("Name: ");
            String name = s.next();
            System.out.print("Author: ");
            String author = s.next();
            System.out.print("Price: ");
            int price = s.nextInt();
            System.out.print("Number of pages: ");
            int numPages = s.nextInt();
            books[i] = new Book(name, author, price, numPages);
        }
```

```
        System.out.println("\nDetails of the books:");
        for (int i = 0; i < n; i++) {
            System.out.println("Book " + (i + 1) + ":\n" + books[i].toString());
        }
    }
}
```

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

```
import java.util.*;

abstract class AbsArea {
    int a, b;

    AbsArea(int x) {
        a = x;
    }

    AbsArea(int x, int y) {
        a = x;
        b = y;
    }

    abstract void area();
}

class rec extends AbsArea {
```

```
    rec(int a, int b) {
        super(a, b);
    }

    void area() {
        System.out.println("The area of the rectangle is: " + a * b);
    }
}

class tri extends AbsArea {
    tri(int a, int b) {
        super(a, b);
    }

    void area() {
        System.out.println("The area of the triangle is: " + (a * b) / 2);
    }
}

class cir extends AbsArea {
    cir(int a) {
        super(a);
    }

    void area() {
        System.out.println("The area of the circle is: " + 3.14 * a * a);
    }
}

class Main {
    public static void main(String args[]) {
System.out.println("This is done by Nehal AK\n1BM22CS176");
        int x, y;
        Scanner n = new Scanner(System.in);

        // Input for rectangle dimensions (x and y) with validation
        System.out.println("Give input for rectangle");
```

```
        x = n.nextInt();
        y = n.nextInt();
        if (x < 0 || y < 0) {
            System.out.println("Invalid input for rectangle. Please enter
positive values.");
            // You might want to handle this situation differently, such as
asking the user to enter values again.
            System.exit(1); // Exiting with status code 1 (indicating abnormal
exit)
        }

        AbsArea r = new rec(x, y);
        r.area();

        // Input for triangle dimensions (x and y) with validation
        System.out.println("Give input for triangle");
        x = n.nextInt();
        y = n.nextInt();
        if (x < 0 || y < 0) {
            System.out.println("Invalid input for triangle. Please enter positive
values.");
            System.exit(1);
        }

        AbsArea t = new tri(x, y);
        t.area();

        // Input for circle radius (x) with validation
        System.out.println("Give input for circle");
        x = n.nextInt();
        if (x < 0) {
            System.out.println("Invalid input for circle. Please enter a positive
value.");
            System.exit(1);
        }

        AbsArea c = new cir(x);
```

```
        c.area();
      }
   }
```

5.Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

```
import java.util.Scanner;
class Account {
   String CustomerName;
   double AccNo, Balance;

   Account(String CustomerName, double AccNo, double Balance) {
      this.CustomerName = CustomerName;
      this.AccNo = AccNo;
      this.Balance = Balance;
   }

   public void Deposit(double Amount) {
      Balance += Amount;
      System.out.println("DEPOSIT SUCCESSFUL");
      DisplayBalance();
   }
```

```
    void DisplayBalance() {
        System.out.println("BALANCE:" + Balance);
    }
}

class CurrentAccount extends Account {
    double MinBalance = 500.0;
    double Charges = 10.0;

    CurrentAccount(String CustomerName, double AccNo, double Balance) {
        super(CustomerName, AccNo, Balance);
    }

    void Withdraw(double Amount) {
        if (Balance >= Amount) {
            Balance -= Amount;
            System.out.println(Amount + " withdrawn successfully");
            DisplayBalance();
        } else {
            System.out.println("insufficient Balance");
        }
    }

    void UpdateBalance() {
        if (Balance <= MinBalance) {
            Balance -= Charges;
```

```
            System.out.println("service charge applied for maintaining low
balance");
            DisplayBalance();
        }
    }
}

class SavingsAccount extends Account {
    SavingsAccount(String CustomerName, double AccNo, double Balance) {
        super(CustomerName, AccNo, Balance);
    }

    double interest = 0.05;

    void UpdateBalance() {
        Balance = Balance + (interest * Balance);
        DisplayBalance();
    }

    void Withdraw(double Amount) {
        if (Balance >= Amount) {
            Balance -= Amount;
            DisplayBalance();
        } else {
            System.out.println("insufficient Balance");
        }
    }
```

```
}

class Bank {
   public static void main(String args[]) {
      String CustomerName;
      double AccNo, Balance;
      double amt, amt1;
      Scanner in = new Scanner(System.in);

