VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

ANUSHREE VK (1BM23CS046)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019

Sep-2024 to Jan-2025

B.M.S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Object Oriented Java Programming (23CS3PCOOJ)" carried out by **ANUSHREE VK (1BM23CS046),** who is a bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Object Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

Sheetal V A Assistant Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

Index

Github Link:
https://github.com/1BM23CS046/java

Program 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.

Algorithm:

Week-1

lab program-1

DATE: 09/10 PAGE:

1. develop a java program that prints all real solutions

```
import java.util.scanner;
import java.lang.Math;
class q{
  public static void main(String[]args){
    Scanner scanner=new Scanner(System.in);
    System.out.print("enter coefficient a:");
    double a=scanner.nextdouble();
    system.out.print("enter coefficient b:");
    double b=scanner.next double();
   system.out.print("enter coefficient c:");
   double c=scanner.nextdouble();
   double d=b*b-4*a*c;

if(d>0){
  double r1=(-b+math.sqrt(d))/(2*a);
  double r2=(-b-math.sqrt(d))/(2*a);
  System.out.println("the roots are real and different:");
  system.out.println("Root1:"+r1);
 System.out.println("Root2:"+r2); }
else if(d==0){
  double r=-b/(2*a);
  System.out.println("the roots are real and the same:"
  System.out.println("Root:"+r); }
else{
  system.out.println("the roots are complex:");
  double realpart=-b/(2*a);
```

```
double imaginaryPart = Math.sqrt(-d) / (2*a);
System.out.println("Root 1:" + realPart + "+" +
        imaginaryPart + "i");
System.out.println("Root 2:" + realPart + "-" + imaginary
                                        Part + "i");
    }
    scanner.close();
    System.out.println("Anekshree 1BM23CS046");
  }
}
```

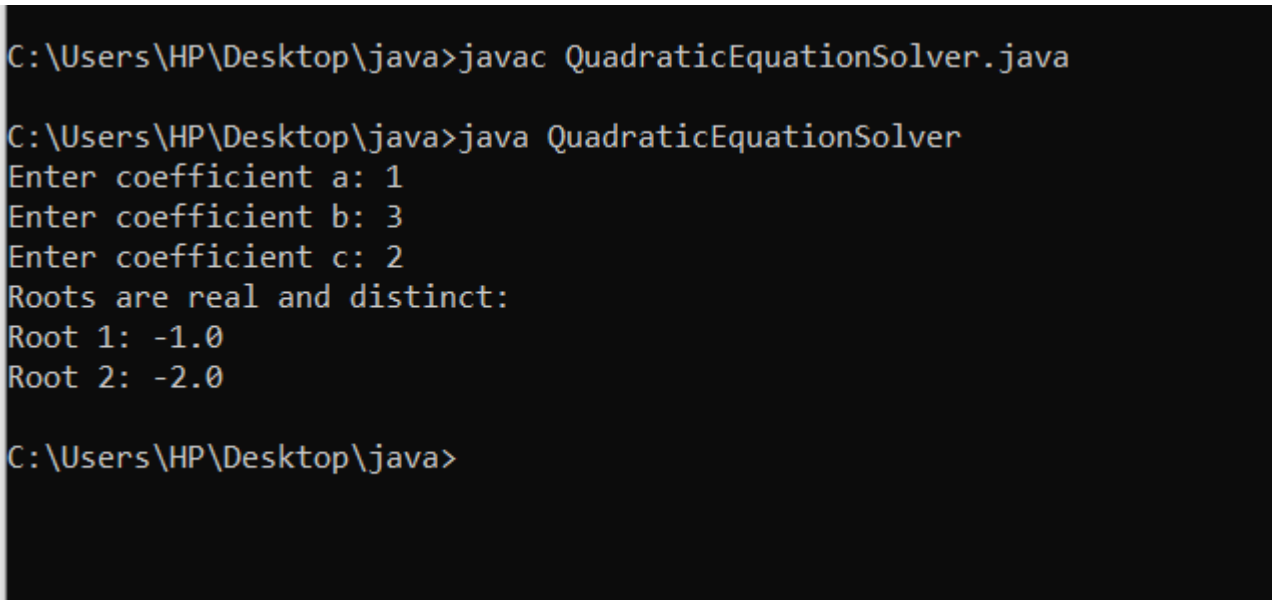
Output:

enter a: 1
enter b: -2
enter c: 1
the root is 1.0

code:

```
import java.util.Scanner;

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


      System.out.print("Enter coefficient a: ");
      double a = scanner.nextDouble();
      System.out.print("Enter coefficient b: ");
      double b = scanner.nextDouble();
      System.out.print("Enter coefficient c: ");
      double c = scanner.nextDouble();


      double discriminant = b * b - 4 * a * c;


      if (discriminant > 0) {

         double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
         double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
         System.out.println("Roots are real and distinct:");
         System.out.println("Root 1: " + root1);
         System.out.println("Root 2: " + root2);
      } else if (discriminant == 0) {

         double root = -b / (2 * a);
         System.out.println("Roots are real and equal:");
         System.out.println("Root: " + root);
      } else {

         double realPart = -b / (2 * a);
         double imaginaryPart = Math.sqrt(-discriminant) / (2 * a);
         System.out.println("Roots are complex:");
         System.out.println("Root 1: " + realPart + " + " + imaginaryPart + "i");
         System.out.println("Root 2: " + realPart + " - " + imaginaryPart + "i");
      }

      scanner.close();
   }
}
```

output:



program: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.

Week-2

DATE: 16/10 PAGE:

Calculate the SGPA of Student.

1) 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.

