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:

NAVNEETH K S

1BM22CS174

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	12-12-23
2.	Lab-2	19-12-23
3.	Lab-3	26-12-23
4.	Lab-4	02-01-24
5.	Lab-5	09-01-24
6.	Lab-6	16-01-24
7.	Lab-7	23-01-24
8.	Lab-8	30-01-24
9.	Lab-9	06-01-24
10.	Lab-10	20-01-24

12/12/23

2, 3, 5, 7

DOMS | Page No. | Date / /

prg-1
Quadratic-Equation

```
import java.util.*;
class Quadratic{
  double a,b,c,d;
  Quadratic(double a,double b,double c){
    this.a=a;
    this.b=b;
    this.c=c;
  }
  void get_root(){
    if(a==0){
      system.out.println("a can't be zero");
    }
    if(d==0){
      System.out.println("Equal Roots");
      System.out.println("The roots are "+
        (-b/(2*a)));
    }
    else if(d>0){
      double r1=(-b+Math.sqrt(d))/(2*a);
      double r2=(-b+math.sqrt(d))/(2*a);
      System.out.println("The root 1 and
      root 2 are,"+r1,r2);
    }
    else{
      double r1=(-b)/(2*a);
      double r2=(-d)/(2*a);
      System.out.println(r1+"+i"+r2);
      System.out.println(r2+"-i"+r2);
    }
}
```

math.sqrt(-d)/(2*a);

$(x-2)(2+5)$
x^2+x-1

DOMS | Page No.
Data / /

```
public static void main(String args[]){
  System.out.println("the value of
  a,b and c");
  Scanner in = new Scanner(System.in);
  double a,b,c;
  a = in.nextDouble();
  b = in.nextDouble();
  c = in.nextDouble();
  Quadratic q = new Quadratic(a,b,c
  q.get_root();
  }
}
```

output:

① The value of a, b, c
0 1 2
a cant be zero

② The value of a, b, c
1 -2 1
Equal roots
The roots are +1

$\frac{8\pi}{12}\left|\frac{12}{12}\right|\frac{23}{}$

③ The value of a, b, c
1 1 -6
The root1 and root2 are 2 3

④ The value of a, b, c
1 2 10
The root1 is -1.0 + i 18
The root2 is -1.0 - i 18

19/12/23

Lab-8

2 Develop JAVA prg to develop class with member usn, name, an array credits and array marks. Include methods to accept and display details and a method to calculate SGPA.

```
class SGPA{
B- import java.util.Scanner;
   import java.util.Arrays;

   class SGPA{          // constructor
      String usn,name;
      int[] credits,marks;
   }

   public double calculate(){ // sgpa calculate
      double tot_credits = 0.0;
      double tot_sum = 0.0;
      for(int i=0; i<8; i++){
         tot_credits += credits[i];
         tot_sum = grade(marks[i])*credits[
      }
      return tot_sum/tot_credits;
   }

   private double grade(int marks){
      if(marks >= 90) return 10;
      else if(marks >=80) return 9;
      else if(marks >=70) return 8;
      else if(marks >=60) return 7;
      else if(marks >=50) return 6;
      else if(marks >=40) return 5;
      else return 0.0;
   }
```

```
public static void main(String arg[]){
  Scanner in = new Scanner(System.in);
  int[] credits = new int[8];
  int[] marks = new int[8];

  sout("Enter credits: ");
  for(int i=0; i<8; i++){
    credits[i] = in.nextInt();
  }
  sout("Enter marks: ");
  for(int i=0; i<8; i++){
    marks[i] = in.nextInt();
  }
  SGPA nav = new SGPA("CS174",
  "Navneeth", credits, marks);
  nav.acc
```

```
public void accept(Scanner in){
  for(int i=0; i<8; i++){
    credits[i] = in.nextInt();
  }
  sout("marks: ");
  for(int i=0; i<8; i++){
    marks[i] = in.nextInt();
  }
}
```

// accepts marks & credits

```
public static void main(Strings args[]){
    Scanner in = new scanner(system.in);
    SGPA nav = new SGPA
    ("CS174", "Nameeth KS", credits,
     marks);   // object
    nav.accept(in);
    System.out.println(Arrays.tostring
                       (nav.credits));
    system.out.println(nav.calculate());
}
```

output:

credits: 4 4 3 3 3 1 1 1
marks: 100 100 90 90 90 70 70 70

[4, 4, 3, 3, 3, 1, 1, 1]