      System.out.println("enter name:");
      CustomerName = in.next();

      System.out.println("enter AccNo:");
      AccNo = in.nextDouble();
      System.out.println("enter Balance:");
      Balance = in.nextDouble();
      System.out.println("enter amount to deposit");
      amt = in.nextDouble();
      System.out.println("enter amount to withdraw");
      amt1 = in.nextDouble();

      CurrentAccount c = new CurrentAccount(CustomerName, AccNo,
Balance);
      c.Deposit(amt);
      c.Withdraw(amt1);
      c.UpdateBalance();
```

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


        SavingsAccount s = new SavingsAccount(CustomerName, AccNo,
Balance);
        s.Deposit(amt);
        s.Withdraw(amt1);
        s.UpdateBalance();
    }
}
```

6.Create a package CIE which has two classes- Student and Internals. The class Student has members like usn, name, sem. The class Internals derived from Student has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

```
package CIE;
public class Student
{
    public String name;
    public String usn;
    public int sem;

public Student(String name,String usn,int sem)
  {
    this.name=name;
    this.usn=usn;
    this.sem=sem;
  }
}
```

```
package CIE;


public class Internals extends CIE.Student
{
     public int [] InternalMarks;


     public Internals(String name , String usn , int sem , int []InternalMarks)
{
     super(name , usn , sem);
     this.InternalMarks=InternalMarks;
}
}
```

```
package SEE;
import CIE.Student;

public class Externals extends Student
   {
        public int [] SeeMarks;

        public Externals(String name , String usn , int sem , int []SeeMarks)
        {
            super(name , usn , sem);
             this.SeeMarks=SeeMarks;
        }
   }
```

```
import CIE.Student;
import CIE.Internals;
import SEE.Externals;
import java.util.Scanner;


public class FinalMarks
{
     public static void main(String [] args)
    {
         Scanner s1=new Scanner(System.in);

        System.out.println("Enter the number of Students");
         int n=s1.nextInt();
```

```
        String []names=new String[n];
           String []usn=new String[n];
         int []sem = new int[n];
         int [][] InternalMarks = new int[n][5];
        int [][] SeeMarks = new int[n][5];

for(int i=0 ; i<n; i++)
{
       System.out.println("Enter details for Student" + (i+1) + ":");
       System.out.println("Name:");
       names[i]=s1.next();
       System.out.println("USN:");
       usn[i]=s1.next();
       System.out.println("SEM:");
       sem[i]=s1.nextInt();

       System.out.println("Enter Internal marks for 5 courses:");
for(int j=0; j<5; j++)
{
       System.out.println("Course"+(j+1)+":");
       InternalMarks[i][j]=s1.nextInt();
}
     System.out.println("Enter Internal marks for 5 courses:");
for(int j=0; j<5; j++)
{
    System.out.println("Course"+(j+1)+":");
    SeeMarks[i][j]=s1.nextInt();
}
}

int [][]FinalMarks =  new int[n][5];
for(int i=0 ; i<n ; i++)
{
          Internals I1 = new Internals(names[i] , usn[i] , sem[i] , InternalMarks[i]);
          Externals E1 = new Externals(names[i] , usn[i] , sem[i] , SeeMarks[i]);


for(int j=0; j<5 ;j++)
{
          FinalMarks[i][j] = I1.InternalMarks[i] + E1.SeeMarks[j];
 }
          System.out.println("Finals Marks for " + n+ "Students in 5 courses:");
for(i=0 ;i<n ;i++)
{
          System.out.println(names[i] +":");

for(int j=0; j<5;j++)
{
          System.out.println(FinalMarks[i][j] + ":");
```

```
}
        System.out.println();
       }
     s1.close();
}
}
}
```

7. Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son class, implement a constructor that cases both father and son’s age and throws an exception if son’s age is >=father’s age.

```
import java.util.Scanner;


class WrongAge extends Exception {
 WrongAge(String message) {
        super(message);
   }

}

class InputScanner {
   static Scanner sc = new Scanner(System.in);
}

class Father extends InputScanner {
   int fatherAge;

   Father() throws WrongAge {
      System.out.println("Enter Father's age");
      fatherAge = sc.nextInt();
      if (fatherAge < 0) {
         throw new WrongAge("Age cannot be negative");

      }
```

```
    }

    void display() {
        System.out.println("Father's age is " + fatherAge);
    }

}

class Son extends Father {
    int sonAge;

    Son() throws WrongAge {
        System.out.println("Enter son's age");
        sonAge = sc.nextInt();
        if (sonAge > fatherAge) {
            throw new WrongAge("Son's age cannot be greater than father's age");
        } else if (sonAge < 0) {
            throw new WrongAge("Age cannot be negative");
        }

    }

    void display() {
        System.out.println("Son's age is " + sonAge);
    }

}

public class ExceptionHandling {
    public static void Main(String args[]) {
        try{
            Son son= new Son();
            son.display();
        }
        catch(WrongAge e){
            System.out.println("Exception "+ e.getMessage());
        }


    }
```