Prog:

```
Import java.util.Scanner;
Class student {
    String usn,name;
    int m[]=new int[3];
    int c[]=new int[3];
    int g.es.tc;
    double sgpa;
    Scanner S=new scanner(System.in);
Void details()
{
  System.out.println("enter usn:");
  usn=S.next();
  System.out.println("enter name:");
  name=S.next();
  System.out.println("enter marks:");
  for(int i=0;i<3;i++)
  {
     m[i]=S.nextint()
  }
System.out.println("enter credits for 3 subjects:");
for(int i=0;i<3;i++)
{
    c[i]=S.nextint();
}
}
```

```
Void display()
{
System.out.println("USN:"+" "+usn);
System.out.println("name:"+" "+name);
System.out.println("marks:");

for(int i=0;i<3;i++)
{
  System.out.println("marks "+" "+(i+1)+" :"+m[i]);
}
}

Void Sgpacalc()
{
System.out.println("USN:"+" "+usn);
for(int i=0;i<3;i++)
{
  g=(m[i]/10)+1;
  es+=g*c[i];
  tc+=c[i];
}
sgpa=(double)es/(double)tc;
System.out.println(sgpa);
}
}
class ti{
  public static void main(String[]args);
  {
    Student s[]=new student[3];
    for(int i=0;i<3;i++)
{
  s[i]=new student();
}
  for(int j=0;j<3;j++)
{
System.out.println("enter details of "+" "+(j+1)+s[j].details();
}
  for(int r=0;r<3;r++)
{
  System.out.println("details of "+" "+(r+1)+" student:");
   s[r].display();
}
  for(int i=0;i<3;i++)
{
  System.out.println("SGPA of "+" "+(i+1)+"student:");
   s[i].Sgpacalc();
}
}
}
```

Output :-

```
enter details of student 1:
enter usn:
1BM23CS001
enter name:
  Aman
enter marks for 3 subjects
80,90,100
enter credits for 3 students
 4 3 2.
```

enter the details of student 2:
enter name mili
enter usn 1BM23CS045
enter marks for 3 subject.
76
88
80
enter credits for 3 subject
4
3
2
enter the detail of student 3
enter name
Anumili
enter usn
1BM23CS046
enter marks for 3 subject
92
85
87
enter credits for 3 subject
4
3
2

Details of student 1:
Usn: 1BM23CS001
Name: Amal
Marks: Marks1: 80
marks2: 90
marks3: 100
sgpa of student 1: 84.5

Details of student 2:
Usn: 1BM23CS045
Name: Mili
Marks:
mark1: 76
mark2: 88
marks3: 80
sgpa of student 2: 80.

Details of student 3:
Usn: 1BM23CS046
Name: Anumili
Marks:
mark1: 92
marks2: 85
mark3: 87

sgpa of student 3: 88.

SGPA = $\frac{\sum \text{Marks} \times \text{Credit}}{\sum \text{Credit}}$ //

code:

```
import java.util.Scanner;

class Student {
   String usn;
   String name;
   int[] credits;
   int[] marks;
   int subjects;
void acceptDetails(Scanner scanner) {
       System.out.print("Enter USN: ");
       usn = scanner.nextLine();
       System.out.print("Enter Name: ");
       name = scanner.nextLine();
System.out.print("Enter the number of subjects: ");
       subjects = scanner.nextInt();

       credits = new int[subjects];
       marks = new int[subjects]
for (int i = 0; i < subjects; i++) {
           System.out.print("Enter credits for subject " + (i + 1) + ": ");
           credits[i] = scanner.nextInt();
           System.out.print("Enter marks for subject " + (i + 1) + ": ");
           marks[i] = scanner.nextInt();
       }
       scanner.nextLine(); // Clear buffer
   }
double calculateSGPA() {
       int totalCredits = 0;
       int weightedGradePoints = 0;

       for (int i = 0; i < subjects; i++) {
           int gradePoint = getGradePoint(marks[i]);
           weightedGradePoints += gradePoint * credits[i];
           totalCredits += credits[i];
       }

       return (double) weightedGradePoints / totalCredits;
   }


   int getGradePoint(int marks) {
       if (marks >= 90) return 10;
       if (marks >= 80) return 9;
```

```
        if (marks >= 70) return 8;
        if (marks >= 60) return 7;
        if (marks >= 50) return 6;
        return 0; // Fail
    }


    void displayDetails() {
        System.out.println("\nStudent Details:");
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);

        System.out.println("\nSubject-wise Details:");
        for (int i = 0; i < subjects; i++) {
            System.out.println("Subject " + (i + 1) + ": Credits = " + credits[i] + ", Marks = " +
marks[i]);
        }

        double sgpa = calculateSGPA();
        System.out.printf("\nSGPA: %.2f\n", sgpa);
    }
}

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


        Student student = new Student();

        System.out.println("Enter student details:");
        student.acceptDetails(scanner);

        System.out.println("\nCalculating SGPA...");
        student.displayDetails();

        scanner.close();
    }
}
```

```
C:\Users\HP\Desktop\java>java SGPA
Enter student details:
Enter USN: 46
Enter Name: anu
Enter the number of subjects: 3
Enter credits for subject 1: 4
Enter marks for subject 1: 4
Enter credits for subject 2: 3
Enter marks for subject 2: 89
Enter credits for subject 3: 3
Enter marks for subject 3: 99

Calculating SGPA...

Student Details:
USN: 46
Name: anu

Subject-wise Details:
Subject 1: Credits = 4, Marks = 4
Subject 2: Credits = 3, Marks = 89
Subject 3: Credits = 3, Marks = 99

SGPA: 5.70
```

program: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.

23/10/24

```
class Book{
    private string name;
    private string author;
    private double price;
    private int num_pages;

    Public book( string name ,string author ,double price ,int num_pages)
    {
        this.name = name;
        this.author=author;
        this.price =price;
        this.num_pages = num_pages;
}

    Public string toString(){
        return"Book name :" +name + "\n author :"\n price :$" +
        price +"\n pages :"+num_pages;
}

ic class main{
ublic static void main (String[] args){
    book book1 =new book("java basics", "john doe", 29.99, 350);
    book book2= new book("advanced java", "jane smith", 39.99, 500);
    System.out.println(book1);
    System.out.println();
    System.out.println(book2);
}
```

Output:

Book name : java basics
author : john doe
price : $29.99
Pages: 350

book name : advanced java

code:

```
import java.util.Scanner;

class Book {
    // Members of the Book class
    private String name;
    private String author;
    private double price;
    private int numPages;

    // Constructor to initialize the Book object
    public Book(String name, String author, double price, int numPages) {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }

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

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

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

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

    // Getter methods
    public String getName() {
        return name;
    }

    public String getAuthor() {
```

```
        return author;
    }

    public double getPrice() {
        return price;
    }

    public int getNumPages() {
        return numPages;
    }

    // toString method to display book details
    @Override
    public String toString() {
        return "Book Details:\n" +
               "Name: " + name + "\n" +
               "Author: " + author + "\n" +
               "Price: " + price + "\n" +
               "Number of Pages: " + numPages;
    }
}

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

        // Input the number of books
        System.out.print("Enter the number of books: ");
        int n = scanner.nextInt();
        scanner.nextLine(); // Clear the buffer

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

        // Input details for each book
        for (int i = 0; i < n; i++) {
            System.out.println("\nEnter details for Book " + (i + 1) + ":");
            System.out.print("Enter name: ");
            String name = scanner.nextLine();
            System.out.print("Enter author: ");
            String author = scanner.nextLine();
```

```
            System.out.print("Enter price: ");
            double price = scanner.nextDouble();
            System.out.print("Enter number of pages: ");
            int numPages = scanner.nextInt();
            scanner.nextLine(); // Clear the buffer

            // Create a new Book object and add it to the array
            books[i] = new Book(name, author, price, numPages);
        }

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

        scanner.close();
    }
}
```

```
C:\Users\HP\Desktop\java>java BookManager
Enter the number of books: 2434

Enter details for Book 1:
Enter name: anvi
Enter author: fdjhsg
Enter price: 500
Enter number of pages: 240

Enter details for Book 2:
Enter name: i hv rached sam
Enter author: hdgefkej
Enter price: 455
Enter number of pages: 223

Enter details for Book 3:
Enter name: iyt ends with us
Enter author: djhkjf
Enter price: 544
Enter number of pages: 232
```

program 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.Scanner;
abstract class Shape {
    int dimension1, dimension2;
    abstract void printArea();
}
class Rectangle extends Shape {
    Rectangle(int length, int width) {
        this.dimension1 = length;
        this.dimension2 = width;
    }
    void printArea() {
        int area = dimension1 * dimension2;
        System.out.println("Area of Rectangle: " + area);
    }
}
class Triangle extends Shape {
    Triangle(int base, int height) {
        this.dimension1 = base;
        this.dimension2 = height;
    }
    void printArea() {
        double area = 0.5 * dimension1 * dimension2;
        System.out.println("Area of Triangle: " + area);
    }
}
class Circle extends Shape {
    Circle(int radius) {
        this.dimension1 = radius;
    }
    void printArea() {
        double area = Math.PI * dimension1 * dimension1;
        System.out.println("Area of Circle: " + area);
    }
}

public class ShapeDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter length and width of the rectangle: ");
        int length = scanner.nextInt();
        int width = scanner.nextInt();
        Rectangle rectangle = new Rectangle(length, width);
        rectangle.printArea();
        System.out.print("Enter base and height of the triangle: ");
```

```
public class ShapeDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter length and width of the rectangle: ");
        int length = scanner.nextInt();
        int width = scanner.nextInt();
        Rectangle rectangle = new Rectangle(length, width);
        rectangle.printArea();
        System.out.print("Enter base and height of the triangle: ");
        int base = scanner.nextInt();
        int height = scanner.nextInt();
        Triangle triangle = new Triangle(base, height);
        triangle.printArea();
        System.out.print("Enter radius of the circle: ");
        int radius = scanner.nextInt();
        Circle circle = new Circle(radius);
        circle.printArea();

        scanner.close();
    }
}
```

```
Microsoft Windows [Version 10.0.19045.5131]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\Desktop\java>javac ShapeDemo.java

C:\Users\HP\Desktop\java>java ShapeDemo
Enter length and width of the rectangle: 22
33
Area of Rectangle: 726
Enter base and height of the triangle: 4
5
Area of Triangle: 10.0
Enter radius of the circle: 6
Area of Circle: 113.09733552923255

C:\Users\HP\Desktop\java>
```

program :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.

lab-5 Class Bank DATE: 13/11 PAGE:

1.
```
Import java.util.Scanner;
Class account {
    protected String customer name;
    protected int account number;
    protected double balance;
    public account ( String customer name, int account number,
                     double balance ){
        this.customer name = customer name;
        this.account number= account number;
        this.balance = balance;
    }
    public void deposit ( double amount ){
        if ( amount >0) {
            balance += amount;
            System.out.println("deposited :" + amount);
        }
        else{
            System.out.println ("Invalid deposit amount");
        }
    }
    public void displaybalance(){
        System.out.println("balance :" + balance);
    }
}
Class savaccount extends account{
    private double interest rate;
    public savaccount (String customer name, int account number, double
        balance, double interest rate ){
```

```
Super (customerName, accountNumber, balance);
    this.interestRate = interestRate;
}
public void computeAndDepositInterest() {
    double interest = balance * (interestRate / 100);
    balance += interest;
    System.out.println("Interest added: " + interest);
}
public void withdraw(double amount) {
    if (amount <= balance) {
        balance -= amount;
        System.out.println("Service charge imposed: " + serviceCharge);
    }
} else {
    System.out.println("Insufficient balance for withdrawal");
}
}
}

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

        SavAcct savAcc = new SavAcct("Alice", 12345, 1000.5);
        CurrAcct currAcc = new CurrAcct("Bob", 67890, 2000.50, 5000.50);

        System.out.println("Choose account type: \n1. Savings account \n2.
                            Current account");
```

```
        int choice = sc.nextInt();

        switch (choice) {
            case 1:
                System.out.println("Saving account Selected");
                savAcc.deposit(500);
                savAcc.computeAndDepositInterest();
                savAcc.withdraw(300);
                savAcc.displayBalance();
                break;

            case 2:
                System.out.println("Current account Selected");
                currAcc.deposit(500);
                currAcc.withdraw(1800);
                currAcc.displayBalance();
                break;

            default:
                System.out.println("Invalid choice");
        }
        sc.close();
    }
}
```

Output

1. Savings account
2. Current account
2.

Current account Selected
deposited: 500.0
withdraw: 1800.0
balance: 700.0

2.

