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:

NIHAL MANJUNATH
1BM22CS178

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());
        }


    }
```

8. 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 a,b,c");
    a = s.nextInt();
    b = s.nextInt();
    c = s.nextInt();
  }
  void compute()
  {
    while (a==0)
    {
      System.out.println("Not a quadratic equation");
      System.out.println("Enter a non zero value of 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.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) / (2a);
r2 = Math.sqrt(-d) / (2a);
System.out.println("Root1 = " + r1 + "i");
System.out.println("Root1 = " + r1 + "-i");
}
}
}
class Quadratic Main
{
Public static void main (String arg[])
{
Quadratic q = new Quadratic();
q.getd();
q.compute();
}
}
```

Output:

```
Enter the coefficients of a, b, c
4 -3 2
Roots are real and distinct
Root1 = 2 Root2 = +4
```

Enter the coefficient of a, b, c
0 2 3
Not a quadratic
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

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 setMarks(){
int i;
For(i=0, i<9; i++){
System.out.println("Enter marks"+(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].
}
```

```
SGPA = (double) credits Gained / total credits;
}
void display result () {
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.getstudent Details (2);
    S1.get Marks ();
    S1.compute SGPA ();
    S1.display Result ();
    }
}
```

Output:

Enter Student name
Nehal
Enter USN
1D21CS176
Enter marks for subject 1
90
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
96
Enter credits for subject 5
3
Enter marks for subject 6
95
Enter credits for subject 6
2
Enter marks for subject 7
91
Enter credits for subject 7
1

Enter marks for subjects
or
Enter credits for subjects
!

Student name: Nehal
student usn : 4NM22CS179
SGPA : 9.5 90450

19/12/23

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 books 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" +
                     "No 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("No.of pages:");
int numPages = s.nextInt();

books[i] = new Book(name, author, price, numPages);
}
System.out.println("\n Book details:");
For (int i=0; i<n; i++) {
System.out.println("Book"+(i+1)+":\n"+books[i]);
}
}
}
```

Output:

Enter the number of books: 2
Enter details for Book 1
Name: Trump
Author: donald
Price: 500
Number of pages: 250
Enter details for Book 2
Name: Friends
Author: Pete
Price: 400
Number of pages: 220

Details of the books:
Book 1:
Book name: Trump
Author name: donald
Price: 500
number of pages: 250
Book 2:
Book name: Friends
Author name: pete
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 x1 extends AbsArea {
    xoc(int a, int b) {
      super(a, b);
    }
    void area() {
      system.out.println("The area of the rectangle is: " + a*b);
    }
}
class y extends AbsArea {
  y(int a, int b) {
    super(a, b);
  }
```

```
    void area() {
        System.out.println("The area of the triangle"
                + (c*1)/2);
    }
}

class C1a extends Abs Area {
    C1a(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 arg[]) {
        Abs Area r = new Rec(4, 4);
        r.area();
        Abs area t = new tri(4, 5);
        t.area();
        Abs area c = new C1a(2);
        c.area();
    }
}
```

Rev 9/11/2024

Output:

The area of the rectangle is: 12

The area of the triangle is: 10

The area of the circle is: 12.56

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 saving account provides compound interest and withdrawal facilities but no cheque book facility. The current account provide 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");
        Display balance()
        {
        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");
    Display Balance ();
    }
    else
    {
    System.out.println ("insufficient balance");
    Display Balance ();
    }
    else
    {
     System.out.println ("insufficient Balance");
    }
    }
    void UpdateBalance ()
    {
    if (Balance < MinBalance)
    {
     Balance -= charges;
    System.out.println ("Service charge applied for maintaining the balance");
    Display
```

```
}}
class SavingsAccount extends Account
{
    SavingsAccount(String CustomerName, double AccNo, double
    {
        super(CustomerName, AccNo, Balance);
    }
    double interest = 0.05;
    void UpdateBalance()
    {
        Balance = Balance + (interest * Balance);
        Display Balance();
    }
void withdraw(double Amount)
{
    if (Balance >= Amount)
    {
        Balance -= Amount;
        Display Balance();
    }
    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: ");
Customer Name = in.next();

System.out.println("Enter Acc no");
Acc No = 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 Name:
Nihal
Enter Acc No
110
Enter balance:
9000

Enter amount to deposit:
10000
Enter amount to withdraw
5000

DEPOSIT SUCCESSFUL
BALANCE: 1000000
5000.0 WITHDRAWN SUCCESSFULLY
BALANCE 950000

DEPOSIT SUCCESSFUL
BALANCE: 100000.0
BALANCE: 950000
BALANCE= 997506

16/1/24

Question

Date
Page

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[] IM
{
super(name, usn, sem);
this.InternalMarks = InternalMarks;
}
}

// Extend

```
package SE;
import CIE.student;

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

// Find marks

```
import CIE.student;
import CIE.InMarks;
import SE.Extend;
import java.util.Scanner;

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

        String[] name = 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 of student"+(i+1)+":");
System.out.println("Name:");
name[i]=sc.next();
System.out.println("USN");
usn[i]=sc.next();
System.out.println("Sem");
sem[i]=sc.nextInt();
System.out.println("Enter internal marks for 5 courses");

for(int j=0;j<5;j++)
{
System.out.println("Course"+(j+1)+":");
Internal_marks[i][j]=sc.nextInt();
}
System.out.println("Enter internal marks for 5 courses");
for(j=0;j<5;j++)
{
System.out.println("Course"+(j+1)+":");
see_marks[i][j]=sc.nextInt();
}
int[][] final_marks=new int[n][5];
for(int i=0;i<n;i++)
{
Internals I1=new Internals(name[i],usn[i],sem[i],internal_marks[i]);
External E1=new External(name[i],usn[i],sem[i],see_marks[i]);
for(int j=0;j<5;j++)
{
final_marks[i][j]=I1.internal_marks[j]+E1.see_marks[j];
}
System.out.println("Final marks for"+n+" students are");
```

```
for(i=0; i<n; i++)
{
  System.out.println(name[i] + ":");
  for(int j=0; j<5; j++)
  {
    System.out.println(final mark[i][j] + " ");
  }
  System.out.println();
}
sc.close();
}
}
```