}

PAGE NO.:

Quadratic

```
import java.util.Scanner;
class Quadratic
{
    int a,b,c;
    double r1,r2, d;
    void getd()
    {
        Scanner s= new Scanner (System.in);
        System.out.println("Enter the coefficients of a,b,c");
        a=s.nextInt() ;
        b = s.next Int();
        c = s.nextInt();
    }
    void compute()
    {
        while (a==0)
        {
            System.out.println("Not a quadratic equation");
            System.out.println("Enter a non zero value for a:");
            Scanner s = new Scanner(System.in);
            a=s.nextInt();
        }
        d=b*b-4*a*c;
        if(d==0)
        {
            r1=(-b)/(2*a);
        System.out.println("Roots are real and equal");
        system.out.printf(" Root1=Root2=",+r1);
    }
    else if(d>0)
    {
        r1=((-b)+(Math.sqrt(d)))/(double)(2*a);
```

```
        r2=((-b) - (Math.sqrt(d)))/(double)(2*a);
        System.out.println("Roots are real and distinct");
        System.out.println("Root1 = " + r1 + "Root2 = " + r2);
    }
    else if(d<0)
    {
        System.out.println("Roots are imaginary");
        r1 = (-b)/(2*a);
        r2 = Math.sqrt(-d)/(2*a);
        System.out.println("Root1 = " + r1 + " + i " + r2);
        System.out.println("Root1 = " + r1 + " - i " + r2);
    }
    }
}
class QuadraticMain
{
    public static void main(String args[])
    {
        Quadratic q = new Quadratic();
        q.getd();

        q.compute();
    }
}
```

Output:

```
Enter the coefficients of a, b, c
4 -> 2
Roots are real and distinct
Root 1 = 2    Root 2 = 1
```

PAGE NO.:

Enter the coefficient of a, b, c
0 2 3
Not a quad eqn
Enter a non zero value of a

Enter the coefficient of a, b, c
1 2 1
Roots are real and Equal
Root 1 = Root 2 = -1

Enter the coefficients of a, b, c
1 1 2
Roots are imaginary
Root 1 = 0.0 + i0.322875
Root 2 = 0.0 - i0.322875

Rw
13/12/23

Develop a Java program to create a class Student with members usn, name, an array credits and an import array marks. Include methods to accept and display details and a method to calculate SGPA of a student.

```
import java.util.Scanner;

class Subject{
    int subjectMarks;
    int credits;
    int grade;
}
class student{
    String name;
    String usn;
    double SGPA;
    Subject[] subject;
    Scanner s;

    Student(){
      int i;
      subject = new Subject[9];
      for(i=0; i<9; i++)
      {
          subject[i] = new Subject();
      }
      s = new Scanner(System.in);
    }
    void getStudentDetails(){
      System.out.println("Enter student name");
      name = s.next();
      system.out.println("Enter student usn");
      usn = s.next();
```

```
void getMarks(){
    int i;
    for(i=0; i<9; i++){
        System.out.println("Enter marks for subject "+(i+1));
        subject[i].subjectMarks= s.nextInt();
        System.out.println("Enter credits for subject"+(i+1));
        subject[i].credits = s.nextInt();
        if(subject[i].subjectMarks>=90){
            subject[i].grade=10;
        }
        else if(subject[i].subjectMarks>=90 && subject[i].marks<100)
            subject[i].grade=9;
        }
        else if(subject[i].subjectMarks>=80 && subject[i].marks<90)
            subject[i].grade=8;
        }
        else if(subject[i].subjectMarks>=60 && subject[i].marks<80)
        {   subject[i].grade=7;
        }
        else if(subject[i].subjectMarks>=50 && subject[i].marks<60)
        {  subject[i].grade=6;
        }
        else if(subject[i].subjectMarks>=40 && subject[i].marks<50)
        {
          subject[i].grade=5;
        }
          else{
              System.out.println("Failed");
              System.exit(0);
          }
    }
}
```

```
void computeSGPA(){
    int totalCredits=0;
    int creditsGained=0;
    int i;

    for(i=0; i<9; i++)
    {
      totalCredits += subject[i].credits;
      creditsGained += subject[i].credits * subject[i].grade;
    }
    SGPA = (double) creditsGained / totalCredits;
    }
    void displayResult(){
      System.out.println("Name = " + name);
      System.out.println("Usn=" + usn);
      System.out.println("SGPA=" + SGPA);
    }
}

public class Main{
      public static void main(String args[]){
      Student s1 = new Student();
      s1.getStudentDetails();
      s1.getMarks();
      s1.computeSGPA();
      s1.displayResult();
      }
}
```