```
import java.util.Scanner;
class account{
    string customername;
    int account number;
    double balance;
public account (string customername, int account no, double
                balance){
    this.customer name=customer name;
    this.account numb=account number;
    this.balance=balance;
}
public vod deposit (double amount){
    if (amount >0) balance+=amount;
}
public void displaybalance(){
    System.out.println("balance:"+balance);
}
}
class savact (string customer name, int account numbr, double
              balance, double interest rate){
super (customer name, account numbr, balance);
    this.interest rate = Interest rate;
}
public void computrand deposit interest(){
    balance += balance * (interest /100);
}
public void withdraw (double amount){
    if (amount <= balance) balance -= amount;
}
}
```

```
class curact extends account{
    private double minimumbalance, penalty charge;

public curact (string customername, int account no, double balance,
               double minimumbalance, double penalty charge){
    this.minimum balance = minimum balance;
    this.penalty charge = penalty charge;
}
    public void withdraw (double amount){
        if (amount <= balance){
            balance -= amount;
        if (balance < minimumbalance) balance -= penalty charge;
    }
}
}
public class bank{
    public static void main (string [] args){
        Scanner sc=new Scanner (system.in);
        System.out.print("enter name:");
        String name = sc.nextline();
        int acc numbr = sc.nextint();
        double balance = sc.nextdouble();

account = null;
while (true){
    System.out.println("1.savings 2.current");
    int type = sc.nextint();
    if (type == 1)
    {
```

```
}
else if (type == 2){
    System.out.print("enter minimum balance:");
    double minbalance = sc.nextDouble();
    System.out.print("enter penalty charge:");
    double penaltyCharge = sc.nextDouble();
    account = new current(name, acc no, balance, min
        balance, penalty(charge);

}
else {
    s.o.pl("invalid Choice");
}
}
while(true){
    int choice = sc.nextInt();

if (choice == 1){
    System.out.println("enter deposit amount:");
    amount.deposit(sc.nextDouble());
}
}

    sc.close();
}
}
```

```
class Bank {
    abstract static class Account {
        String accountHolder;
        double balance;

        Account(String accountHolder, double balance) {
            this.accountHolder = accountHolder;
            this.balance = balance;
        }

        abstract void withdraw(double amount);

        abstract void displayAccountInfo();
    }

    static class SavingsAccount extends Account {
        double interestRate;

        SavingsAccount(String accountHolder, double balance, double interestRate) {
            super(accountHolder, balance);
            this.interestRate = interestRate;
        }


        void withdraw(double amount) {
            if (amount <= balance) {
                balance -= amount;
                System.out.println("Withdrawal successful. New balance: " + balance);
            } else {
                System.out.println("Insufficient funds for withdrawal.");
            }
        }

        void addInterest() {
            balance += balance * interestRate / 100;
            System.out.println("Interest added. New balance: " + balance);
        }


        void displayAccountInfo() {
            System.out.println("Savings Account Holder: " + accountHolder);
            System.out.println("Balance: " + balance);
            System.out.println("Interest Rate: " + interestRate + "%");
        }
```

```
    static class CurrentAccount extends Account {
        double minimumBalance;
        double serviceCharge;

        CurrentAccount(String accountHolder, double balance, double minimumBalance, double serviceCharge) {
            super(accountHolder, balance);
            this.minimumBalance = minimumBalance;
            this.serviceCharge = serviceCharge;
        }

        @Override
        void withdraw(double amount) {
            if (balance - amount >= minimumBalance) {
                balance -= amount;
                System.out.println("Withdrawal successful. New balance: " + balance);
            } else {
                System.out.println("Withdrawal denied. Balance would fall below minimum required balance.");
            }
        }

        void imposeServiceCharge() {
            if (balance < minimumBalance) {
                balance -= serviceCharge;
                System.out.println("Service charge applied. New balance: " + balance);
            }
        }

        @Override
        void displayAccountInfo() {
            System.out.println("Current Account Holder: " + accountHolder);
            System.out.println("Balance: " + balance);
            System.out.println("Minimum Balance: " + minimumBalance);
            System.out.println("Service Charge: " + serviceCharge);
        }
    }

    public static void main(String[] args) {
        SavingsAccount savings = new SavingsAccount("John Doe", 1000, 5);
        CurrentAccount current = new CurrentAccount("Jane Smith", 1500, 500, 20);
savings.displayAccountInfo();
        current.displayAccountInfo();
        savings.withdraw(200);
        savings.addInterest();
```

```
C:\Users\HP\Desktop\java>javac Bank.java

C:\Users\HP\Desktop\java>java Bank
Savings Account Holder: John Doe
Balance: 1000.0
Interest Rate: 5.0%
Current Account Holder: Jane Smith
Balance: 1500.0
Minimum Balance: 500.0
Service Charge: 20.0
Withdrawal successful. New balance: 800.0
Interest added. New balance: 840.0
Withdrawal denied. Balance would fall below minimum required balance.
Savings Account Holder: John Doe
Balance: 840.0
Interest Rate: 5.0%
Current Account Holder: Jane Smith
Balance: 1500.0
Minimum Balance: 500.0
Service Charge: 20.0

C:\Users\HP\Desktop\java>
```

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.

```
class Account {
    String customerName;
    String accountNumber;
    String accountType;
    double balance;

    public Account(String customerName, String accountNumber, String accountType, double initialBalance) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = initialBalance;
    }
public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposit successful. Updated balance: " + balance);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }
    public void displayBalance() {
        System.out.println("Account Holder: " + customerName);
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Account Type: " + accountType);
        System.out.println("Current Balance: " + balance);
    }
}

class SavAcct extends Account {
    double interestRate;

    public SavAcct(String customerName, String accountNumber, double initialBalance, double interestRate) {
        super(customerName, accountNumber, "Savings", initialBalance);
        this.interestRate = interestRate;
    }
    public void addInterest() {
        double interest = balance * (interestRate / 100);
        balance += interest;
        System.out.println("Interest added: " + interest);
        System.out.println("Updated balance after adding interest: " + balance);
    }
```

```
    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawal successful. Updated balance: " + balance);
        } else {
            System.out.println("Insufficient balance for withdrawal.");
        }
    }
}
class CurAcct extends Account {
    double minimumBalance;
    double penalty;

    public CurAcct(String customerName, String accountNumber, double initialBalance, double minimumBalance, double penalty) {
        super(customerName, accountNumber, "Current", initialBalance);
        this.minimumBalance = minimumBalance;
        this.penalty = penalty;
    }
    public void withdraw(double amount) {
        if (balance - amount >= minimumBalance) {
            balance -= amount;
            System.out.println("Withdrawal successful. Updated balance: " + balance);
        } else {
            System.out.println("Balance would fall below minimum. Penalty imposed.");
            balance -= penalty;
            System.out.println("Penalty of " + penalty + " applied. Updated balance: " + balance);
        }
    }
}

public class BankAccount {
    public static void main(String[] args) {
        SavAcct savings = new SavAcct("John Doe", "SA12345", 2000, 5);
        savings.displayBalance();
        savings.deposit(500);
        savings.addInterest();
        savings.withdraw(1000);

