VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

OBJECT ORIENTED JAVA PROGRAMMING(23CS3PCOOJ)

Submitted by

THILAK K(1WA23CS021)

in partial fulfilment 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 **THILAK K(1WA23CS021),** 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. The Lab report has been approved as it satisfies the academic requirements in respect of a **"OBJECT ORIENTED JAVA PROGRAMMING (23CS3PCOOJ)"**work prescribed for the said degree.

Dr. Prasad GR Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

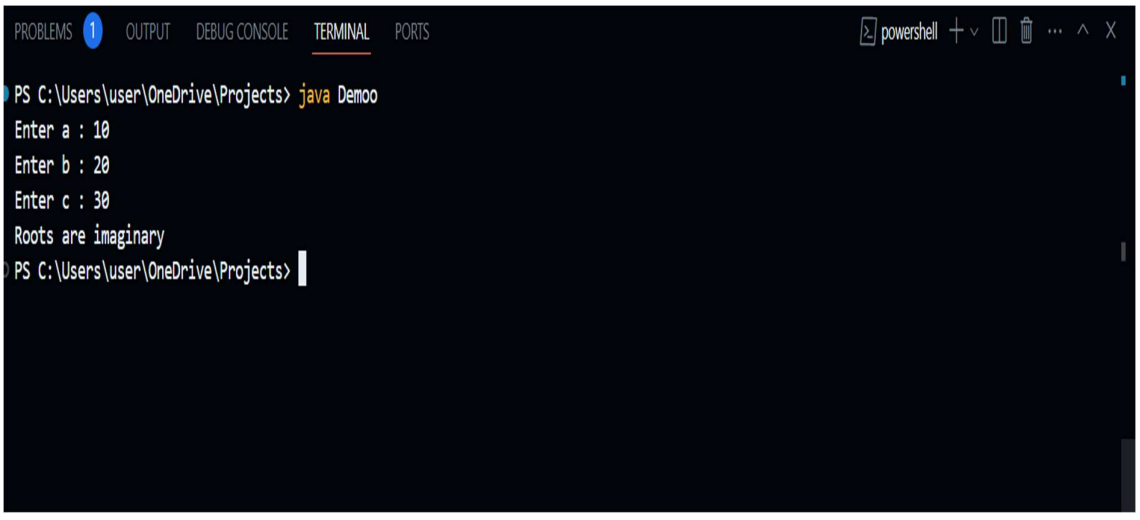
Index

PROGRAM -1

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

```
class Quadratic
{
   double a;
   double b;
   double c;
   double root1, root2;

   public void calculateroots()
   {
      double discriminant = (b * b) - (4 * a * c);

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

```
    }
  }
}

public class Demoo
{
  public static void main(String args[])
  {
    Quadratic q = new Quadratic();
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter a : ");
    q.a = sc.nextDouble();
    System.out.print("Enter b : ");
    q.b = sc.nextDouble();
    System.out.print("Enter c : ");
    q.c = sc.nextDouble();
    q.calculateroots();
  }
}
```



Q) Develop java program that prints all real solutions to the quadratic equations ax^2+bx+c

```
class Quadratic
{
    double a;
    double b;
    double c;
    double root1, root2;
    public void calculateRoots()
    {
        double discriminant = (b*b)-(4*a*c);
        if (discriminant==0)
        {
         System.out.println("Roots are real & equal");
          root1 = (-b + Math.sqrt(discriminant)/(2*a));
          System.out.println("Root1 :" + root1);
        }
       else if ( discriminant <0)
        {
          System.out.println(" Roots are imaginary");
        }
       else
        {
          System.out.println(" Roots are real & distinct");
          root1 = (-b+ Math.sqrt(discriminant)/(2*a);
```

```
        root2 = (-b/- math.sqrt (discriminant) / (2*a));
        System.out.println (" Root 1 " + root 1
                                + "Root 2 " + root 2);

        }
    }
}

public class Demoo
{
    public static void main (String a[])
    {
        Quadratic q = new Quadratic();
        Scanner sc = new Scanner (System.in);
        System.out.println ("Enter a : ");
        q.a = sc.nextDouble();
        System.out.println ("Enter b : ");
        q.b = sc.nextDouble();
        System.out.println ("Enter c : ");
        q.c = sc.nextDouble();
        q.calculateroots();
    }
}
```

Output :

```
Enter a : 1
Enter b : 3
Enter c : 2
```

Roots are real & distinct

Root 1 : -2.5 Root 2 : -3.5.

PROGRAM -2

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

```
import java.util.Scanner;


class Example
{
   String name;
   String usn;
   public int[] marks = new int[8];
   public int[] grades = new int[8];
   public int[] credits = new int[8];


   public int creditssum(int arr[])
   {
      int s = 0;
      for (int i : arr)
      {
         s += i;
      }
      return s;
   }


   public void accept()
   {
      System.out.println("Enter the details : ");
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter name : ");
      name = sc.nextLine();
      System.out.print("Enter your USN : ");
      usn = sc.nextLine();
```

```
for (int i = 0; i < 8; i++)
{
   System.out.print("Enter subject " + (i + 1) + " marks : ");
   marks[i] = sc.nextInt();
   System.out.print("Enter the subject credit : ");
   credits[i] = sc.nextInt();
   if (marks[i] >= 90 && marks[i] <= 100)
   {
      grades[i] = 10;
   }
   else if (marks[i] >= 80 && marks[i] < 90)
   {
      grades[i] = 9;
   }
   else if (marks[i] >= 70 && marks[i] < 80)
   {
      grades[i] = 8;
   }
   else if (marks[i] >= 60 && marks[i] < 70)
   {
      grades[i] = 7;
   }
   else if (marks[i] >= 50 && marks[i] < 60)
   {
      grades[i] = 6;
   }
   else if (marks[i] >= 40 && marks[i] < 50)
   {
      grades[i] = 5;
   }
   else if (marks[i] >= 30 && marks[i] < 40)
```