PAGE NO :

Output:

Enter student name
Nehal
Enter usn
1BM22CS176
Enter marks for subject 1
90
Enter credits for subject 1
4
Enter marks for subject 2
89
Enter credits for subject 2
4
Enter marks for subject 3
85
Enter credits for subject 3
4
Enter marks for subject 4
92
Enter credits for subject 4
3
Enter marks for subject 5
80
Enter credit for subject 5
3
Enter marks for subject 6
95
Enter credit for subject 6
2
Enter marks for subject 7
91
Enter credits for subject 7
1

Enter ~~marks~~ marks for subject 8
85
Enter credits for subject 8
1

Student name : Nehal
student usn : 1BM22CS176
SGPA : 9.590990

19/12/23

Week 3

Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString() method that could display the complete details of the book. Develop a Java program to create n book objects.

```
import java.util.Scanner;

class Book {
    String name;
    String author;
    int price;
    int numPages;

    Book(String name, String author, int price, int numPages)
    {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }

    public String toString(){
        String bookDetails = "Book name: " + this.name + "\n" +
                             "Author name: " + this.author + "\n" +
                             "Price: " + this.price + "\n" +
                             "Number of pages: " + this.numPages + "\n";

        return bookDetails;
    }
}

public class Main {
    public static void main(String[] args){
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
```

```
Book[] books = new Book[n];

for (int i=0; i<n; i++)
{
System.out.println("Enter details for book " + (i+1) + ":");
System.out.print("Name : ");
String name = s.next();

System.out.print("Auther : ");
String auther = s.next();

System.out.print("Price : ");
int price = s.nextInt();

books[i]=new Book(name, auther, price, numPages);
}
System.out.println("\n Details of the books :");
for (int i=0; i<n; i++) {
    System.out.println("\n Details of the books :");
    for (int i=0; i<n; i++) {
        System.out.println("Book " + (i+1) + ":\n" +
            books[i].toString());
    }
}
}
```

Output:

```
Enter the number of books : 2
Enter the details for Book 1
Name : Trump
Auther : donald
Number of pages : 250
Enter details for Book 2
```

PAGE NO.:

Name : Friends
Author : Dale
Price : 400
Number of pages : 220

Details of the books :
Book 1 :
Book name : Trump
Author name : donald
Price : 800
Number of pages : 250

Book 2 :
Book name : Friends
Author name : Dale
Price : 400
Number of pages : 220

26/12/23

LAB - 4 2-1-23

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

```
abstract class AbsArea {
    int a, b;

    AbsArea(int x, int y) {
        a=x
        a=y;
    }

    abstract void area();
}
class re extends AbsArea {
    re(int a, int b) {
        super(a, b);
    }
    void area() {
        System.out.println("The area of the rectangle
                            is: " + a*b);
    }
}
class tri extends AbsArea {
    tri(int a, int b) {
        super(a, b);
    }
```

```
    void area () {
        System.out.println("The area of the triangle is :"
                                + (a*b)/2);
    }
}

class cir extends AbsArea{
    cir(int a){
        super(a);
    }
    void area(){
        System.out.println("The area of the circle is :"
                                + 3.14*a*a);
    }
}

class Main {
    public static void main(String args[]){
        AbsArea r= new ree(3,4);
        r.area();
        AbsArea t = new tri(4,5);
        t.area();
        AbsArea c = new cir(2);
        c.area();
        }
}
```

Output:

The area of the rectangle is: 12

The area of the triangle is: 10

The area of the circle is: 12.56

9/1/24

LAB - 5

Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed.

```
import java.util.Scanner;
class Account
{
String CustomerName;
double AccNo, Balance;
Account(String CustomerName, double AccNo, double Balance)
{
this.CustomerName = CustomerName;
this.AccNo = AccNo;
this.Balance = Balance)
}
public void Deposit(double Amount)
{
Balance += Amount;
System.out.println("Deposit Successful");
DisplayBalance();
}
void DisplayBalance();
{
System.out.println("Balance: " + Balance);
}
{
class CurrentAccount extends Account
```

```
{
double MinBalance = 500.0;
double Charges = 10.0;
CurrentAccount(String CustomerName, double AccNo, double Balance)
{
super(CustomerName, AccNo, Balance);
}
void withdraw(double Amount)
{
if(Balance >= Amount)
 {
 Balance -= Amount;
System.out.println(Amount + "Withdrawn successfully");
DisplayBalance();
}
else
{
System.out.println("insufficient Balance");
DisplayBalance();
}
else
{
System.out.println(" insufficient Balance");
}
}
 void UpdateBalance()
{
it(Balance <= M.inBalance)
{
Balance -= Charges;
System.out.println("Service charge applied for maintaining low
balance");
DisplayBalance();
 }
```

```
}
}
    class SavingsAccount extends Account
    {
        SavingsAccount(String CustomerName, double AccNo, double Balance)
        {
            super(CustberNme, AccNo, Balance);
        }
        double interest=0.05;
        void updateBalance()
        {
            Balance=Balance+(interest*Balance);
            DisplayBalance();
        }
void withdraw(double Amount)
{
    if (Balance>=Amount)
    {
        Balance-=Amount;
        DisplayBalance();
    }
else
    {
        System.out.println("insufficient Balance");
    }
}

class Bank {
    public static void main(String[])
    {
        String CustomerName;
        double AccNo, Balance;
        double amt, amtW;
```

```
Scanner in = new Scanner(System.in);
System.out.println("Enter name:");
CustomerName = in.next();

System.out.println("Enter AccNo");
AccNO = in.nextDouble();
System.out.println("Enter Balance:");
Balance = in.nextDouble();
System.out.println("enter amount to deposit");
amt = in.nextDouble();
System.out.println("enter amount to withdraw");
amt1 = in.nextDouble()
CurrentAccount c = new CurrentAccount(CustomerName, AccNo
                                      Balance);

c.Deposit(amt);
c.withdraw(amt1);
c.UpdateBalance();
system.out.println("");
SavingsAccount s = new SavingsAccount(CustomerName,
                                      AccNO, Balance);

S.Deposit(amt);
S.withdraw(amt1);
 S.UpdateBalance();
}
}
```

Output:

~~Enter AccNo.~~

Enter Name:

Nehal

Enter AccNO:

110

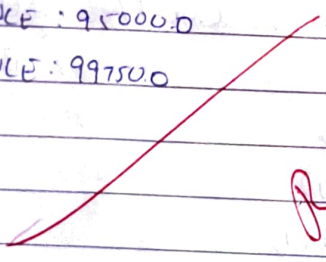
Enter balance:

90000

Enter amount to deposit
10000
Enter amount to withdraw
5000

DEPOSIT SUCCESSFUL
BALANCE: 100000.0
5000.0 WITHDRAWN SUCCESFULLY
BALANCE: 95000.0

DEPOSIT SUCCESFUL
BALANCE: 100000.0
BALANCE: 95000.0
BALANCE: 99750.0



21/12/24

23-1-24

PAGE NO

LAB-6

1. //Student

```
package CIE;
public class student
{
    public String name;
    public String usn;
    public int sem;

    public Student (String name, String usn, int sem)
    {
        this.name = name;
        this.usn = usn;
        this.sem = sem;
    }
}
```