        CurAcct current = new CurAcct("Jane Smith", "CA67890", 3000, 1000, 50);
        current.displayBalance();
        current.deposit(700);
        current.withdraw(2500);
        current.withdraw(500); // Causes penalty
```

```
C:\Users\HP\Desktop\java>javac BankAccount.java

C:\Users\HP\Desktop\java>java BankAccount
Account Holder: John Doe
Account Number: SA12345
Account Type: Savings
Current Balance: 2000.0
Deposit successful. Updated balance: 2500.0
Interest added: 125.0
Updated balance after adding interest: 2625.0
Withdrawal successful. Updated balance: 1625.0
Account Holder: Jane Smith
Account Number: CA67890
Account Type: Current
Current Balance: 3000.0
Deposit successful. Updated balance: 3700.0
Withdrawal successful. Updated balance: 1200.0
Balance would fall below minimum. Penalty imposed.
Penalty of 50.0 applied. Updated balance: 1150.0

C:\Users\HP\Desktop\java>
```

program 6

Create a package CIE which has two classes - Personal 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 Personal. 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.

lab-8 CIE & SEE

DATE: 13/11 PAGE:

1. 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 that of the student. Create another package SEE which has the class External which is derived class of student. this class has an array that stores the SEE marks scored in five courses of the current sem of the stud. Import the 2 package in the file that declare the final marks

Prog:-

```
package com.CIE;
public class Student {
    public string usn;
    public string name;
    public int sem;
public Student (String usn, String name, int sem)
{
    this.usn=usn;
    this.name=name;
    this.sem=sem;
}

public void display details() {
    System.out.println("usn:"+usn);
    System.out.println("name:"+name);
    System.out.println("Sem:"+sem);
}

public class internal {
    public int[] internal marks;
    public if (marks.length !=5) {
        System.out.println("enter 5 marks!");
    }
```

```
        this.Internal marks=marks;
    }
    public void display_Imarks(){
        System.out.println("Internal marks:");
        for(int i=0;i<5;i++)
        {
            System.out.print(marks+" ");
        }
        System.out.println();
    }
}

package com.SEE;
    import com.CIE.Student;
    public class external extends Student{
        public int[] external marks;
    public external(String name, String usn, int sem, int[]
                    marks){
        super(name, usn, sem);
        if(marks.length!=5){
            System.out.println("enter 5 subjects!")
            this.external marks=marks;
    }
public void display_Emarks(){
    System.out.println("SEE marks:");
    for(int i=0;i<5;i++)
        S.o.p.(marks+" ");
    System.out.println()
}
```

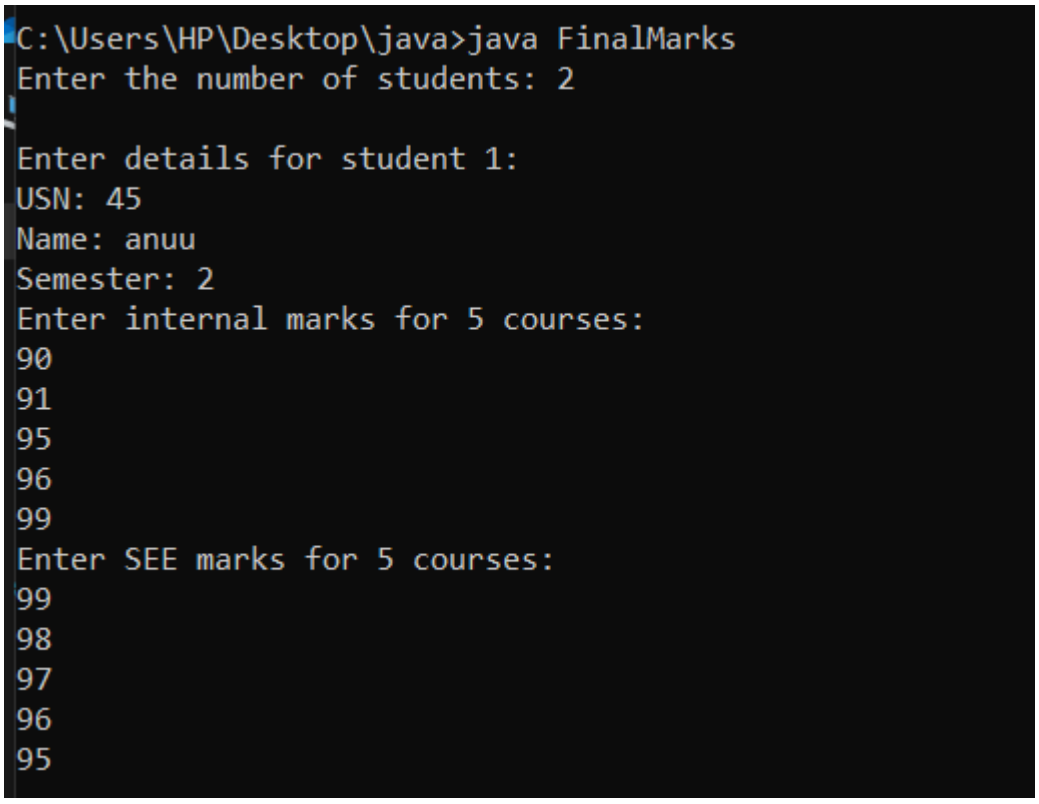
```
import com.CIE.*;
import com.SEE.*;
import java.util.Scanner;
public class main {
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
    System.out.println("enter no of students:");
        int n = Scanner.nextInt();

external[] students = new external[n];
Internal[] int marks = new Internal[n];
for(int i=0;i<n;i++){
    System.out.println("enter usn:");
        String usn=sc.nextLine();
    System.out.println("entername:");
        String name=sc.nextLine();
    Sy.out.println("enter sem:");
        int sem = sc.nextInt();

System.out.println("enter usn:");
    String[] marks=new int[5];
for(int j=0;j<5;j++){
    Imarks[j]=sc.nextInt();
}
    System.out.println("enter external marks:");
        int emarks=new int[5];
    for(int k=0;k<5;k++){
        Emarks[k]=sc.nextInt();
    }.
```

DATE: PAGE:

```
int marks[i]=new Internals (imarks);
    students[i]=new external (name, usn, sem, emarks);
}
    sy.out.println("final marks of student:")
        for(int i=0;i<n;i++){
         students[i].display details();
        int marks[i].display marks();
         student[i].display marks[i];