9.7

19/12/2023

3. Create a class Book and set methods

```
class Book{
    String name;
    String author;
    double price;
    int page;

    Book(String name, String Author,
double price, int page){
    this.name = name;
    this.author = author;
    this.price = price;
    this.page = page;
}

public void setName(String name){
    this.name = name;
}

public String getName(){
    return name;
}

public void setAuthor(String author){
    this.author = author;
}

public void setPrice(double price){
    this.price = price;
}
```

```
public String getAuthor(){
    return String;
}

public double getPrice(){
    return price;
}

public void getpages(int pages){
    this.page = page;
}

public int getPage(){
    return page;
}

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

public String toString(){
    String n, a, p, N;
    n = "\n" + "Name of Book: " + name + "\n";
    a = "author" + "\n";
    p = "price of book: Rs" + price + "\n";
    N = "Number of pages: " + pages + "\n";
    return n + a + p + N;
}
```

```
public static void main(String[]
args){
  Scanner sc = new Scanner(System.in)
  System.out.print("Enter details
  for BOOKS " + (i+1) + ": ");

  int n = sc.nextInt();
  BOOK[] books = new BOOK[n];

  for(int i=0; i<n; i++){
    sout("Enter details for Book "+
    (i+1) + ":");
    sout("Name: ");
    String name = sc.next();
    sout("Author: ");
    String author = sc.next();
    SOUT("Price: Rs");
    double price = sc.nextDouble();
    sout("Number of Pages");
    int pages = sc.nextInt();

    books[i] = new BOOK(name,
    author, price, pages);

  for(int i=0; i<n; i++){
    sout("\nBook" +(i+1) + "Details:\n"+
    books[i].toString());
  }
}
```

output:

Enter the number of books: 2
Enter details for BOOK 1
Name: Phy
Author: hc_verma
Price: Rs 1243
page: 865

Enter the details for Book 2
Name: maths
Author: Rd-sharma
price: Rs 3456
pages: 865

Book 1 details
Name of book: Phy
Author: hc-verma
price: Rs 1234
pages: 345

Book 2 details
Name of book: maths
Author: rd-sharma
price: Rs 3456
pages: 865

invalid o/p!

Rs 21120/-

Lab - 4

2. Implement abstract class

```
abstract class AbsArea {
    int a, b;
    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: " + a*b);
    }
}

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

```
class tri extends AbsArea{
    tri(int a, int b){
        super(a,b);
    }
    void area(){
        System.out.println("The area of the
        triangle" + (a*b)/2);
    }
}

class Main{
    public static void main(String args[]){
        AbsArea r = new rec(3,4);
        r.area();

        AbsArea t = new tri(4,5);
        t.area();

        AbsArea c = new cri(2);
        c.area();
    }
}
```

output:

The area of rectangle is: 12
The area of triangle is: 10
The area of circle is: 12.56

21/2024

savings → compound & withdraw

DOMS | Page No.
Date 7/1/24

lab-5

Develop a JAVA program to class Bank that maintains 2 kinds of account.

```
class Account {
    String name;
    int accno;
    String type;

    Account (String name, int accno,
                String type) {
        this.name = name;
        this.accno = accno;
        this.type = type;
    }

class sav extends Account {
    float balance;

    sav (String name, int accno, String ty

        super(name, accno, type

    void accept
```

```
class Account {
    String name;
    int accno;

    Account(String name, int accno) {
        this.name = name;
        this.accno = accno;
    }
}

class sav extends Account {
    float balance;
    sav(String name, int accno) {
        super(name, accno);
    }

    void deposite(float amt) {
        balance += amt;
    }

    float withdraw(float amt) {
        ~~balance -= amt;~~
        if (balance >= amt) {
            balance -= amt;
            return amt;
        } else {
            sout("insufficient balance");
            return 0.0f;
        }
    }
}
```

$P(1+\frac{R}{100})^n$

```
float compound(int y){
    return balance*(float)Math.pow
                        (1.08,y);
}

void details(){
    System.out.println("
"Name:"+name+", AccNo:+",
Balance:", balance);
}
}

class Cur extends Account{
    float bal;
    float min;

    Cur(String name, int accno, float min
        super(name, accino);
        this.min = min
}

void depositecheque(float amt){
    Balance += amt;
}

float withdraw(float amt){
    if(balance-amt){
        balance -= amt
        return amt balance;
    }else{
        Print("
}
```