//Internals

```
package CIE;

public class Internals extends CIE.Student
{
    public int[] InternalMarks;

    public Internals (String name, String usn, int sem, int [] InternalMarks)
    {
        super(name, usn, sem);
        this.InternalMarks = InternalMarks;
    }
}
```

```
//Externals
package SEE;
import CIE.Student;

public class Externals extends Student
{
    public int[] SeeMarks;
    public Externals( String name, String usn, int sem, int[] SeeMarks
    {
        super(name, usn, sem);
        this.SeeMarks = SeeMarks;
    }
}
```

```
//Final marks

import CIE.Student;
import CIE.Internals;
import SEE.Externals;
import java.util.Scanner;

public class FinalMarks
{
    public static void main(String[] args)
    {
        Scanner sl = new Scanner(System.in);
        System.out.println("Enter the number of Students");
        int n= sl.nextInt();

        String [] names = new String[n];
        String[] usn = new Sking[n];
        int [] sem = new int[n];
        int [][] InternalMarks = new int[n][5];
        int [][] SeeMarks = new int[n][5];
```

```
for(int i=0; i<n; i++)
{
    System.out.println("Enter details for student " + (i+1)+ ":");
    System.out.println("Name :");
    names[i] = sl.next();
    System.out.println("USN: ");
    usn[i] = sl.next();
    System.out.println("SEM :");
    sem[i] = sl.nextInt();
    System.out.println("Enter Internal Marks for 5 courses:");

for (int j=0; j<5; j++)
{
    System.out.println("Course " + (j+1)+ ":");
    InternalMarks[i][j] = sl.nextInt();
}
    System.out.println("Enter internal marks for 5 courses:");
for (int j=0; j<5; j++)
{
    System.out.println("Course " + (j+1) + ":");
    seeMarks[i][j] = sl.nextInt();
}

    int[][] FinalMarks = new int[n][5];
for (int i=0; i<n; i++)
{
    Internals Il = new Internals(names[i], usn[i], sem[i], InternalMarks[i]);
    Externals El = new Externals(names[i], usn[i], sem[i], seeMarks[i]);

for (int j=0; j<5; j++)
{
  FinalMarks[i][j] = Il.InternalMarks[i] + El.seeMarks[j];
}
  System.out.println("Final Marks for " + n + "Students in 5
```

```
marks:");
for(i=0;i<n;i++)
{
    System.out.println(names[i]+":");
  for(int j=0; j<5; j++)
{
    System.out.println(FinalMarks[i][j]+ ":");
}
    System.out.println();
}
sl.close();
}
}
}
```

Output:

```
Enter number of students: 1
Enter name: A
Enter USN: 1
Enter sec: 1
Enter internal marks for 5 courses:
30
10
9
10
9
Enter See marks for 5 courses
10
9
9
10
9
```

LAB - 7

Write a program that demonstrate handling of exceptions in inheritance tree. Create a base class called "Father" and derived class called "Son" which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge() when the input age<0. In Son class, implement a constructor that cases both father and son's age and throws an exception if son's age >= father's age.

```
import java.util.Scanner;

class WrongAge extends Exception {
    WrongAge(String message){
        super(message);
    }

}
class InputScanner {
    Scanner sc = new Scanner(System.in)
}
class Father extends InputScanner {
        Father() throws WrongAge{
        System.out.println("Enter Father's age");
        fatherAge= sc.nextInt();
        if(fatherAge<0){
            throw new WrongAge("Age cannot be negative");
        }

    }
    void display() {
        System.out.println("Father's age is " + fatherAge);
    }
}
```

```
class Son extends Father{
    int SonAge;
    Son() throws WrongAge{
        System.out.println("Enter Son's age");
        sonAge = s.nextInt();
        if(sonAge > FatherAge){
        throw new WrongAge("Son's age cannot be greater
            than father's age");
        }
        else if(sonAge<0){
          throw new WrongAge("Age cannot be negative");
        }
    }

{
  void display(){
    System.out.println("Son's age is" + sonAge);
  }
}

public class ExceptionHandling {
    public static void main(String[] args){
        try{
            Son son = new Son();
            son.display();
        }
        catch(WrongAge e){
        System.out.println("Exception" + e.getMessage());
        }
    }
}
```

Output:

Enter father's age

50

Enter son's age

21

Father's age is 50

Son's age is 21

Enter father's age

11

Enter son's age

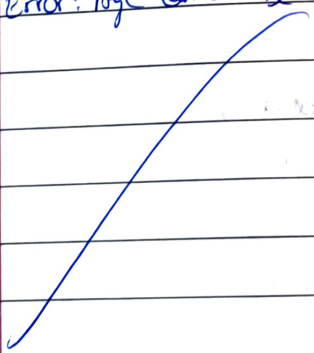
20

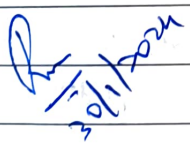
Error: Father's age must be greater than son

Enter father's age:

-1

Error: Age cannot be negative





6-2-24

PAGE NO.:

LAB - 5

Write a program which creates two threads, one thread displaying "BMS College of Engineering" once every two seconds.

```
class DisplayMessageThread extends Thread{
    private final String message;
    private final long interval;

    DisplayMessageThread (String message, long interval){
        this.message = message;
        this.interval = interval;
    }

    public void run(){
        try{
            while(true){
            System.out.println(message);
            Thread.sleep(interval);
            }
        }catch(InterruptedException e)
        {
            System.out.println(Thread.currentThread().getName()+
                                " interrupted");
        }
    }
}
```

```
public class ThreadExample {
    public static void main (String[] args) {
        DisplayMessageThread Thread1 = new DisplayMessageThread("BMS
                College of Engineering", 10000);
        DisplayMessageThread Thread2 = new DisplayMessageThread
                ("CSE", 2000);

        Thread1.setName("Thread 1");
        Thread2.setName("Thread 2");

        Thread1.start();
        Thread2.start();

        try {
            Thread.sleep(30000);
        } catch(InterruptedException e) {
            System.out.println("Main thread interrupted.");
        }

        Thread1.interrupt();
        Thread2.interrupt();

        System.out.println("Main thread exiting.");
    }
}
```

Output:

```
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
```

CSE

CSE

CSE

CSE

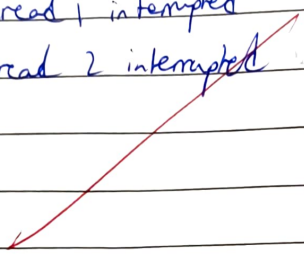
CSE

BMS College of Engineering

Main Thread exiting.

Thread 1 interrupted

Thread 2 interrupted



6/2/2024

20.2.24

LAB - 9

Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were zero, the program would throw an ArithmeticException. Display the exception in a message dialog box.

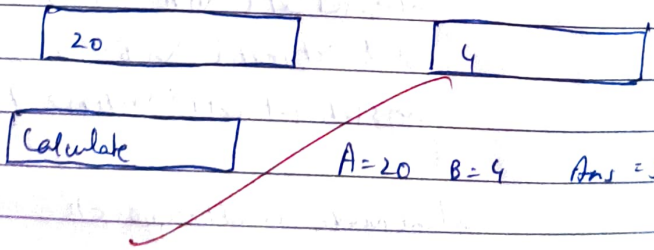
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class SwingDemo{
    SwingDemo(){
        JFrame jfrm= new JFrame("Divider App");
        jfrm.setSize(275, 150);
        jfrm.setLayout(new FlowLayout());
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel jlab= new JLabel("Enter the divider and dividend:");

        JTextField ajtf= new JTextField(8);
        JTextField bjtf = new JTextField(8);

        JButton button = new JButton("Calculate");

        JLabel err= new JLabel();
        JLabel alab= new JLabel();
        JLabel blab= new JLabel();
        JLabel anslab = new JLabel();
```

```
jfrm.add(err);
jfrm.add(alab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);

ActionListener l = new ActionListener(){
  public void actionPerformed(ActionEvent evt){
    System.out.println("Action event from a text field");
  }
};
ajtf.addActionListener(l);
bjtf.addActionListener(l);
button.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent evt)
{
    try{
      int a = Integer.parseInt(ajtf.getText());
      int b = Integer.parseInt(bjtf.getText());
      int ans = a/b;

      alab.setText("\n A= "+a);
      blab.setText("\n B= "+b);
      anslab.setText("\n Ans = "+ans);
    }
  catch(NumberFormatException e){
    alab.setText("");
    blab.setText("");
    anslab.setText("");
    err.setText("Enter Only Integers!");
    }
```

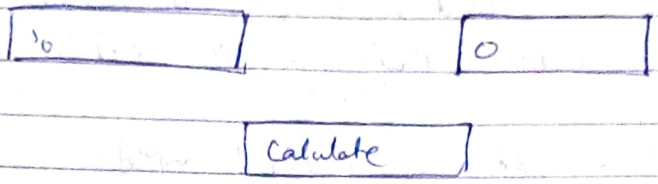
```
        catch(ArithmeticException e)
        {
            alab.setText(" ");
            blab.setText(" ");
            ans.setText(" ");
            exc.setText("B shall be NON zero!");
        }
    }
});
    jfrm.setVisible(true);
}

    public static void main (String args[])
    {
        SwingUtilities.invokeLater(new Runnable(){
            public void run(){
                new SwingDemo();
            }
        });
    }
}
```

Output.

Enter the divider and dividend



Enter the divider and dividend



B should be Non Zero!

AWT functions:

1. Jframe : It is a class in Java that is part of the swing library, which is used for creating graphical user interfaces in Java application.

2. SetSize(): setsize() is a method which is used with components such as Jframe Jpanel etc to set their size.

3. SetLayout(): It is a method which is used to set the layout manger for a container such as Jroom or any other container component.

4. Jlabel : It is a class which is used to display non-editable text or image on GUI.

5. Set default close operation : It is a method in Java swing used to specify the default close operation for a 'Jframe'.

6. JTextfield : class in Java Swing provides a text input field in a GUI It allows user to enter and edit single text line.

6-2-24

PAGE NO.:

LAB - 10

Demonstrate Inter process Communication and deadlock

