VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SINCHANA(1BM23CS329)

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-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**" carried out by **SINCHANA(1BM23CS329),** who is 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 during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of **Object-Oriented Java Programming Lab - (23CS3PCOOJ)** work prescribed for the said degree.

Dr. Nandhini Vineeth

Associate Professor,
Department of CSE,
BMSCE, Bengaluru

Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

INDEX

PROGARM 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.

```
import java.util.Scanner;

public class QuadraticSolver {
    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) {
            System.out.println("There are no real solutions.");
        } else {

            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);

            if (discriminant == 0) {
                System.out.println("There is one real solution: " +
root1);
            } else {
                System.out.println("The real solutions are: ");
                System.out.println("Root 1: " + root1);
                System.out.println("Root 2: " + root2);
            }
        }


        scanner.close();
    }
}
```

OUTPUT

```
Enter coefficient a: 1
Enter coefficient b: -3
Enter coefficient c: 2
The real solutions are:
Root 1: 2.0
Root 2: 1.0
```

```
Enter coefficient a: 7
Enter coefficient b: 8
Enter coefficient c: 5
There are no real solutions.
```

```
Enter coefficient a: 2
Enter coefficient b: 4
Enter coefficient c: 2
There is one real solution: -1.0
```

NOTES:

PAGE NO :
DATE :

1. Develop a java program that prints all real solutions to the quadratic equation $ax^2+bx+c=0$. Read in a, b, c and use the quadratic formula. If the discriminate b^2-4ac is negative, display a message stating that

->
```
import java.util.Scanner;
public class QuadraticEquation{
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("The equation has two
real solutions");
System.out.println("Root1 "+root1);
System.out.println("Root2 "+root2);
}
else if (discriminant==0){
double root = -b/(2*a);
system.out.println("The equation has one
real Solution");
System.out.println("Root "+Root);
```

PAGE NO :
DATE :

```
}
else
{
System.out.println("There are no real solutions");
}
scanner.close();
}
}
```

2. output

1
-3
2
2.0 and 1.0

PROGARM 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.

CODE:

```
import java.util.Scanner;

class Student {

    String usn;
    String name;
    int[] credits;
    int[] marks;

    public void acceptDetails(int numberOfSubjects) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter USN: ");
        usn = scanner.nextLine();

        System.out.print("Enter Name: ");
        name = scanner.nextLine();

        credits = new int[numberOfSubjects];
        marks = new int[numberOfSubjects];


        for (int i = 0; i < numberOfSubjects; 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();
        }
    }

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

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


    public double calculateSGPA() {
        int totalCredits = 0;
        double weightedMarks = 0;

        for (int i = 0; i < credits.length; i++) {
            totalCredits += credits[i];
            weightedMarks += credits[i] * (marks[i] / 10.0); // Assuming
marks are out of 100
        }


        return weightedMarks / totalCredits;
    }
}

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

```
        Scanner scanner = new Scanner(System.in);


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

        Student student = new Student();


        student.acceptDetails(numberOfSubjects);

        student.displayDetails();


        double sgpa = student.calculateSGPA();
        System.out.println("\nThe SGPA of " + student.name + " is: " +
sgpa);

        scanner.close();
    }
}
```

OUTPUT:

```
Enter the number of subjects: 4
Enter USN: 1BM23CS329
Enter Name: SINCHANA
Enter credits for subject 1: 4
Enter marks for subject 1: 99
Enter credits for subject 2: 3
Enter marks for subject 2: 98
Enter credits for subject 3: 2
Enter marks for subject 3: 97
Enter credits for subject 4: 3
Enter marks for subject 4: 99

Student Details:
USN: 1BM23CS329
Name: SINCHANA
```

```
Credits and Marks:
Subject 1 - Credits: 4, Marks: 99
Subject 2 - Credits: 3, Marks: 98
Subject 3 - Credits: 2, Marks: 97
Subject 4 - Credits: 3, Marks: 99

The SGPA of SINCHANA is: 9.841666666666667
```

NOTES:

PAGE NO :
DATE :

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

```
import java.util.Scanner;

class Student{
    String usn;
    String name;
    int[] credits;
    int[] marks;

void acceptdetails(int numsubjects){
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter USN");
    usn = sc.nextLine();
    System.out.print("Enter name");
    name = sc.nextLine();

    credits = new int[numsubjects];
    marks = new int[numsubjects];

for(int i=0; i<numsubjects; i++){
System.out.print("Enter credits for subject"
+(i+1)+":");
credits[i] = sc.nextInt();
System.out.print("Enter marks for subject"
+(i+1)+":");
```

PAGE NO :
DATE :

```
void display_details(){
System.out.println("USN" + usn);
System.out.println("Name" + name);
System.out.println("Credits and marks");
for(int i=0; i<credits.length; i++){
System.out.println("subject" + (i+1) + "(cred
+ (credits[i] + "marks" + marks[i]);
}
}

double calculateSGPA(){
  int totalcredits=0;
  double weightedsum=0.0;
for(int i=0; i<credits.length; i++){
  int gradepoint = getgradepoint(marks[i
weighted sum+ = (credits[i] * grade point;
  totalcredits+ = credits[i];
}
return weighted sum/totalcredits;
}

int getgradepoint(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;
```

PAGE NO :
DATE :

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

System.out.print ("Enter number of subjects");

int numsubjects = s.nextInt ();

Student st = new Student();

    st.acceptdetails (numsubjects);
    st.display details ();
    System.out.println ("SGPA" + st.calculateSGPA());
}
}
```

Output

```
Enter USN:
1BM23CS329
Enter name
Sinchana
Enter number of subjects:
3
Enter credits and marks for each subject

Credits : 4
marks: 96
credits : 4
```

PAGE NO :
DATE :

Student details
USN: 1BM23CS329
name: Sinchana
credits and marks:

Subject 1: credits = 4, marks: 96
Subject 2: credits = 4, marks: 96
Subject 3: credits = 3, marks = 91

9.6

24/10/24

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.