System.out.println("final marks");
    for(j=0;j<5;j++)
{
    int final=int marks[i].imarks[j]+(students[i].emarks[j]);
        System.out.println(final+"\n");
}
    system.out.println("\n");
}
}
```

```
import java.util.Scanner;

class Personal {
    String usn;
    String name;
    int sem;

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

class Internals {
    int[] internalMarks = new int[5];

    public Internals(int[] marks) {
        System.arraycopy(marks, 0, internalMarks, 0, 5);
    }
}

class External extends Personal {
    int[] externalMarks = new int[5];

    public External(String usn, String name, int sem, int[] marks) {
        super(usn, name, sem);
        System.arraycopy(marks, 0, externalMarks, 0, 5);
    }
}

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

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

        External[] students = new External[n];
        Internals[] internalMarks = new Internals[n];

        for (int i = 0; i < n; i++) {
            System.out.println("\nEnter details for student " + (i + 1) + ":");
            System.out.print("USN: ");
```

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

    External[] students = new External[n];
    Internals[] internalMarks = new Internals[n];

    for (int i = 0; i < n; i++) {
        System.out.println("\nEnter 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();

        int[] internal = new int[5];
        System.out.println("Enter internal marks for 5 courses:");
        for (int j = 0; j < 5; j++) {
            internal[j] = scanner.nextInt();
        }

        int[] external = new int[5];
        System.out.println("Enter SEE marks for 5 courses:");
        for (int j = 0; j < 5; j++) {
            external[j] = scanner.nextInt();
        }

        internalMarks[i] = new Internals(internal);
        students[i] = new External(usn, name, sem, external);
    }

    System.out.println("\nFinal Marks of Students:");
    for (int i = 0; i < n; i++) {
        System.out.println("Student " + (i + 1) + " (" + students[i].usn + " - " + students[i].name + "):");
        for (int j = 0; j < 5; j++) {
            int finalMark = internalMarks[i].internalMarks[j] + (students[i].externalMarks[j] / 2);
            System.out.println("Course " + (j + 1) + ": " + finalMark);
        }
    }

    scanner.close();
}
```



program :7

implement exception handling
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 uses both father and son’s age and throws an exception if son’s age is >=father’s age.

lab-7

DATE: 20/11 PAGE:

* WAP that demonstrates handling of exceptions in Inheritance String. Create a base class called as "Father" and derive class called as "Son" which extends the base class. in Fathers class implement a constructor which takes age and throw the exceptions wrong age when the input age is less than 0 (age<0) in son's class implement a const. the father and son's age and throws an exceptions if Sun's age ≥ fathers age.

Prog:-

```
Import java.util.Scanner;
class wrong age exception extends exception{
  public wrong age exception(String message){
    super(message);
  }
}
class father{
  private int age;
  public father(int age) throws wrong age exception{
    if(age<0){
      throw new wrong age exception("wrong age");
    }
    this.age=age;
  }
  public int getage(){
    return age;
  }
}
class son extends father{
  private int sonage;
```

```
public Son(int fatherage, int sonage) throws
  wrongage exception, Sonage exception {
      super(fatherage);
      if (sonage >= fatherage) {
        throw new sonageexception("son's age
  cannot be greater than or equal to father's age");
  }
    this.sonage = sonage;
}
  public int getsonage(){
    return sonage;
  }
}

public class FatherSon{
  public static void main(String[] args) {
    while(true){
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter father's age:");
    int fatherage=sc.nextint();
    System.out.print("enter son's age:");
    int sonage=sc.nextint();
    try{
      Son son = new Son(fatherage, sonage);
      System.out.println("accepted successfully");
  }
    catch (wrong age exception e){
      System.out.println(e.getmessage());
}
```

```
catch (Sonage exception e) {
  System.out.println(e.getmessage());
}

System.out.println("would you like to re-enter details[y/n]");
  String input = sc.next();
  if (input.equalsIgnoreCase("n")) {
    break;
  }
}
}
}
```

Output:

enter father's age: 45
enter son's age: 5
accepted successfully

Scan
Gl
20/11/24

```
class WrongAgeException extends Exception {
    public WrongAgeException(String message) {
        super(message);
    }
}

class Father {
    int age;

    public Father(int age) throws WrongAgeException {
        if (age < 0) {
            throw new WrongAgeException("Father's age cannot be negative.");
        }
        this.age = age;
    }
}

class Son extends Father {
    int sonAge;

    public Son(int fatherAge, int sonAge) throws WrongAgeException {
        super(fatherAge); // Calls the constructor of Father
        if (sonAge < 0) {
            throw new WrongAgeException("Son's age cannot be negative.");
        }
        if (sonAge >= fatherAge) {
            throw new WrongAgeException("Son's age cannot be greater than or equal to Father's age.");
        }
        this.sonAge = sonAge;
    }
}

public class InheritanceExceptionDemo {
    public static void main(String[] args) {
        try {
            System.out.println("Creating Father with age 40...");
            Father father = new Father(40);

            System.out.println("Creating Son with age 15...");
            Son son = new Son(40, 15);

            System.out.println("Father's age: " + father.age);
            System.out.println("Son's age: " + son.sonAge);
```

```
C:\Users\HP\Desktop\java>javac InheritanceExceptionDemo.java

C:\Users\HP\Desktop\java>java InheritanceExceptionDemo
Creating Father with age 40...
Creating Son with age 15...
Father's age: 40
Son's age: 15

Trying invalid ages...
Exception: Son's age cannot be greater than or equal to Father's age.

C:\Users\HP\Desktop\java>
```

program :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.