Output:

Enter number of students:
2
Enter name:
A
Enter USN:
1
Enter Sem:
1
Enter internal marks for 5 courses:
30
10
9
10
9
Enter SEE marks for 5 course

10
9
9
10
9

Final marks of 5 courses:

LAB-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 that takes the age and throws the exception WrongAge if when the input age<0. In son class, implement a constructor that cases both father and son's age & throws an exception if son's age is >= father age

```
import java.util.Scanner;

class WrongAge extends Exception {
    WrongAge (String message){
        super(message);
    }
}
class Input Scanner {
    Scanner x = new Scanner(System.in)
}
class Father extends Input Scanner {
    Father() throws WrongAge {
    System.out.println("Enter Father age");
    fatherAge = x.nextInt();
    if(fatherAge <0){
        throw new WrongAge("Age Cannot be negative
    }
}
void display(){
System.out.println("Father age is" + fatherAge);
}
}
```

```
class son extends Father {
    int SonAge;
    Son() throws WrongAge {
    System.out.println("Enter son age");
    SonAge = x.nextInt();
    if (SonAge > FatherAge) {
    throw new WrongAge("Son age cannot be greater than
        Father 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 age is 50
Son age is 21

Enter father's age
11
Enter son's age
20
Error: Father's age must be greater than son

Enter father age:
-1
Error: Age cannot be negative

LAB- 8

Write a program which creates two threads, one thread displaying "BMS College of Engineering" once every ten seconds and another displaying "CSE" 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 Main {
    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

CSE
BMS college of Engineering
CSE
CSE
CSE
CSE
CSE
BMS college of Engineering
CSE
CSE
CSE
Main thread exiting
Thread1 interrupted
Thread2 interrupted

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("DivideApp");
    jfrm.setSize(275, 150);
    jfrm.setLayout(new FlowLayout());
    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CL

    JLabel jlab = new JLabel("Enter the two numbers");

    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();
```

```
jfm.add(ess)
jfm.add(jlab);
jfm.add(aytH)
jfm.add(bff);
jfm.add(button);
jfm.all(alab);
jfm.add(blab);
jfm.ado(ans);

ActionListener l = new ActionListener() {
    public void actionReferred(ActionEvent ae) {
        System.out.println("Action sent from a");
    }
};
a.jtf.addActionListener(l);
b.jtf.addActionListener(l);
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent ae)
    {
        try {
            int a = Integer.parseInt(ajtf.getText());
            int b = Integer.parseInt(bjtf.getText());
            int ans = a/3;

            alab.setText("\nA=" + a);
            blab.setText("\nB=" + b);
            ansLab.setText("\nAns=" + ans);
        }
        catch (NumberFormatException e) {
            alab.setText("");
            blab.setText("");
            ansLab.setText("");
            ej.setText("Enter only Integers");
```

```
catch (ArithmeticException e)
{
    a1ab.setText("");
    b1ab.setText("");
    ans1ab.setText("");
    err.setText("B should be non zero!");
}
});
f.frm.setVisible(true);
}
    public static void main(String args[])
    {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new SwingDemo();
            }});
    }
}
```

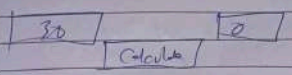
output

Enter the divider and dividend

20	4

calculate A=20 B=4 Ans=5

Enter the divisor 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 applications

2. Set Size (): setsize () is a method which is used with components such as JFrame JPanel etc used to set size

3. Set Layout () - It is a method which is used to set the layout manager for a container such as JFrame 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. JText field: class in Java swing Provides a text input field in a GUI It allows user to enter and edit single text line.

6-02-24

LAB - 10

Demonstrate interprocess 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("InterruptedException caught");
}
System.out.println("Got: " + n);
valueSet = false;
System.out.println("\n Intimate Producer\n");
notify();
return n;
}
synchronized void put(int n) {
while(valueSet)

try {
    System.out.println("\n Producer waiting\n");
wait();
} catch (InterruptedException e) {
System.out.println("InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
```

```
System.out.println("Put Interrupted");
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++;
}
}
```

Date
Page

```
}
class Main {

public Static void main (String arg[]) {
Q q = new Q();

new producer (q);
new Consumer (q);

System.out.println("press Control-C to stop");
}
}
```

not executed

8/2/2024.

Output:

```
pes control -C to stop

put: 0
Initiate consumer
producer waiting
Got: 0
Intimate producer
Put: 1
Intimate Consumer
Producer waiting
Consumed: 0
Got: 1
Intimated Produce
Consumed: 1
put: 2
Intimate Consumer
Producer waiting
Got: 2
Intimate Producer
Consumed: 2
Put: 3
Intimate Consumer
Producer Waiting
Got: 3
Intimate Producer
Consumed: 3
```

Deadlock

```
class A {
synchronised 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 {
synchronised 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("Main Thread");
        Thread t = new Thread(this, "Racing Thread");
    }

public void main() {
    b.bar(a);
    System.out.println("Back in other Thread");
}

public static void main(String args[])
{
    new Deadlock();
}
}
```

Output:

Main Thread entered A.foo
Racing Thread entered B.bar
Main Thread trying to call B.last()
Inside A.last
Back in main thread
Racing Thread trying to call A.last()
Inside A.last
Back in other thread

13.09.24