```
void details(){
 sout
}

float penalty(){
  if(balance<min){
    sout("\n insufficient balance");
    return 100;
  else
    return 0;
  }
}

class Bank
 public static void main(String args[]){
  sout("This the work of Naveeth KS\
 n 1BM22CS174");
  int x, y;
  Scanner = new Scanner(System.in);

  sout("Give options
 public static void opt(Account s){
  int c;

 switch(c){
  case 1:
    sout("Enter deposite amount:");
    float amt = in.nextFloat();
    s.deposite(amt);
    break;
```

```
case 2:
sout("Enter withdrawal amount:");
float amt = in.nextFloat();
float with = s.withdraw(amt);
sout("Amount is " + with);
break;

case 3:
sout("Enter the no. of years:");
int y = in.nextInt();
sout("The compound is:" + s.compound(amt,
break;

case 4:
sout("Enter amount:");
float amt = in.nextFloat();
s.depositcheque(amt)

case 5:
s.details();
break;

case 6:
s.penalty();
break;

case 7:
sout("Exiting...");
break
} while(c!=7)
```

output

Options:
1. Deposite
2. Withdraw
3. Compound
4. Cheque
5. Details
6. Penalty
7. Exit
1
Enter amount 10000
2
Enter withdraw amount
3000
3
Enter the no. of years
4
~~Pre~~ ~~Cnt~~ is invalid method
5
Name: Nawneeth, Acc No: 1001, Balance: 9200
6
Invalid method
7
exiting.

16/1/25

Practice

Output
11-6
11
Welc
[87, 101, 108, 99, 111]
true
false
true
false
sorted words
apple
ball
Cat
dog
Original String buffer: Hello, World
After setLength(5): Hello
charAt(1): e
After setCharAt(1, 'E'): HElLo
getChars(0, 5, charArray, 'E'): Hello
0 17
Enter the details for student 1
Regno : 1001
Name : Name KS
Semester : 1
CGPA : 9.3

DOMS | Page No.
Date 30 / 1 / 24

Week 7

Q. Create Father and Son class and take input age and give out Exception

```
import java.util.*
class WrongAge extends Exception{
    WrongAge(String msg){
        super(message);
    }
}

class Father{
    int age;
    Father(int age) throws WrongAge{
        if(age<0){
            throw new WrongAge("Age can't
            be negative");
        }
        this.age = age;
    }
}

class Son extends Father{
    int sage;
    Son(int age, int sage) throws WrongAge{
        super(age);
        if(sage >= age){
            throws new WrongAge("Son's Age
            should be less than Father's age");
        }
        this.sage = sage;
    }
}
```

Date / /

```
public class Error2
 public static void main (System.in);
 try{
    System.out.println("Enter the
    daddy's age:");
    int a = sc.nextInt();
    Father f = new Father(a);
    system.out.println("Enter the
    sonny's age:");
    int b = sc.nextInt();
    Son son = new Son(a,b);
 }Catch(WrongAge e){
    System.out.println("Exception"
  + e.getmessage());
  }
 }
}
```

output:

Enter the daddy's age: -110
Exception: Age can't be negative

Enter the daddy's age: 45
Enter the sonny's age: 46
Exception: Son's age must be less dad's age

Enter the daddy's age: 50
Enter the son's age: 25

30/1/20

6/2/24

Page No.
Date / /

week-8

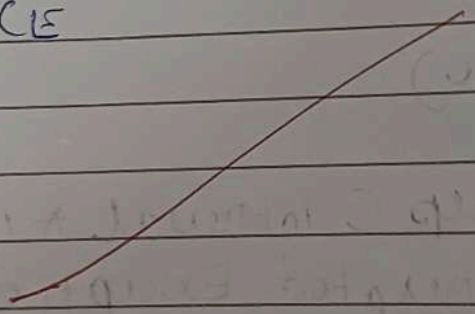
```
class DisplayThread extends Thread {
    private String message;
    private int interval;

    public DisplayThread(String message,
    int interval) {
        this.message = message;
        this.interval = interval;
    }

    public void run() {
        while(true) {
            sout(message);
            try {
                Thread.sleep(interval * 1000);
            } catch(InterruptedException e) {
                e.getMessage();
            }
        }
    }
}

public class main {
    public static void main(String[] args) {
        DisplayThread thread1 = "BMSCE", 10";
        DisplayThread thread2 = "CSE", 2;

        thread1.start();
        thread2.start();
    }
}
```

output:

BMSCE
CSE
CSE
:

BMSCE
CSE
CSE
:

BMSCE
:
:



6/2/2024

6/2/24

DOMS | Pages No.
Date / /

Interprocess communication
Lab-16