CODE:

```
import java.util.Scanner;

class Book {

    private String name;
    private String author;
    private double price;
    private int numPages;

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


    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;
    }

    public String getName() {
        return name;
    }

    public String getAuthor() {
        return author;
    }

    public double getPrice() {
        return price;
    }

    public int getNumPages() {
        return numPages;
    }

    @Override
    public String toString() {
        return "Book Name: " + name + "\nAuthor: " + author + "\nPrice: "
+ price + "\nNumber of Pages: " + numPages;
    }
}

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

```
        System.out.print("Enter the number of books you want to create:
");
        n = scanner.nextInt();
        scanner.nextLine(); // Consume newline character

        Book[] books = new Book[n];  // Array to hold multiple Book
objects

        int count = 0;  // Counter for the books created

        while (true) {
            System.out.println("\n--- Book Store Menu ---");
            System.out.println("1. Create a Book");
            System.out.println("2. Set Book Details");
            System.out.println("3. Get Book Details");
            System.out.println("4. Display All Books");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline character

            switch (choice) {
                case 1:
                    if (count < n) {
                        // Create a new book object with user input
                        System.out.print("Enter Book Name: ");
                        String name = scanner.nextLine();
                        System.out.print("Enter Author Name: ");
                        String author = scanner.nextLine();
                        System.out.print("Enter Book Price: ");
                        double price = scanner.nextDouble();
                        System.out.print("Enter Number of Pages: ");
                        int numPages = scanner.nextInt();
                        scanner.nextLine(); // Consume newline character

                        // Create the Book object and store it in the
array
                        books[count] = new Book(name, author, price,
numPages);
```

```
                        count++; // Increment the counter after creating
the book
                        System.out.println("Book created successfully!");
                    } else {
                        System.out.println("You have already created the
maximum number of books.");
                    }
                    break;

                case 2:
                    // Set details for a specific book
                    System.out.print("Enter the book index (0 to " + (n -
1) + "): ");
                    int index = scanner.nextInt();
                    scanner.nextLine(); // Consume newline character

                    if (index >= 0 && index < n && books[index] != null) {
                        System.out.print("Enter new Book Name: ");
                        String newName = scanner.nextLine();
                        System.out.print("Enter new Author Name: ");
                        String newAuthor = scanner.nextLine();
                        System.out.print("Enter new Price: ");
                        double newPrice = scanner.nextDouble();
                        System.out.print("Enter new Number of Pages: ");
                        int newNumPages = scanner.nextInt();
                        scanner.nextLine(); // Consume newline character

                        books[index].setName(newName);
                        books[index].setAuthor(newAuthor);
                        books[index].setPrice(newPrice);
                        books[index].setNumPages(newNumPages);
                        System.out.println("Book details updated
successfully!");
                    } else {
                        System.out.println("Invalid book index or no book
created at this index.");
                    }
                    break;

                case 3:
```

```
                    System.out.print("Enter the book index (0 to " + (n -
1) + "): ");
                    index = scanner.nextInt();
                    scanner.nextLine(); // Consume newline character

                    if (index >= 0 && index < n && books[index] != null) {
                        System.out.println("Book details:\n" +
books[index].toString());
                    } else {
                        System.out.println("Invalid book index or no book
created at this index.");
                    }
                    break;

                case 4:

                    System.out.println("\n--- All Books ---");
                    for (int i = 0; i < n; i++) {
                        if (books[i] != null) {
                            System.out.println("\nBook " + (i + 1) + ":");
                            System.out.println(books[i].toString());
                        } else {
                            System.out.println("No details for Book " + (i
+ 1) + " yet.");
                        }
                    }
                    break;

                case 5:

                    System.out.println("Exiting the program.");
                    scanner.close();
                    return;

                default:
                    System.out.println("Invalid choice. Please try
again.");
            }
        }
```

```
    }
}
```

OUTPUT:

```
Enter the number of books you want to create: 2

--- Book Store Menu ---
1. Create a Book
2. Set Book Details
3. Get Book Details
4. Display All Books
5. Exit
Enter your choice: 1
Enter Book Name: SINCHANA
Enter Author Name: SIN
Enter Book Price: 499
Enter Number of Pages: 129
Book created successfully!

--- Book Store Menu ---
1. Create a Book
2. Set Book Details
3. Get Book Details
4. Display All Books
5. Exit
Enter your choice: 1
Enter Book Name: SATWIK
Enter Author Name: SAT
Enter Book Price: 699
Enter Number of Pages: 128
Book created successfully!

--- Book Store Menu ---
1. Create a Book
2. Set Book Details
3. Get Book Details
4. Display All Books
5. Exit
Enter your choice: 2
```

```
Enter the book index (0 to 1): 1
Enter new Book Name: AMBIKA
Enter new Author Name: AMB
Enter new Price: 899
Enter new Number of Pages: 126
Book details updated successfully!

--- Book Store Menu ---
1. Create a Book
2. Set Book Details
3. Get Book Details
4. Display All Books
5. Exit
Enter your choice: 3
Enter the book index (0 to 1): 1
Book details:
Book Name: AMBIKA
Author: AMB
Price: 899.0
Number of Pages: 126

--- Book Store Menu ---
1. Create a Book
2. Set Book Details
3. Get Book Details
4. Display All Books
5. Exit
Enter your choice: 4

--- All Books ---

Book 1:
Book Name: SINCHANA
Author: SIN
Price: 499.0
Number of Pages: 129

Book 2:
Book Name: AMBIKA
Author: AMB
```

```
Price: 899.0
Number of Pages: 126

--- Book Store Menu ---
1. Create a Book
2. Set Book Details
3. Get Book Details
4. Display All Books
5. Exit
Enter your choice: 5
Exiting the program.
```

NOTES:

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.

CODE:

```
import java.util.Scanner;

abstract class Shape {

    int dimension1, dimension2;

    abstract void printArea();
}

class Rectangle extends Shape {

    public Rectangle(int length, int breadth) {
        this.dimension1 = length;
        this.dimension2 = breadth;
    }
    @Override
    void printArea() {
        int area = dimension1 * dimension2;
        System.out.println("Area of Rectangle: " + area);
```

```
    }
}


class Triangle extends Shape {

    public Triangle(int base, int height) {
        this.dimension1 = base;
        this.dimension2 = height;
    }

    @Override
    void printArea() {
        double area = 0.5 * dimension1 * dimension2;
        System.out.println("Area of Triangle: " + area);
    }
}
class Circle extends Shape {

    public Circle(int radius) {
        this.dimension1 = radius;
        this.dimension2 = 0; // No second dimension for circle
    }


    @Override
    void printArea() {
        double area = Math.PI * dimension1 * dimension1;
        System.out.println("Area of Circle: " + area);
    }
}

public class ShapeAreaCalculator {

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

        while (true) {
            System.out.println("\n--- Shape Area Calculator ---");
            System.out.println("1. Rectangle");
```

```
            System.out.println("2. Triangle");
            System.out.println("3. Circle");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