Lab-8 DATE: 27/11 PAGE:

1. WJP to create two threads, one ~~displaying~~ thread displaying "BMS college of engineering" once every ten seconds and another displaying "CSE" once every two seconds.

prg:

```
public class threadexample{
    static class printBms implements runnable{
        public void run(){
            while(true){
                System.out.println("Bms college of engineering");
    try{
        thread.sleep(10000);
    }
    catch (Interrupted exception e){
        thread.currentthread().Interrupt();
    }
    }
}
}
static class printCSE Implements runnable{
    public void run(){
        while(true){
            System.out.println("CSE");
    try{
        thread.sleep(2000);
    }
    catch (Interruptedexception e){
        thread.currentthread().Interrupt();
    }
}
}
```

DATE: PAGE:

```
public static void main (String[] args){
    thread thread1 = new thread (new printBms());
    thread1.start();

    thread thread2 = new thread (new printCSE());
    thread2.start();
}
}
```

output:

```
Bms college of engineering
CSE
CSE
CSE
CSE
Bms college of engineering
CSE
CSE
...
...
```

```
class CollegeThread extends Thread {

    public void run() {
        try {
            while (true) {
                System.out.println("BMS College of Engineering");
                Thread.sleep(10000); // Pause for 10 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("CollegeThread interrupted.");
        }
    }
}

class CSEThread extends Thread {

    public void run() {
        try {
            while (true) {
                System.out.println("CSE");
                Thread.sleep(2000); // Pause for 2 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("CSEThread interrupted.");
        }
    }
}

public class ThreadDisplayDemo {
    public static void main(String[] args) {
        CollegeThread collegeThread = new CollegeThread();
        CSEThread cseThread = new CSEThread();

        collegeThread.start(); // Start thread for "BMS College of Engineering"
        cseThread.start();     // Start thread for "CSE"
    }
}
```

```
C:\Users\HP\Desktop\java>javac ThreadDisplayDemo.java

C:\Users\HP\Desktop\java>java ThreadDisplayDemo
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
```

program: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.

Lab-9

DATE: 27/11 PAGE:

Q. WAP 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 no. format exception. If Num2 were zero, the prog would throw an arithmetic exception display the exception in a message dialog box.

Prog:

```
Import.
class swingdemo{
  Swingdemo(){
  JFrame jfrm=new JFrame("divides 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();
```

DATE: PAGE:

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

Output:

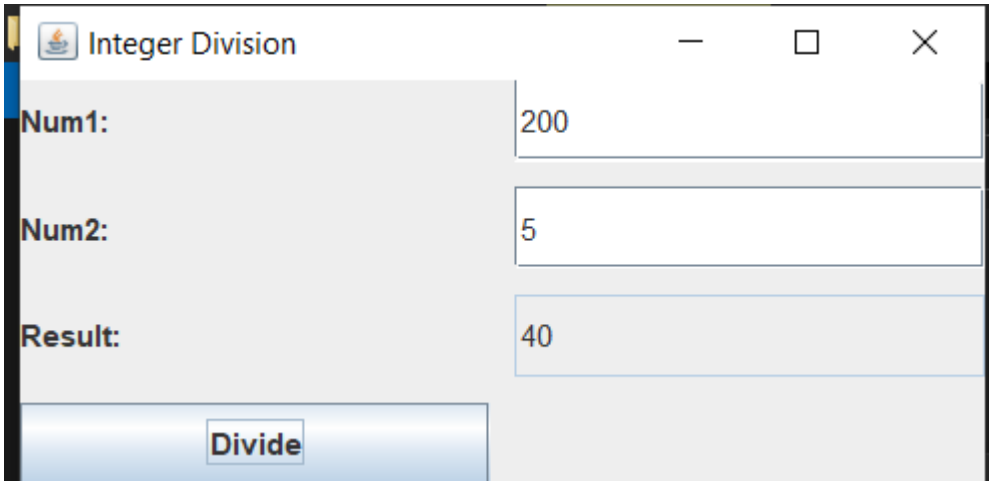
enter the divider and dividend:

[2435] [6]

[calculate] A=2435 B=6

Ans=405

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class IntegerDivisionUI {
    public static void main(String[] args) {
      JFrame frame = new JFrame("Integer Division");
        frame.setSize(400, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new GridLayout(4, 2, 10, 10));
JLabel num1Label = new JLabel("Num1:");
        JTextField num1Field = new JTextField();
        JLabel num2Label = new JLabel("Num2:");
        JTextField num2Field = new JTextField();
        JLabel resultLabel = new JLabel("Result:");
        JTextField resultField = new JTextField();
        resultField.setEditable(false);
        JButton divideButton = new JButton("Divide");
   frame.add(num1Label);
        frame.add(num1Field);
        frame.add(num2Label);
        frame.add(num2Field);
        frame.add(resultLabel);
        frame.add(resultField);
        frame.add(divideButton);
divideButton.addActionListener(new ActionListener() {

          public void actionPerformed(ActionEvent e) {
             try {

                int num1 = Integer.parseInt(num1Field.getText());
                int num2 = Integer.parseInt(num2Field.getText());