```
class Q {
  int n;
  boolean valueset = false
  synchronized int get() {
    while(!valueset){
      try{
        wait();
      } catch(InterruptedException e){
        sout("caught");
      }
      sout("Got: " + n);
      valueset = false;
      notify();
      return n;
  }

synchronized void put(int n){
  while(valueset)
    try{
      wait();
    } catch(InterruptedException e){
    sout("caught");
    }
    this.n = n;
    valueset = true
    notify();
  }
}
```

```
class Procedure implements Runnable{
  Q q;
  Procedure(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<5){
      int n = q.get();
      i++;
    }
  }
}
```

output:

put: 1
not: 1
put: 2
got: 2
put: 3
got: 3
put: 4
got: 5
put: 5
got: 5

13-27-24 lab-10

deadlock

```
class A {
    synchronized void foo(B b) {
        String name = Thread.currentThread().
        System.out.println(name + " entered A.foo");

        try {
            Thread.sleep(100);
        } catch (Exception e) {
            System.out.println("A Interrupted");
        }

        System.out.println(name + " trying to call B.last()");

    }

    void last() {
        System.out.println("inside A.last");
    }
}

class B {

    synchronized void bar(A a) {
        String name = Thread.currentThread().
        System.out.println(name + " entered B.bar");

        try {
            Thread.sleep(1000);
        }
```

```
        catch(Exception e){
            sout("B interrupted");
        }
        sout(name + " trying to call A.last()");
        a.last();
    }

    void last(){
        sout("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");
        t.start();
        a.foo(b);
        sout("Back in main thread");
    }

    public void run(){
        b.bar(a);
        sout("Back in other thread");
    }
    pubic static void main(String args[]){
        new DeadLock();
    }
}
```

output

```
mainThread entered A.foo
RacingThread entered B.bar
mainThread trying to call B.last()
Inside B.last
Back in mainThread
RacingThread trying to call A.last()
Inside A.last
Back in otherThread
```

13/2/2024

20-2-24

week-9

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

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

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

  JTextField ajtf = new JTextField(8);
  JTextField ajtf = 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(jlab);
  jfrm.add(atf);
  jfrm.add(bjtf);
```

```
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);

ActionListner l = new ActionListner(){
 public void
          actionPerform(ActionEvent
                              eVt){
   sout("Action went from text field");
 }
};
ajtf.addActionListner(new Action
ajtf.addActionListner(l);


button.addActionListner(new Action List
                              ner(){

 try{
  int a = Integer.parseInt(ajtf.getTex
  int b = Integer.parseInt(bjtf.getTex
  int ans = a/b;

  alab.setText("");
  blab.setText("");
  anslab.setText("");
  jfrm.setText("Integer");
 }
 catch(ArithmeticException e){
  alab.setText("");
```

```
        blab.setText(" ");
        anslab.setText(" ");
        chn.setText(" B shd be NON zero ");
    }
    }
});

jfrm.setVisible(true);

psvm (String args[])
{
    SwingUtilities.invokeLater(new Runnable
    ()
    {
        public void run()
        {
            new swingDemo();
        }
    });
}
}
```

Output:

Enter the dividen and divident:

39 | 13

Calculate

A = 39 B = 13 Ans = 3

22/2/24

Lab-6

```
package CIE;
public class Student{
    String USN;
    String name;
    int sem;
    public Student(String usn, String na
      this.usn = usn;
      this.usn = usn;
      this.usn = usn;
    }
}
```

```
package CIE;
public class Internals extend Student{
  public int[] internalmarks;
  public Internals(String usn, String nam
      int sem, int[] internalmarks){
    super(usn, name, sem);
    this.internalmarks = internalmark
  }
}
```

```
package SEE;
import CIE.student;
public class External extends Student{
  public int[] seemarks;
  public External(String usn, String nam
    int sem, int[] semarks){
```

```
        super(usn, name, sem);
        this.seemarks = seemarks;
    }
}

import CIE.Internals
import SEE.Externals
import java.util.*;
public class finalmarks{
    public static void main (String[] args){
        Scanner sc = new scanner(System.in);
        sout("Enter no of students");
        int n = sc.nextInt();
        Internals[] cies = new Internals[n];
        External[] sics = new External[n];
        for(int i=0; i<n; i++){
            sout("Enter CIE details"+(i+1));
            sout("USN:")
            String usn = sc.nextInt();
            sout("Name:");
            sout(" String name = sc.nextInt();
            sout("scm");
            int scm = sc.nextInt();
            int[] cicmarks = new int[5];
            sout("Enter the CIE marks");
            for(int i=0; i<5; i++){
                cicmarks[i] = sc.nextInt();
            }
            ciestudent[i] = new Internals(usn, name,
                                          scm);
        }
    for(int i=0; i<n; i++){
```

```
Date / /

for
  sout("Enter SEE details" + (i+1));
  sout("USN");
  String usn = scanner.nextInt();
  sout("Name");
  sout("scm")
  int scm = sc.nextInt();
  int[] seemarks = new int[5];
  sout("Enter SEE marks for 5 subj
  for(int j=0; j<5; j++){
    seemarks[j] = sc.nextInt();
  }
  ~~seestudent[i] = new~~
  }
}
```

Output:

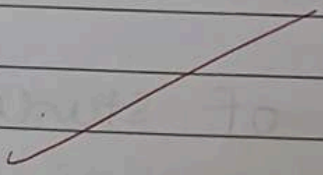
Enter the number of student : 2
Enter the details of CIE of student 1.
USN : 1BM22CS001
name : Avinash
scm : 4
Enter CIE marks for 5:
34
39
37
40
42

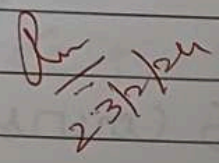
DOMS | Page No.
Data / /

Enter details for CIE of student 2
USN : 1BM22CS003
name : Chandan
scm : 3
Enter CIE marks for 5 course : 20
14
21
18
30
Enter details for SEE of student 1
USN : 1BM14CS001
name : Avinash
scm : 4
Enter see marks for 5 courses : 40
37
43
23
43
Enter see details for student 2
USN : 1BM22CS005
name : Bhuvan
scm : 7
enter see marks for 5 courses :
50
49
45
47
46

23/2/24

Details of Student 1	Student 2
USN: 1BM22CS001	1BM22CS003
name: avinash	chandan
Scm: 4	Scm: 7
CIE marks:	CIE marks
49 34 34 37 40	20 14 26 18 30
See marks:	SEE marks
40 37 43 23 45	50 99 48 47 46





WEEK - 1

1. **Develop a Java program that prints all real solutions to the quadratic equation ax2+bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminate b2 -4ac is negative, display a message stating that there are no real solutions.**

Program:

```
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("Prajwal P\n1BM22CS200");
      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);
        }

    }
}
```

WEEK - 2

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.

Program:

```
import java.util.Scanner;
import java.util.Arrays;
```

```
class SGPA {
    String usn, name;
    int[] credits;
    int[] marks;

    SGPA(String usn, String name, int[] credits, int[] marks) {
        this.usn = usn;
        this.name = name;
        this.credits = credits;
        this.marks = marks;
    }

    // Method to calculate SGPA
    public double calculateSGPA() {
        double totalCredits = 0.0;
        double weightedSum = 0.0;

        for (int i = 0; i < credits.length; i++) {
            totalCredits += credits[i];
            weightedSum += calculateGradePoints(marks[i]) * credits[i];
        }

        return weightedSum / totalCredits;
    }

    private double calculateGradePoints(int marks) {
        if (marks >= 90) {
            return 10.0;
```

```
    } else if (marks >= 80) {
        return 9.0;
    } else if (marks >= 70) {
        return 8.0;
    } else if (marks >= 60) {
        return 7.0;
    } else if (marks >= 50) {
        return 6.0;
    } else if (marks >= 40) {
        return 5.0;
    } else {
        return 0.0;
    }
}

public void accept(Scanner in) {
    System.out.println("Enter 8 credits:");
    for (int i = 0; i < 8; i++) {
        credits[i] = in.nextInt();
    }

    System.out.println("Enter 8 marks:");
    for (int i = 0; i < 8; i++) {
        marks[i] = in.nextInt();
    }
}

public static void main(String args[]) {
    Scanner scanner = new Scanner(System.in);
```

```
        // Initialize arrays for credits and marks
        int[] creditsArray = new int[8];
        int[] marksArray = new int[8];

        SGPA nav = new SGPA("123", "Navneeth KS", creditsArray, marksArray);
        nav.accept(scanner); // Pass the Scanner object to the accept method
        System.out.println("Marks array: " + Arrays.toString(nav.marks));
        System.out.println("SGPA: " + nav.calculateSGPA());
    }
}
```