            Shape shape = null;

            switch (choice) {
                case 1:
                    System.out.print("Enter length of rectangle: ");
                    int length = scanner.nextInt();
                    System.out.print("Enter breadth of rectangle: ");
                    int breadth = scanner.nextInt();
                    shape = new Rectangle(length, breadth);
                    shape.printArea();
                    break;

                case 2:
                    System.out.print("Enter base of triangle: ");
                    int base = scanner.nextInt();
                    System.out.print("Enter height of triangle: ");
                    int height = scanner.nextInt();
                    shape = new Triangle(base, height);
                    shape.printArea();
                    break;

                case 3:
                    System.out.print("Enter radius of circle: ");
                    int radius = scanner.nextInt();
                    shape = new Circle(radius);
                    shape.printArea();
                    break;

                case 4:
                    System.out.println("Exiting the program.");
                    scanner.close();
                    return;

                default:
```

```
                    System.out.println("Invalid choice. Please try
again.");
            }
        }
    }
}
```

OUTPUT:

```
--- Shape Area Calculator ---
1. Rectangle
2. Triangle
3. Circle
4. Exit
Enter your choice: 1
Enter length of rectangle: 45
Enter breadth of rectangle: 95
Area of Rectangle: 4275

--- Shape Area Calculator ---
1. Rectangle
2. Triangle
3. Circle
4. Exit
Enter your choice: 2
Enter base of triangle: 88
Enter height of triangle: 44
Area of Triangle: 1936.0

--- Shape Area Calculator ---
1. Rectangle
2. Triangle
3. Circle
4. Exit
Enter your choice: 3
Enter radius of circle: 9
Area of Circle: 254.46900494077323

--- Shape Area Calculator ---
1. Rectangle
```

```
2. Triangle
3. Circle
4. Exit
Enter your choice: 4
Exiting the program.
```

NOTES:

PAGE NO :
DATE :

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

```
import java.util.Scanner;
abstract class Shape{
    int dim1;
    int dim2;
Shape(int dim1, int dim2){
    this.dim1 = dim1;
    this.dim2 = dim2;
abstract void printArea();
}
class Rectangle extends Shape{
    Rectangle(int width, int height){
        super(width, height);
}
@override
void printArea(){
    int area = dim1 * dim2;
    System.out.println("Area of Rectangle:"
        + area);
}
}
```

PAGE NO :
DATE :

```
@ override
void printArea () {
  double area = 0.5 * dim1 * dim2;
  System.out.println ("Area of Triangle " + area);
}
}

class Circle extends Shape {
      Circle (int radius) {
        super (radius, 0);
      }

@ override
  void printArea () {
    double area = Math.PI * dim1 * dim1;
System.out.println ("Area of Circle " + area);
}
}

class Main {
  public static void main (String [] args) {
  Shape rectangle = new Rectangle (5, 10);
  Shape triangle = new Triangle (4, 6);
  Shape circle = new Circle (3);
    rectangle.printArea ();
    triangle.printArea ();
    circle.printArea ();
  }
}
```

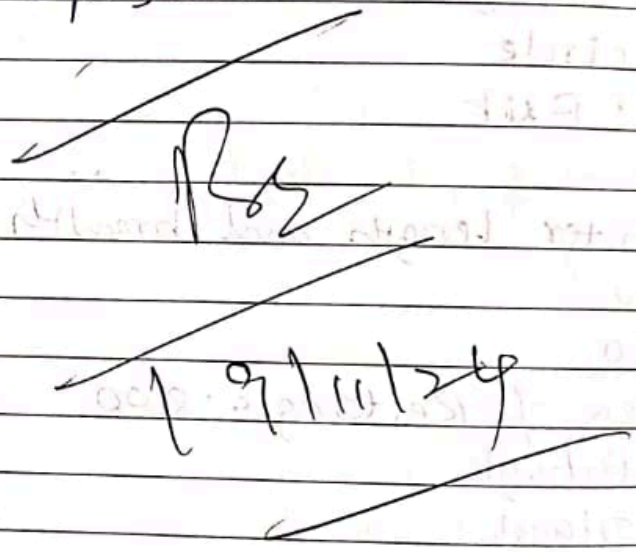
PAGE NO :
DATE :

output

```
Enter Shape of your choice.

1. Rectangle
2. Triangle
3. circle
4. Exit
1
Enter length and breadth
20
40
Area of Rectangle: 800
1. Rectangle
2. triangle
3. circle
4. Exit
2
Enter base and height
12
45
Area of triangle : 270.0
1. Rectangle
2. triangle
3. circle
4. Exit
3
Enter Radius of circle
10
Area of circle : 314.0
```

PAGE NO :
DATE :

1. Rectangle
2. Triangle
3. Circle
4. Exit
4
Exiting the program...



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.

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.

CODE:

```
import java.util.Scanner;


class Account {
    protected String customerName;
    protected int accountNumber;
    protected String accountType;
    protected double balance;


    public Account(String customerName, int accountNumber, String
accountType, double balance) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = balance;
    }


    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposit Successful. New Balance: " + balance);
    }
```

```
    public void displayBalance() {
        System.out.println("Current Balance: " + balance);
    }


    public void withdraw(double amount) {
        balance -= amount;
        System.out.println("Withdrawal Successful. New Balance: " +
balance);
    }
}


class CurAcct extends Account {
    private static final double MIN_BALANCE = 1000;
    private static final double PENALTY = 50;


    public CurAcct(String customerName, int accountNumber, double balance)
{
        super(customerName, accountNumber, "Current", balance);
    }

    @Override
    public void withdraw(double amount) {
        if (balance - amount < MIN_BALANCE) {
            System.out.println("Insufficient balance. Minimum balance of "
+ MIN_BALANCE + " must be maintained.");
        } else {
            super.withdraw(amount);
            if (balance < MIN_BALANCE) {
                balance -= PENALTY;
                System.out.println("Minimum balance breached. Penalty of "
+ PENALTY + " imposed. New Balance: " + balance);
            }
        }
    }


    public void issueChequeBook() {
```

```
        System.out.println("Cheque Book Issued.");
    }
}

class SavAcct extends Account {
    private static final double INTEREST_RATE = 0.05; // 5% compound
interest
    private static final int COMPOUND_FREQUENCY = 1; // yearly compound
interest

    // Constructor for Savings Account
    public SavAcct(String customerName, int accountNumber, double balance)
{
        super(customerName, accountNumber, "Savings", balance);
    }