                int result = num1 / num2;
                resultField.setText(String.valueOf(result));
             } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(frame, "Please enter valid integers.", "Number Format Error", JOptionPane.ERROR_MESSAGE);
             } catch (ArithmeticException ex) {
                JOptionPane.showMessageDialog(frame, "Division by zero is not allowed.", "Arithmetic Error", JOptionPane.ERROR_MESSAGE);
             }
          }
       });
```



program 10

Demonstrate Inter process Communication and deadlock

DATE: PAGE:

Inter process communication

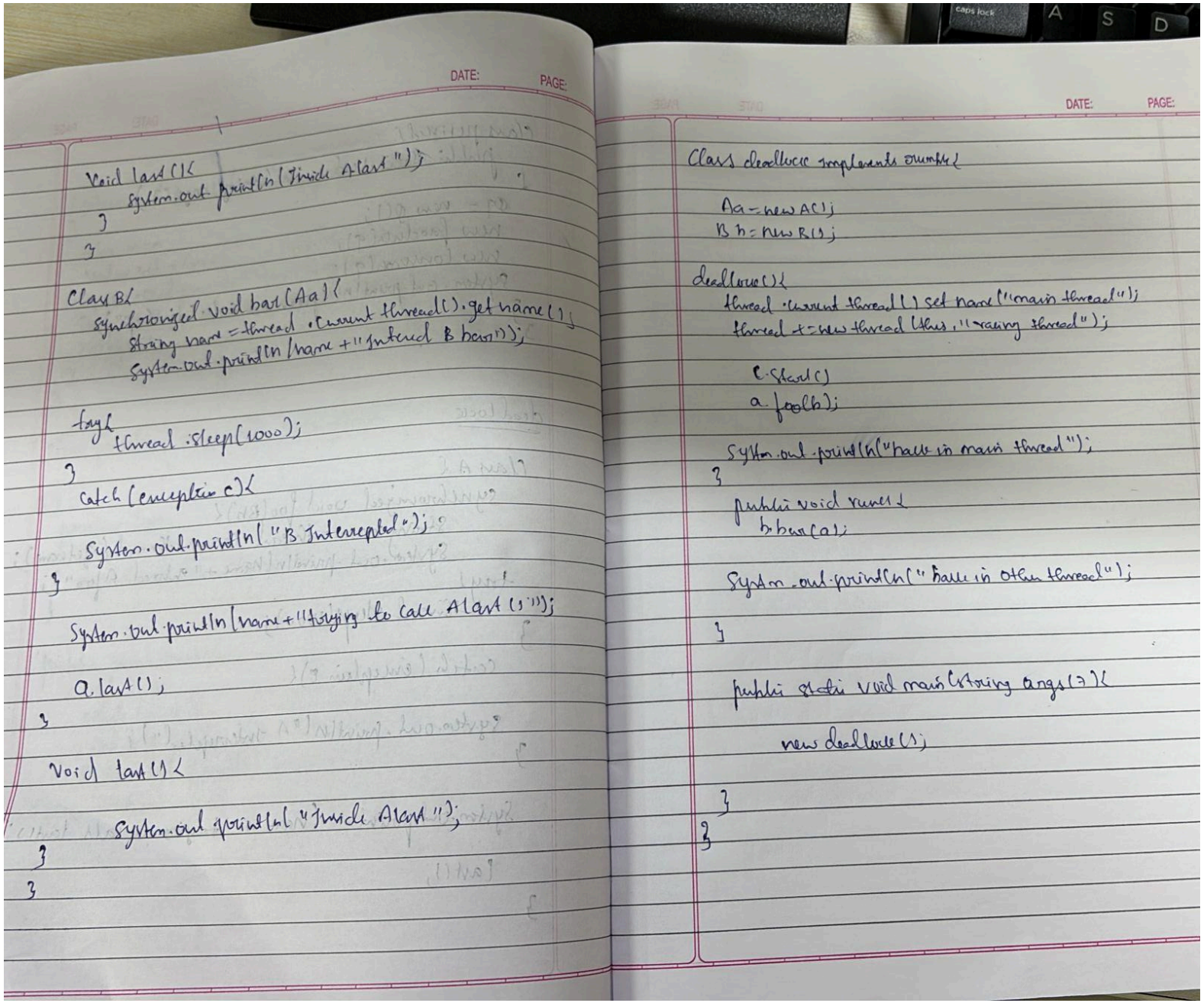
```
class Q {
  int n;
  boolean valueset = false;
  synchronized int get() {
    while (!valueset)
    try {
      System.out.println("\n consumer waiting\n");
      wait();
    }
    catch (InterruptedException e) {
    System.out.println("\n Interrupted exception caught");
    }
    System.out.println("got : " 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\g");
      wait();
  }
```

```
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){
  int r=q.get();
  System.out.println("consumed:"+r);
  i++;
  }
}
}
```

```
class prsimed{
  public static void main(String args[]);
{
  Q q=new Q();
  new producer(q);
  new consumer(q);
  System.out.println("press control+c to stop");
}
}
```

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 last()");

last();
}
```



```
Void last(){
    System.out.println("Inside Alast");
}
}

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 Alast()");
a.last();
}
Void last(){
    System.out.println("Inside Alast");
}
}
```

```
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);

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();

}
}
```

```
class SharedResource {
    // Shared resource for demonstration
    synchronized void sendMessage(String message) {
        System.out.println(Thread.currentThread().getName() + ": Sending message: " + message);
        try {
            Thread.sleep(1000); // Simulate some delay
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted.");
        }
    }
}

class ProcessA extends Thread {
    private final SharedResource resource1;
    private final SharedResource resource2;

    public ProcessA(SharedResource resource1, SharedResource resource2) {
        this.resource1 = resource1;
        this.resource2 = resource2;
    }

    @Override
    public void run() {
        synchronized (resource1) {
            System.out.println(Thread.currentThread().getName() + ": Locked resource1");
            try {
                Thread.sleep(1000); // Simulate delay
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            synchronized (resource2) {
                System.out.println(Thread.currentThread().getName() + ": Locked resource2");
                resource2.sendMessage("Hello from ProcessA");
            }
        }
    }
}
```

```
class ProcessB extends Thread {
    private final SharedResource resource1;
    private final SharedResource resource2;

    public ProcessB(SharedResource resource1, SharedResource resource2) {
        this.resource1 = resource1;
        this.resource2 = resource2;
    }

    @Override
    public void run() {
        synchronized (resource2) {
            System.out.println(Thread.currentThread().getName() + ": Locked resource2");
            try {
                Thread.sleep(1000); // Simulate delay
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            synchronized (resource1) {
                System.out.println(Thread.currentThread().getName() + ": Locked resource1");
                resource1.sendMessage("Hello from ProcessB");
            }
        }
    }
}

public class IPCAndDeadlockDemo {
    public static void main(String[] args) {
        SharedResource resource1 = new SharedResource();
        SharedResource resource2 = new SharedResource();

        // Create and start threads simulating processes
        ProcessA processA = new ProcessA(resource1, resource2);
        ProcessB processB = new ProcessB(resource1, resource2);

        processA.setName("ProcessA");
        processB.setName("ProcessB");

        processA.start();
        processB.start();
    }
}
```

```
D:\Anu\Engineering\java>java PCFixed
Press Control-C to stop.
Put: 0

Intimate Consumer


Producer waiting

Got: 0

Intimate Producer

Put: 1

Intimate Consumer

consumed:0

Producer waiting

Got: 1

Intimate Producer

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
Put: 4

Intimate Consumer


Producer waiting

Got: 4

Intimate Producer

consumed:4
Put: 5
```

```
Intimate Consumer


Producer waiting

Got: 5

Intimate Producer

consumed:5
Put: 6

Intimate Consumer


Producer waiting

Got: 6

Intimate Producer

Put: 7

Intimate Consumer


Producer waiting

consumed:6
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

Got: 10

Intimate Producer

consumed:10
Put: 11

Intimate Consumer


Producer waiting

Got: 11

Intimate Producer

consumed:11
Put: 12

Intimate Consumer


Producer waiting

Got: 12

Intimate Producer

consumed:12
Put: 13

Intimate Consumer


Producer waiting

Got: 13

Intimate Producer

consumed:13
Put: 14

Intimate Consumer

Got: 14

Intimate Producer

consumed:14
```

```
D:\Anu\Engineering\java>javac deadlock.java

D:\Anu\Engineering\java>java Deadlock
MainThread entered A.foo
RacingThread entered B.bar
RacingThread trying to call A.last()
MainThread trying to call B.last()
Inside A.last
Back in main thread
Inside A.last
Back in other thread
```