WEEK-3

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.**

Program:

```
import java.util.Scanner;

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

    // Constructor to set the values for the members
    public Book(String name, String author, double price, int numPages) {
        this.name = name;
        this.author = author;
```

```
    this.price = price;
    this.numPages = numPages;
}

// Methods to set and get the details of the objects
public void setName(String name) {
    this.name = name;
}

public String getName() {
    return name;
}

public void setAuthor(String author) {
    this.author = author;
}

public String getAuthor() {
    return author;
}

public void setPrice(double price) {
    this.price = price;
}

public double getPrice() {
    return price;
}

public void setNumPages(int numPages) {
    this.numPages = numPages;
}

public int getNumPages() {
    return numPages;
}

// toString() method to display the complete details of the book
public String toString() {
String n = "\n" + "Name of Book: " + name + "\n";
String a = "Author of Book: " + author + "\n";
```

```
    String p = "Price of Book: Rs" + price + "\n";
    String N = "Number of pages: " + numPages + "\n";
    return n + a + p + N;
}

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

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

        // Create an array to store n book objects
        Book[] books = new Book[n];

        // Input details for each book
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for Book " + (i + 1) + ":");
            System.out.print("Name: ");
            String name = scanner.next();
            System.out.print("Author: ");
            String author = scanner.next();
            System.out.print("Price: Rs");
            double price = scanner.nextDouble();
            System.out.print("Number of Pages: ");
            int numPages = scanner.nextInt();

            // Create a book object with the input details
            books[i] = new Book(name, author, price, numPages);
        }

        // Display details of each book
        for (int i = 0; i < n; i++) {
            System.out.println("\nBook " + (i + 1) + " Details:\n" +
books[i].toString());
        }

        scanner.close();
    }
}
```

WEEK-4

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.**

Program:

```
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 the work of Navneeth KS\n1BM22CS174");
        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();
   }
}
```

WEEK-5

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. Create a class Account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:**

   a) Accept deposit from customer and update the balance.
   b) Display the balance.
   c) Compute and deposit interest
   d) Permit withdrawal and update the balance Check for the minimum balance, impose penalty if necessary and update the balance.