    @Override
    public void withdraw(double amount) {
        if (amount <= balance) {
            super.withdraw(amount);
        } else {
            System.out.println("Insufficient funds for withdrawal.");
        }
    }

    public void computeAndDepositInterest() {
        double interest = balance * Math.pow((1 + INTEREST_RATE),
COMPOUND_FREQUENCY) - balance;
        balance += interest;
        System.out.println("Interest of " + interest + " computed and
added to your balance. New Balance: " + balance);
    }
}

public class amb {
    private static Scanner scanner = new Scanner(System.in);

    public static Account createAccount() {
        System.out.print("Enter customer name: ");
```

```
        String customerName = scanner.nextLine();
        System.out.print("Enter account number: ");
        int accountNumber = scanner.nextInt();
        System.out.print("Enter initial balance: ");
        double balance = scanner.nextDouble();
        scanner.nextLine(); // consume newline character

        System.out.println("Select Account Type (1 for Current, 2 for
Savings): ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // consume newline character

        if (choice == 1) {
            return new CurAcct(customerName, accountNumber, balance);
        } else if (choice == 2) {
            return new SavAcct(customerName, accountNumber, balance);
        } else {
            System.out.println("Invalid choice. Defaulting to Savings
Account.");
            return new SavAcct(customerName, accountNumber, balance);
        }
    }

    public static void main(String[] args) {
        System.out.println("Welcome to the Bank!");

        Account account = createAccount();

        boolean exit = false;
        while (!exit) {
            System.out.println("\nChoose an operation:");
            System.out.println("1. Deposit");
            System.out.println("2. Withdraw");
            System.out.println("3. Display Balance");
            System.out.println("4. Compute and Deposit Interest (only for
Savings Account)");
            System.out.println("5. Issue Cheque Book (only for Current
Account)");
            System.out.println("6. Exit");
```

```
            int option = scanner.nextInt();
            scanner.nextLine(); // consume newline character

            switch (option) {
                case 1:
                    System.out.print("Enter amount to deposit: ");
                    double depositAmount = scanner.nextDouble();
                    account.deposit(depositAmount);
                    break;

                case 2:
                    System.out.print("Enter amount to withdraw: ");
                    double withdrawAmount = scanner.nextDouble();
                    account.withdraw(withdrawAmount);
                    break;

                case 3:
                    account.displayBalance();
                    break;

                case 4:
                    if (account instanceof SavAcct) {
                        ((SavAcct) account).computeAndDepositInterest();
                    } else {
                        System.out.println("Interest can only be
calculated for Savings Account.");
                    }
                    break;

                case 5:
                    if (account instanceof CurAcct) {
                        ((CurAcct) account).issueChequeBook();
                    } else {
                        System.out.println("Cheque Book can only be issued
for Current Account.");
                    }
                    break;

                case 6:
                    exit = true;
```

```
                    System.out.println("Thank you for using our bank
services.");
                    break;

                default:
                    System.out.println("Invalid choice. Please try
again.");
            }
        }

        scanner.close();
    }
}
```

Output:

```
Welcome to the Bank!
Enter customer name: sinchana
Enter account number: 19839803
Enter initial balance: 700
Select Account Type (1 for Current, 2 for Savings):
1

Choose an operation:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute and Deposit Interest (only for Savings Account)
5. Issue Cheque Book (only for Current Account)
6. Exit
1
Enter amount to deposit: 800
Deposit Successful. New Balance: 1500.0
```

```
Choose an operation:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute and Deposit Interest (only for Savings Account)
5. Issue Cheque Book (only for Current Account)
6. Exit
2
Enter amount to withdraw: 450
Withdrawal Successful. New Balance: 1050.0

Choose an operation:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute and Deposit Interest (only for Savings Account)
5. Issue Cheque Book (only for Current Account)
6. Exit
5
Cheque Book Issued.

Choose an operation:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute and Deposit Interest (only for Savings Account)
5. Issue Cheque Book (only for Current Account)
6. Exit
6
Thank you for using our bank services.
```

NOTES:

PAGE NO :
DATE :

5. Develop a java program to create a class Bank that maintains two kinds of account for its customers, one called saving 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 maintaining 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 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 deposite 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 penality if necessary and update the balance.

PAGE NO:
DATE:

```
import java.util.Scanner;

class Account{
    protected String customername;
    protected int accountnumber;
              String accounttype;
              double balance;
    Account(String name, int accnum, String
    type, double initialbalance){

    customername = name;
     accountname = name;
     accounttype = type;
      balance = initialbalance

    void deposit(double amount){
       balance += amount
       System.out.println("Amount deposited" +
    amount);
       displaybalance();
    }

     void displaybalance(){
       System.out.println("balance" + balance
    }
    }

    class Savacct extend Account{
```

PAGE NO :
DATE :

```
Savacct (String name, int accNum, double
Initial balance) {

    Super (name, accNum, "Savings", initialbala
-nce);

void computeAndDepositInterest (int years)
  {
    double Interest = balance * Math.pow(1+
Interestrate).years) - balance;
    balance += Interest;
    System.out.println("Interest added" +
Interest);
displaybalance();
}

void withdraw (double amount) {
    if (balance >= amount) {
        balance -= amount;
    System.out.println("Amount withdrawn
- amount);
}
else {
    System.out.println("Insufficient
balance for withdrawal");
  }

displaybalance();
}
```

PAGE NO :
DATE :

```
class CurAcct extends Account {
 double minimumbalance = 500.0;

 double servicecharge = 50.0;

 CurAcct (String name, int accNum, double
 initialbalance) {
 super (name, accNum, "Current", initial
balance);
}

 void withdraw (double amount) {
  if (balance >= amount) {
   balance -= amount;
  System.out.println ("Amount withdrawn" + amount
  if (balance < minimumbalance) {
        balance -= service charge;
          System.out.println (servicecharge);
   }
  else {
  System.out.println ("Insufficient balance
  for with draw");
  }
   displaybalance();
 }
}
public class bank {
     public static void main (String[] args)
{
  Scanner sc = new
```

PAGE NO :
DATE :

```
double intbalance = sc.nextDouble();
Savacct Savaccount = new Savacct (name,
acc, intbalance);
    savacct.deposit (1000);
    Savacct.withdraw (500);
     sc.nextLine();
System.out.println ("Enter details of current
account");
String n2 = sc.nextLine();
Int acc2 = sc.nextInt();
double intbalance 2 = sc.nextDouble();

Curracct (n2, acc 2, intbalance 2);

   Curaccount.deposit (2000);
   Curaccount.withdraw (1800);
     sc.close();
}
}.
```

output