```
class Q {
int n;
boolean valueSet = false;
synchronized int get(){
while(!valueSet)
try{
    System.out.println("\n Consumer Waiting\n");
wait();
} catch (InterruptedException e) {
    System.out.println("Interrupted Exception caught");
}
System.out.println("Got: " + n);
valueSet = false;
System.out.println("\n Producer waiting\n");
wait();
} catch(InterruptedException e) {
    System.out.println("Interrupted Exception caught");
}
this.n = n;
valueSet = true;
    System.out.println("Put: " + n);
    System.out.println("\n Intimate Consumer\n");
    notify();
}
}
class Producer implements Runnable {
Q q;
    Producer(Q q) {
            this.q = q;
            new Thread(this, "Producer").start();
        }
```

```
public void run(){
        int i=0;
        while(i<15){
        q.put(i++);
        }
    }
class Consumer implements Runnable{
Q q;
Consumer(Q q){
        this.q=q;
        new Thread(this,"Consumer").start();
}
public void run(){
    int i=0;
    while(i<15){
        int r=q.get();
        System.out.println("Consumed:"+r);
        i++;
        }
    }
}
class Main{
    public static void main(String args[]){
      Q q=new Q();
      new Producer(q);
      new Consumer(q);
      System.out.println("Press Ctrl C to stop.");
    }
}
```

PAGE NO

Output.

Press Control-C to stop.
Put : 0
Intimate Consumer
Producer waiting
Got : 0
Intimate Producer
Put : 1
Intimate Consumer
Producer waiting
Consumed : 0
Got : 1
Intimated Producer
consumed : 1
Put : 2
Intimate Consumer
Producer waiting
Got : 2
Intimate Producer
consumed : 2
Put : 3
Intimate Consumer
Producer waiting
Got : 3
Intimate Produce
Consumed : 3
Put : 4
Producer waiting
Got : 4
Intimate Producer
consumed : 4
Put : 5
Intimated Consumer
Producer waiting
Got : 5
Intimate Producer
Producer waiting
consumed : 5
Put : 6
Intimate Consumer
Producer waiting
Got : 6
Intimate Consumer
consumed : 6
Put : 7
Intimate Consumer
Producer waiting
Got : 7
Intimate Producer
consumed : 7
Put : 8
Intimate Consumer
Producer waiting
Got : 8
Intimate Producer
consumed : 8
Put : 9
Intimate Consumer
Producer waiting
Got : 9
Intimate Producer
consumed : 9
Put : 10
Intimate Consumer
Producer waiting

13-2-24

PAGE NO:

Deadlock

```
class A{
synchronized void foo(B b){
String name = Thread.currentThread().getName();
System.out.println(name + "entered A.foo");
try {
    Thread.sleep(1000);
}
catch(Exception e){
    System.out.println("A Interrupted");
}
System.out.println(name + "trying to call B last()");
b.last();
}
void last(){
    System.out.println("Inside A.last");
}
}
class B {
synchronized void bar(A a){
String name = Thread.currentThread().getName();
System.out.println(name + "entered B.bar");
try {
    Thread.sleep(1000);
}
catch(Exception e)
{
    System.out.println("B Interrupted");
}
System.out.println(name + "trying to call A.last()");
a.last();
}
```

```
    void last(){
        System.out.println("Inside A.last");
    }
}
class Deadlock implements Runnable
{
    A a = new A();
    B b = new B();
    Deadlock(){
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b);
        System.out.println("Back in main Thread");
    }
public void run(){
    b.bar(a);
    System.out.println("Back in other Thread");
}
public static void main(String args[])
{
    new Deadlock();
}
}
```

Output:

```
MainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call B.last()
Inside A.last
Back in main Thread
RacingThread trying to call A.last()
Inside A.last
Back in other Thread
```

13.02.24