Program:

```
import java.util.*;

class Main {
   static Scanner in = new Scanner(System.in);
   public static void main(String args[]) {
      Account a = new Cur("Navneeth",1001,20000);
      Account a1 = new Sav("Pranav",1002,10000);
      opt(a);



   }
   public static void opt(Account s){
      int choice;
      do {
         System.out.println("Options:");
         System.out.println("1. Deposit");
         System.out.println("2. Withdraw");
         System.out.println("3. Compound");
         System.out.println("4. Cheque");
         System.out.println("5. Details");
         System.out.println("6. Penalty");
         System.out.println("6. Exit");

         System.out.print("Enter your choice: ");
         choice = in.nextInt();

         switch (choice) {
```

```
case 1:
   System.out.print("Enter deposit amount: ");
   float amtDeposit = in.nextFloat();
   s.deposit(amtDeposit);
   break;

case 2:
   System.out.print("Enter withdrawal amount: ");
   float amtWithdraw = in.nextFloat();
   float withdrawnAmount = s.withdraw(amtWithdraw);
   System.out.println("Amount withdrew is: " + withdrawnAmount);
   break;

case 3:
   System.out.print("Enter number of years for compound interest: ");
   int years = in.nextInt();
   System.out.println("The compound is: " + s.compound(years));
   break;

case 4:
   System.out.print("Enter amount: ");
   float amt = in.nextFloat();
   s.depositCheque(amt);
   break;

case 5:
   s.details();
   break;
```

```
            case 6:
                s.penalty();
                break;

            case 7:
                System.out.println("Exiting...");
                break;

            default:
                System.out.println("Invalid choice. Please enter a valid option.");
        }

    } while (choice != 7);
  }
}

abstract class Account {
  String name;
  int accno;

  Account(String name, int accno) {
    this.name = name;
    this.accno = accno;
  }

  abstract void deposit(float amount);

  abstract float withdraw(float withdraw);
```

```
    abstract float compound(int years);

    abstract void details();

    abstract void depositCheque(float amount);

    abstract void penalty();
}

class Sav extends Account {
    float balance = 0.0f;

    Sav(String name, int accno, float initialBalance) {
        super(name, accno);
        this.balance = initialBalance;
    }

    void deposit(float amount) {
        balance += amount;
    }

    float withdraw(float amount) {
        if (balance >= amount) {
            balance -= amount;
            return amount;
        } else {
            System.out.println("Insufficient bank balance");
            return 0.0f;
        }
```

```
    }

    float compound(int years) {
        return balance * (float) Math.pow(1.08, years);
    }

    void penalty() {
        System.out.println("Invalid methods\n");

    }

    void details() {
        System.out.println("Name: " + name + ", AccNo: " + accno + ", Balance: " + balance);
    }
    void depositCheque(float amount) {
        System.out.println("Invalid methods\n");
        return;
    }
}



class Cur extends Account {
    float balance = 0.0f;
    float min;

    Cur(String name, int accno, float min) {
        super(name, accno);
        this.min = min;
```

```
}
void deposit(float amount) {
   depositCheque(amount);
}

void depositCheque(float amount) {
   balance += amount;
}

float withdraw(float amount) {
   balance -= amount;
   return amount;
}

void details() {
   System.out.println("Name: " + name + ", AccNo: " + accno + ", Balance: " + balance);
}

float compound(int years) {
   System.out.println("Invalid methods\n");
   return 0;

}

void penalty() {
   if (balance < min) {
      System.out.println("Insufficient balance, penalty of 1000 added");
   } else {
      System.out.println("Proper balance");
```

```
    }
  }
}
```

WEEK-6

6. **Create a package CIE which has two classes- Student and Internals. The class Personal has members like usn, name, sem. The class internals 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.**

Program:

```
import CIE.Internals;
import
SEE.External;
import
java.util.Scanner;


public class MainProgram {


    public static void main(String[] args) {
        Scanner scanner = new
        Scanner(System.in);
```

```
System.out.println("Enter the number of
students:"); int numStudents = scanner.nextInt();


Internals[] internalsArray = new
Internals[numStudents]; External[] externalsArray =
new External[numStudents];


for (int i = 0; i < numStudents; i++) {
   System.out.println("Enter details for Student " + (i +
   1)); System.out.print("USN: ");
   String usn = scanner.next();


   System.out.print("Name: ");
   String name =
   scanner.next();


   System.out.print("Semester:
   "); int sem =
   scanner.nextInt();
   System.out.println("Enter Internal Marks for 5
   courses:"); int[] internalMarks = new int[5];
   for (int j = 0; j < 5; j++) {
      System.out.print("Course " + (j + 1) + ": ");
      internalMarks[j] = scanner.nextInt();
   }


   internalsArray[i] = new Internals(usn, name, sem, internalMarks);
   System.out.println("Enter SEE Marks for 5 courses:");
```

```
        int[] seeMarks = new
        int[5]; for (int j = 0; j < 5;
        j++) {
            System.out.print("Course " + (j + 1) + ": ");
            seeMarks[j] = scanner.nextInt();
        }

        externalsArray[i] = new External(usn, name, sem, seeMarks);
    }

    System.out.println("\nFinal Marks of
    Students:"); for (int i = 0; i < numStudents;
    i++) {
        int[] internalMarks =
        internalsArray[i].getInternalMarks(); int[] seeMarks =
        externalsArray[i].getSeeMarks();

        int[] finalMarks = new
        int[5]; int totalMarks = 0;

        System.out.println("Student " + (i + 1) + " - " +
internalsArray[i].getName());

        for (int j = 0; j < 5; j++) {
            finalMarks[j] = internalMarks[j] + seeMarks[j];
            totalMarks += finalMarks[j];
            System.out.println("Course " + (j + 1) + ": " + finalMarks[j]);
        }
```

```
            System.out.println("Total Marks: " + totalMarks + "\n");
        }
    }
}
package CIE;


public class Internals extends Student {
    protected int[] internalMarks;


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


    public int[]
        getInternalMarks() {
        return internalMarks;
    }
}


//
CIE/Student.ja
va package
CIE;


public class Student {
    protected String
    usn; protected
```

```
    String name;
    protected int sem;


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


    public
        Student() {
        this("", "",
        0);
    }


    public String
        getName() { return
        name;
    }
}
package SEE;


import CIE.Student;


public class External extends Student {
    protected int[] seeMarks;
```