```
Enter details saving account
customer name : Alice
   acc no : 12345
  initial deposit : 5000
amount deposit : 1000.0
balance = 6000.0
Interest added : 240.0
   balance : 6240.0
   Amount withdrawn : 500.0
```

PAGE NO :
DATE :

Enter details for current account
customer name : Bob
acc =6780
Initial deposit : 2000
Amount deposit : 2000.0
balance : 4000.0
Amount withdrawn : 1800.0
balance below minimum : service charge
imposed : 50.0
balance : 2150.0

19/11/2

6 create a package CIE which has a classes - stud
& internals the class student has members US
usn, name, sem. the class internals has an
array that stores the internal marks stud
which has the class External which is a
derived class of student that declaring the
final marks of n student in all 5 courses

PROGRAM 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.

CODE:

```
import CIE.Student;
import CIE.Internals;
import SEE.Externals;
import java.util.Scanner;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of students whose details you
want to enter");
        int n = sc.nextInt();
        Internals[] i1 = new Internals[n];
        Externals[] e1 = new Externals[n];
        for(int i=0;i<n;i++) {
            System.out.println("Student "+(i+1)+" details:");
            e1[i] = new Externals();
            i1[i] = new Internals();
            e1[i].getd();
            i1[i].getMarks();
            e1[i].getMarks();
        }
        for(int i=0;i<n;i++) {
            e1[i].display();
            e1[i].calcTotalMarks(i1[i]);
        }

    }
}


//student.java

package CIE;
import java.util.Scanner;
public class Student {
    String usn;
    String name;
    int sem;
```

```
    public void getd() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter student USN");
        usn = sc.nextLine();
        System.out.println("Enter student name");
        name = sc.nextLine();
        System.out.println("Enter semester");
        sem = sc.nextInt();
    }
    public void display() {
        System.out.println();
        System.out.println("Student USN: "+usn);
        System.out.println("Student name: "+name);
        System.out.println("Semester: "+sem);
        System.out.println();
    }
}


//internal.java
package CIE;
import java.util.Scanner;
public class Internals {
    public int marksCie[];
    public void getMarks() {
        Scanner sc=new Scanner(System.in);
        System.out.println("enter no of subjects in cie");
        int n=sc.nextInt();
        marksCie=new int[n];
        // System.out.println("Enter No. Of Students: ");

        for(int i=0;i<n;i++) {
            System.out.println("Enter CIE marks in subject "+(i+1));
            marksCie[i]=sc.nextInt();
        }
    }
    public int returnCieMarks(int i) {
        return marksCie[i];
    }
}
```

```java
//external.java



package SEE;

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

public class Externals extends Student {
    Scanner sc = new Scanner(System.in);
    int n;
    int marksSee[];

    public void getMarks() {
        System.out.println("Enter the number of subjects");
        n = sc.nextInt();
        marksSee = new int[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Enter SEE marks in subject " + (i + 1));
            marksSee[i] = sc.nextInt();
        }
    }

    public void calcTotalMarks(Internals i1) {
        for (int i = 0; i < n; i++) {
            System.out.println("Subject " + (i + 1) + ": " +
(i1.returnCieMarks(i) + (marksSee[i] / 2)));
        }
        System.out.println();

    }
}
```

OUTPUT:

```
Enter the number of students whose details you want to enter
2
Student 1 details:
Enter student USN
1BM23CS001
Enter student name
Alice
Enter semester
5
Enter the number of subjects in cie
2
Enter CIE marks in subject 1
18
Enter CIE marks in subject 2
20
Enter the number of subjects
2
Enter SEE marks in subject 1
72
Enter SEE marks in subject 2
68
Student 2 details:
Enter student USN
1BM23CS002
Enter student name
Bob
Enter semester
5
Enter the number of subjects in cie
2
Enter CIE marks in subject 1
15
Enter CIE marks in subject 2
17
Enter the number of subjects
2
Enter SEE marks in subject 1
80
Enter SEE marks in subject 2
```

```
75



Student USN: 1BM23CS001
Student name: Alice
Semester: 5

Subject 1: 54
Subject 2: 54

Student USN: 1BM23CS002
Student name: Bob
Semester: 5

Subject 1: 55
Subject 2: 54
```

NOTES:

PAGE NO :
DATE :

```
package CIE;
import java.util.Scanner;
public class student {
    String name;
    String usn;
    int sem;
    public void getd() {
    Scanner sc = new Scanner (System.in);
    usn = sc.nextLine();
    name = sc.nextLine();
    sem = sc.nextInt();
}

public void display() {
    System.out.println();
    System.out.println("student USN" +
    usn);
System.out.println("student name" +name);
System.out.println("semester" +sem);
System.out.println();
}

package CIE;
import java.util.Scanner;
public class Internals {
public int marksde[] = new int[5];
public void getmarks() {
for (int i=0; i<5; i++) {
    Scanner x = new Scanner(System.in);
```

PAGE NO :
DATE :

```
public int returnmarks(int i){
        return marks[i];
    }
}

package SEE;
import CIE.Student;
import CIE.Internals;
import java.util.Scanner;
public class Externals extends Student {
    int marksSee[]=new int[5]

public void getmarks(){
        for(int i=0;i<5;i++){
            Scanner sc=new Scanner(System.in
        System.out.println("Enter SEE marks in
            Subject"+(i+1));
            marksSEE[i]= sc.nextInt();
    }

public void costotalmarks(Internals ii)
{
    for(int i=0;i<5;i++){
    System.out.println("subject"+(i+1)+"t
        (i+returnmembrksCIE(i)+(marks[i]/2
    }
}
}
```

PAGE NO :
DATE :

```
public static void main (String args[])
{
    Scanner sc = new Scanner (System.in);
    System.out.println (" Enter number of
students");
    int n = sc.nextInt();
Internals [] i1 = new Internals [n];
Externals [] e1 = new Externals [n];
for (int i=0; i<n; i++) {
        e1[i] = new Externals();
        i1[i] = new Internals();
        e1[i].getd();
        i1[i].getmarks();
        e1[i].getmarksc();
}

for (int i=0; i<n; i++){
        e1[i].display();
        e1[i].calctotalmarks (i1[i]);
}
}
}
```

output

Enter the number of students
2
Student 1 details:
Enter student USN
1BM23CS329
enter student name
sinchana
Enter semester

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

CODE:

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


class Father {
    int fatherAge;


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

```
    }


    public int getFatherAge() {
        return this.fatherAge;
    }
}


class Son extends Father {
    int sonAge;


    public Son(int fatherAge, int sonAge) throws WrongAge {
        // Call the constructor of the Father class
        super(fatherAge);


        if (sonAge < 0) {
            throw new WrongAge("Son's age cannot be negative.");
        }
        if (sonAge >= fatherAge) {
            throw new WrongAge("Son's age cannot be greater than or equal
to Father's age.");
        }

        this.sonAge = sonAge;
    }

    public int getSonAge() {
        return this.sonAge;
    }
}

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

            Father father = new Father(40);
```

```
            Son son = new Son(40, 20);
            System.out.println("Father's age: " + father.getFatherAge());
            System.out.println("Son's age: " + son.getSonAge());


         //   Son invalidSon = new Son(40, 45); // Throws exception
            // Son invalidAgeSon = new Son(40, -5); // Throws exception

        } catch (WrongAge e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
Father's age: 40
Son's age: 20
```

NOTES:

Lab-7.

PAGE NO :
DATE :

1. write a program that demonstrates handling of exceptions in inheritance tree. Create a base called "Father" and derive of class called "son" which extends the base class. In Father class implement a constructor which takes the age and throws the exception wrongage() when input age<0. the son class, implement a constructor that uses both father and son's age and throws an exception if son's age is >= father age.

→

```
class wrongage extends exception{
    public wrongage (String message){
        super(message);
    }
}
class father{
    int fatherage;
    public Father(int age) throws wrongage
{
    if (age<0){
        throw new wrongage ("Father's age cannot be negative");
}
this.Fatherage =age;
}
}
Class son extends Father{
    int sonage;
    Public Son (int Fatherage, int sonage)
```

```
super (fatherage);
    if (sonage >= fatherage){
      throw new wrongage ("son's age
cannot be greater or equal father age");
}

this.son = sonage;
}
}

public class main{
      psvm(){
try {
  // Son  son3 = new Son(-1,10);
    Son son1 = new son(50,25);
    Sout ("Father age" + son1.fatherage
  +  "son's age" + son1.sonage);
  }
  catch (wrong age e) {
        Sout ("Exleption caught " + e.
    getMessage());
    }
    }
```

Output

Father's age 50

Son's age 25

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.

CODE:

```
class CollegeThread extends Thread {
    @Override
    public void run() {
        try {
            while (true) {
                System.out.println("BMS College of Engineering");
                Thread.sleep(10000); // Sleep for 10 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("CollegeThread interrupted: " +
e.getMessage());
        }
    }
}


class CSEThread extends Thread {
    @Override
    public void run() {
        try {
            while (true) {
                System.out.println("CSE");
                Thread.sleep(2000); // Sleep for 2 seconds
            }
        } catch (InterruptedException e) {
```

```
            System.out.println("CSEThread interrupted: " +
e.getMessage());
        }
    }
}


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

        CollegeThread collegeThread = new CollegeThread();
        CSEThread cseThread = new CSEThread();


        collegeThread.start();
        cseThread.start();
    }
}
```

OUTPUT

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

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

NOTES:

Lab-8 :

PAGE NO :
DATE :

1. Write a program which creates two threads, one thread displaying "BMS college of engineering" once every ~~two~~ ten seconds and another displaying "CSE" once every two seconds.

→

```
class BMSCollegeThread extends Thread{
    public void run(){
        try{
            while (True){
                System.out.println("BMS College of Engineering");
                Thread.sleep(10000);
            }
        catch (InterruptedException e){
            System.out.println("BMS College Thread interrupted");
        }
    }
}
class CSEThread extends Thread{
    public void run(){
        try{
            while (true){
                System.out.println ("CSE");
                Thread.sleep(2000);
            }
        } catch (InterruptedException e){
```

PAGE NO :
DATE :

```
public class Main{
    public static void main (String[] args)
    {
    BMScollegeThread bmsThread = new BMS
    collegeThread()
     CSEThread cseThread =new CSEThreadin;
      bmsThread.start();
      cseThread.start();
    }
}
```

output

BMS college of engineering
CSE
CSE
CSE
CSE
CSE.

PROGRAM 9

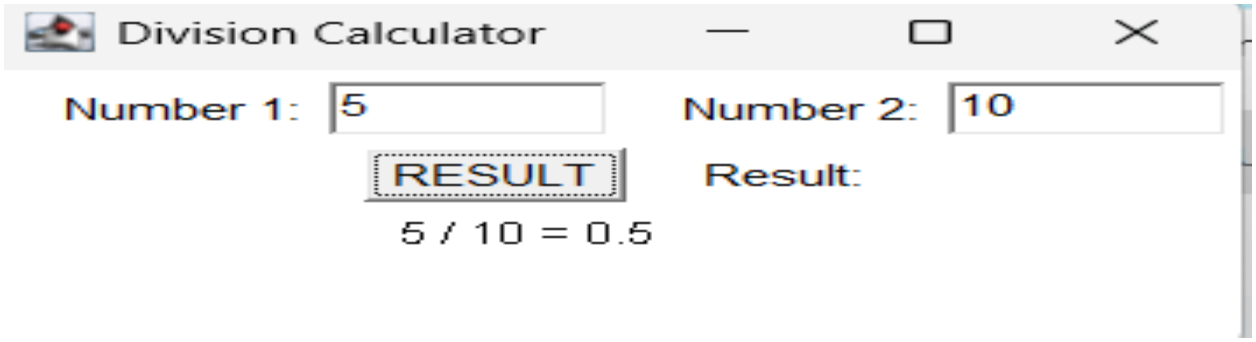
Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException.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

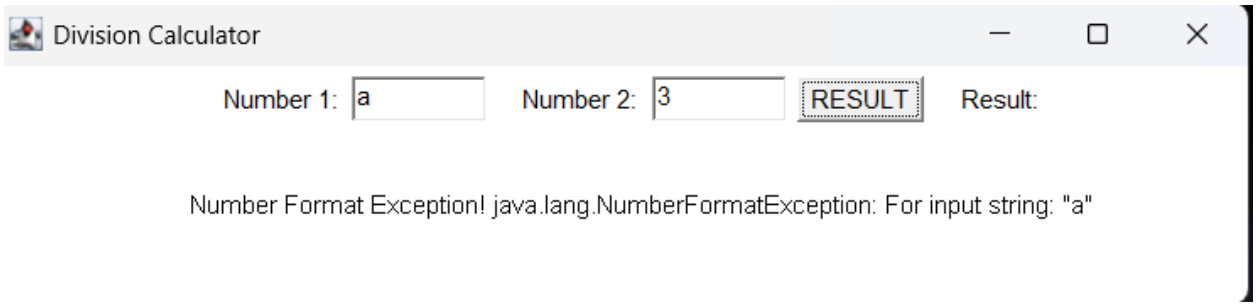
```
import java.awt.*;
import java.awt.event.*;