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

    public int[] getSeeMarks() {
      return seeMarks;
    }
  }
```

WEEK-7

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.

Program:

```
import java.util.*;
class WrongAge extends Exception {
  WrongAge(String message) {
    super(message);
  }
}

class Father {
  int age;
```

```
    Father(int age) throws WrongAge {
        if (age < 0) {
            throw new WrongAge("Age cannot be negative");
        }
        this.age = age;
    }
}



class Son extends Father {
    int sage;

    Son(int fatherAge, int sonAge) throws WrongAge {
        super(fatherAge);

        if (sonAge >= fatherAge) {
            throw new WrongAge("Son's age should be less than Father's age");
        }

        this.sage = sonAge;
    }
}

public class Error {
    public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
        try {
    System.out.println("Enter the daddy's age: ");
    int a = sc.nextInt();
```

```
        Father father = new Father(a);
    System.out.println("Enter the sonny's age: ");
        int b = sc.nextInt();
        Son son = new Son(a, b);
      } catch (WrongAge e) {
        System.out.println("Exception: " + e.getMessage());
      }
    }
}
```

WEEK-8

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.**

Program:

```
class DisplayThread extends Thread {
    private String message;
    private int interval;

    public DisplayThread(String message, int interval) {
        this.message = message;
        this.interval = interval;
    }

    public void run() {
        while (true) {
```

```
            System.out.println(message);
            try {
                Thread.sleep(interval * 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class Main {
    public static void main(String[] args) {
        DisplayThread thread1 = new DisplayThread("BMS College of Engineering", 10);
        DisplayThread thread2 = new DisplayThread("CSE", 2);

        thread1.start();
        thread2.start();
    }
}
```

WEEK-9

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 Arithmetic Exception Display the exception in a message dialog box.

Program:

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


class
   SwingDemo
   {
   SwingDemo(
   ){
      JFrame jfrm = new JFrame("Divider App");
      jfrm.setSize(275, 200);
      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); // to display error message
jfrm.add(jlab);
jfrm.add(ajtf)
;
jfrm.add(bjtf)
;
jfrm.add(butt
on);
jfrm.add(alab
);


jfrm.add(blab
);
jfrm.add(ansl
ab);


ActionListener l = new ActionListener() {
   public void actionPerformed(ActionEvent evt) {


   }
};
ajtf.addActionListene
r(l);
bjtf.addActionListene
r(l);
```

```
button.addActionListener(new ActionListener()
   { public void actionPerformed(ActionEvent
   evt) {
      try{
         int                  a                  =
         Integer.parseInt(ajtf.getText());  int
         b                                       =
         Integer.parseInt(bjtf.getText());   if
         (b == 0) {
            throw new ArithmeticException();
         }
         int ans = a/b;


         alab.setText("A = " + a);
         blab.setText("B = " + b);
         anslab.setText("Ans = "+ ans);
         err.setText(""); // Clear any previous error message
      }
      catch(NumberFormatException e){
         clearLabels();
         err.setText("Enter Only Integers!");
      }
      catch(ArithmeticException e){
         clearLabels();
         err.setText("B should be NON zero!");
      }
   }
```

```
        private void clearLabels() {
            alab.setText("");
            blab.setText("");
            anslab.setText("");

        }
    });

    jfrm.setVisible(true);
  }

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

WEEK-10

10.Demonstrate Inter process Communication and deadlock Program:

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); // get lock on a in this thread.
        System.out.println("Back in main thread");
```

```
  }

  public void run() {
    b.bar(a); // get lock on b in other thread.
    System.out.println("Back in other thread");
  }

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

InterprocessCommunication:

```
class Q {
int n;
boolean valueSet = false;
synchronized int get() {
while(!valueSet)
try {
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
System.out.println("Got: " + n);
valueSet = false;
notify();
return n;
```

```
}

synchronized void put(int n) {
while(valueSet)
try {
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + 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();
i++;
}
}
}

class PCFixed {
public static void main(String args[]) {
Q q = new Q();
new Producer(q);
new Consumer(q);
System.out.println("Press Control-C to stop.");
}
}
```