public class DivisionMain1 extends Frame implements ActionListener {
    TextField num1, num2;
    Button dResult;
    Label outResult;
    String out = "";
    double resultNum;
    int flag = 0;

    public DivisionMain1() {
        setLayout(new FlowLayout());
```

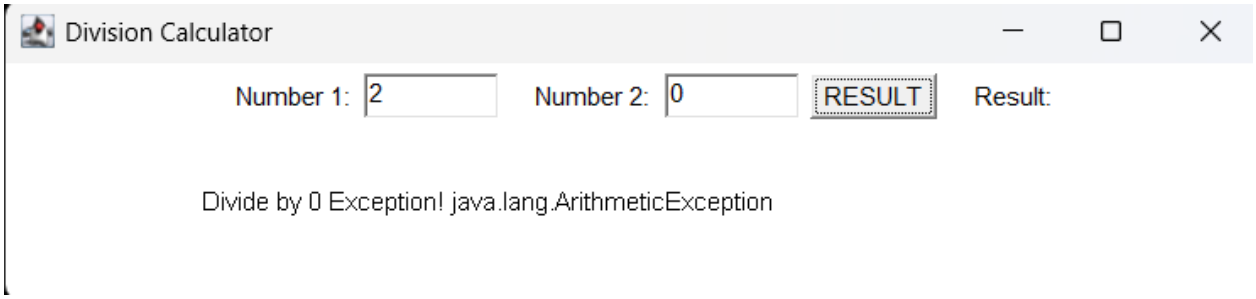
```
        dResult = new Button("RESULT");
        Label number1 = new Label("Number 1:", Label.RIGHT);
        Label number2 = new Label("Number 2:", Label.RIGHT);
        num1 = new TextField(5);
        num2 = new TextField(5);
        outResult = new Label("Result:", Label.RIGHT);

        add(number1);
        add(num1);
        add(number2);
        add(num2);
        add(dResult);
        add(outResult);

        num1.addActionListener(this);
        num2.addActionListener(this);
        dResult.addActionListener(this);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });

        setTitle("Division Calculator");
        setSize(300, 150);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent ae) {
        int n1, n2;
        try {
            if (ae.getSource() == dResult) {
                n1 = Integer.parseInt(num1.getText());
                n2 = Integer.parseInt(num2.getText());

                if (n2 == 0) {
                    throw new ArithmeticException();
                }

                resultNum = (double) n1 / n2;
```

```
                out = n1 + " / " + n2 + " = " + resultNum;
                repaint();
            }
        } catch (NumberFormatException e1) {
            flag = 1;
            out = "Number Format Exception! " + e1;
            repaint();
        } catch (ArithmeticException e2) {
            flag = 1;
            out = "Divide by 0 Exception! " + e2;
            repaint();
        }
    }

    public void paint(Graphics g) {
        if (flag == 0) {
            g.drawString(out, 100, 100);
        } else {
            g.drawString(out, 100, 100);
            flag = 0;
        }
    }

    public static void main(String[] args) {
        new DivisionMain1();
    }
}
```

OUTPUT:


Division Calculator
Number 1:
5
Number 2:
10
RESULT
Result:
5 / 10 = 0.5


Division Calculator
Number 1:
a
Number 2:
3
RESULT
Result:
Number Format Exception! java.lang.NumberFormatException: For input string: "a"


Division Calculator
Number 1:
2
Number 2:
0
RESULT
Result:
Divide by 0 Exception! java.lang.ArithmeticException

NOTES:

Lab-9

PAGE NO :
DATE :

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 filed when divide button is clicked. if num1 or num2 were not a integer an arithematic exception Display the exceptions in the message dialog box.

→
```
import java.awt.*;
import java.awt.event.*;

public class DivisionMain extends Frame
implements ActionListener
{
  TextField
  num1,num2;
  Button dResult;
  label outResult;
  String out = "";
  double result+Num;
  int flag = 0;

public DivisionMain()
  { setLayout(new FlowLayout());

  dResult = new Button("Result");
  label number1 = new label("Number1",
  Label.Right);
```

PAGE NO :
DATE :

```
Label number2 = new Label("Number 2:",
Label.RIGHT);
num1 = new TextField(5);
num2 = new TextField(5);
outResult = new Label("Result ", Label.RIGHT);

add(number1);
add(num1);
add(number2);
add(num2);
add(dResult);
add(outRerult);
num1.addActionListner(this);
num2.addActionListner(this);
addwindowListner(this);
windowAdapter()
{
  public void windowClosing(windowEvent
we)
  {
    System.exit(0);
  }
});
public void actionperformed(ActionEvent ae)
  {
    int n1, n2;
    try{
      if(ae.getSource() == dResult)
```

PAGE NO :
DATE :

```
if (n2==0)
 throw new
ArithmeticException();
out=n1+" "+n2+" ";
 resultNum= n1/n2;
 out+=String.valueOf(resultNum);
 repaint();
}
}
catch (NumberFormatException e1)
{
 flag=1;
  out= "Number Format Exception! "+e1;
  repaint();
}
 catch (ArithmaticException e2)
 {
  flag=1;
   out= "Divide by 0 Exception! " +e2;
  repaint();
  }
}

public void paint(Graphics g)
{
 if(flag==0)
  g.drawString(out, outResult.getX()+
  outResult.getWidth(), outResult.getY
  ()+(outResult.getHeight-8);
   else
```

PROGRAM 10:

Demonstrate Inter process Communication

10 A)

CODE:

```
class Q {
    int n;
    boolean valueSet = false;

    synchronized int get() {
        while (!valueSet) {
            try {
                System.out.println("\nConsumer waiting\n");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }
        System.out.println("Got: " + n);
        valueSet = false;
        System.out.println("\nNotify Producer\n");
        notify();
        return n;
    }

    synchronized void put(int n) {
        while (valueSet) {
            try {
                System.out.println("\nProducer waiting\n");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }
        this.n = n;
```

```
        valueSet = true;
        System.out.println("Put: " + n);
        System.out.println("\nNotify Consumer\n");
        notify();
    }
}

class Producer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }
    public void run() {
        int i = 0;
        while (i < 15) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;
    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }
    public void run() {
        int i = 0;
        while (i < 15) {
            int r = q.get();
            System.out.println("Consumed: " + r);
            i++;
        }
    }
}

public 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.");
    }
}
```

OUTPUT:

```
Consumed: 1
Put: 2

Notify Consumer


Producer waiting

Got: 2

Notify Producer

Consumed: 2
Put: 3

Notify Consumer


Producer waiting

Got: 3

Notify Producer

Consumed: 3
Put: 4

Notify Consumer


Producer waiting
```

```
Got: 4

Notify Producer

Consumed: 4
Put: 5

Notify Consumer


Producer waiting

Got: 5

Notify Producer

Consumed: 5
Put: 6

Notify Consumer


Producer waiting

Got: 6

Notify Producer

Consumed: 6
Put: 7

Notify Consumer


Producer waiting

Got: 7

Notify Producer
```

```
Consumed: 7
Put: 8

Notify Consumer


Producer waiting

Got: 8

Notify Producer

Consumed: 8
Put: 9

Notify Consumer


Producer waiting

Got: 9

Notify Producer

Consumed: 9
Put: 10

Notify Consumer


Producer waiting

Got: 10

Notify Producer

Consumed: 10
Put: 11
```

```
Notify Consumer


Producer waiting

Got: 12

Notify Producer

Consumed: 12
Put: 13

Notify Consumer


Producer waiting

Got: 13

Notify Producer

Consumed: 13
Put: 14

Notify Consumer

Got: 14

Notify Producer

Consumed: 14
```

NOTES:

Lab - 10

PAGE NO :
DATE :

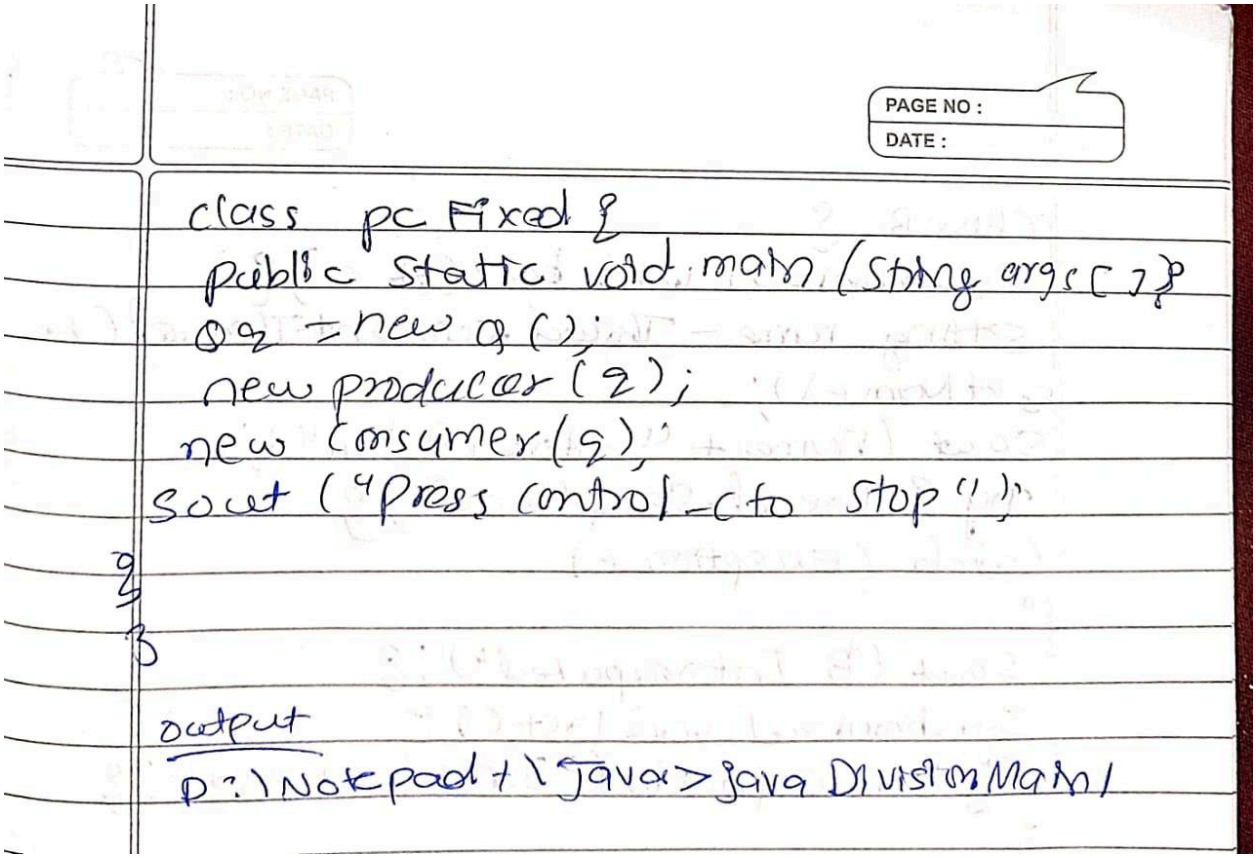
a) Demonstrate Interprocess communication and deadlock.

=>

```
class Q {
 int n;
 boolean valueset = false;
 Synchronized int get() {
 while(!valueset)
 try {
 System.out.println("\n customer waiting");
 wait();
 }
 catch (InterruptedException e) {
   Sout("Interrupted Exception caught");
 }
 Sout("Got " + n);
 valueset = false;
 Sout("Intimate producer");
 notify();
 return n;
 }
 synchronized void put(int n) {
 while(valueset)
 try {
 Sout("producer waiting");
 wait();
 }
 catch (InterruptedException e)
 {
 Sout("Interrupted Exception caught");
 }
 this.n = n;
```

PAGE NO :
DATE :

```
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();
Sout ("consumed "+r);
 i++;
}
}
}
```



10 B)
DEMONSTRATION OF DEADLOCK

CODE:

```
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();
    }
```

```
    synchronized 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();
    }

    synchronized void last() {
        System.out.println("Inside B.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();
    }
}
```

OUTPUT:

```
MainThread entered A.foo
RacingThread entered B.bar
RacingThread trying to call A.last()
MainThread trying to call B.last()
```

NOTES:

(5) Demonstration of 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();}
Synchronized void last(){
System.out.println("Inside A.last"); }
}
```

PAGE NO :
DATE :

```
class B {
synchronized void bar(A a){
String name = Thread.currentThread().
getName();
Sout (name + "entered B.bar");
try { Thread.sleep(1000); }
Catch (Exception e)
{
Sout ("B Interrupted"); }
Synchronized void last() {
System.out.println ("Inside A last"); }
}

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