```
            {
                grades[i] = 4;
            }
            else if (marks[i] >= 20 && marks[i] < 30)
            {
                grades[i] = 3;
            }
            else if (marks[i] >= 10 && marks[i] < 20)
            {
                grades[i] = 2;
            }
            else if (marks[i] >= 0 && marks[i] < 10)
            {
                grades[i] = 1;
            }
        }
    }

    public void sgpa()
    {
        double sum = 0;
        System.out.println("\nName : " + name);
        System.out.println("USN : " + usn);
        int cr = creditssum(credits);
        for (int i = 0; i < 8; i++)
        {
            sum += (credits[i] * grades[i]);
        }
        System.out.println("SGPA is :" + sum / cr);
    }
}
```

```
public class Student
{
   public static void main(String[] args)
   {
      Example ex = new Example();
      ex.accept();
      ex.sgpa();
   }
}
```

OUTPUT

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 method to calculate SGPA & CGPA of a student

NOTE: Add sem 1 and sem 2 marks of all subject and compute SGPA and CGPA.

```
import java.util.Scanner;
class Student
{
    public String usn, name;
    public int credits[] = new int[8];
    public int marks[] = new int[8];
    public int grades[] = new int[8];
    public void accept()
    {
     Scanner sc = new Scanner(System.in);
     System.out.println("Enter details :");
     System.out.println("Enter name :");
     name= sc.NextLine();
     System.out.println("Enter USN :");
     usn = sc.nextLine();
     for (int i=0; i<8; i++)
     {
```

```
System.out.println("Enter "+(i+1)+" subject mark:");
marks[i] = sc.nextInt();
System.out.println("Enter subject "+(i+1)+" marks");
credit[i] = sc.nextInt();
if (marks[i] > 90 && marks[i] <=100)
{
   grades[i] = 10;
}
else if (marks[i] >=80 && marks[i] < 90)
{
   grades[i] = 9;
}
else if (marks[i] >=70 && marks[i] <80)
{
   grades[i] = 8;
}
else if (marks[i] >=60 && marks[i] <70)
{
   grades[i] = 7;
}
else if (marks[i] >=50 && marks[i] <60)
{
   grades[i] = 6;
}
else if (marks[i] >=40 && marks[i] <50)
{
   grades[i] = 5;
}
```

```
        else if (grades[i]>=30 && grades[i]<40)
        {
            grades[i]=4;
        }
        else if (grades[i]>=20 && grades[i]<30)
        {
            grades[i]=3;
        }
        else if (grades[i]>=10 && grades[i]<20)
        {
            grades[i]=2;
        }
        else if (grades[i]>=0 && grades[i]<10)
        {
            grades[i]=1;
        }
    }
    public int creditSum()
    {
        int sum=0;
        for (int i=0; i<8; i++)
        {
            sum+=credits[i];
        }
        return sum;
    }
```

```
public double sgpa()
{
    double sgpa=0;
    for (int i=0; i<8 ; i++)
    {
      sgpa += (credits[i] * grades[i];
    }
    return (sgpa /(creditssum()));
  }
}
public class cgpa
{
   public static void main (String args[])
   {
     Student stud[ ] = new Student[2];
     for (int i=0 ; i<2 ; i++)
     {
       stud[i]= new student();
       system.out. println (" Enter semester details");
       stud[i]. accept();
     }
     double cgpa=0,
     for (int i=0; i<2 ; i++)
     {
       cgpa += stud[i]. sgpa();
     }
```

```
        System.out.println (" CGPA : " + (cgpa/2));
        System.out.println (" Sem 1 SGPA :" + Stud[0] SGPA
        System.out.println (" Sem 2 SGPA :" + Stud[i] SGPA
    }
}
```

OUTPUT:

```
Enter student 1 sem details :
Enter details
Enter your name : Nilak K.
Enter your usn : 1WA23CS021
Enter subject 1 marks : 97
Enter subject 1 credits : 4
Enter subject 2 marks : 96
Enter subject 2 credits : 4
Enter subject 3 marks : 87
Enter subject 2 credits : 3
Enter subject 4 marks : 88
Enter subject 4 credits : 3
Enter subject 5 marks : 87
Enter subject 5 credits : 3.
```

Enter subject 6 marks : 88
Enter subject 6 credit : 1

Enter subject 7 marks : 99
Enter subject 7 credits : 1

Enter subject 8 marks : 90
Enter subject 8 credits : 1.

Enter details
Enter your name : Thilak k
Enter your usn : 1WA23CS021

Enter subject 1 marks : 98
Enter subject 1 credits : 4

Enter subject 2 marks : 96
Enter subject 2 credits : 4

Enter subject 3 marks : 88
Enter subject 3 credits : 3

Enter subject 4 marks : 90
Enter subject 4 credits : 3

Enter subject 5 marks : 96
Enter subject 5 credits : 3

Enter subject 6 marks : 96
Enter subject 6 credits : 1

Enter subject 7 marks : 92
Enter subject 7 credits : 1

Enter subject 8 marks : 95
Enter subject 8 credits : 1

CGPA is 9.675
Sem 1 SGPA : 9.5
Sem 2 SGPA : 9.85

PROGRAM -3:

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

NOTE: 1: Use normal display method

2: Use Override toString method

Eg: public String toString() {Write code to display data }

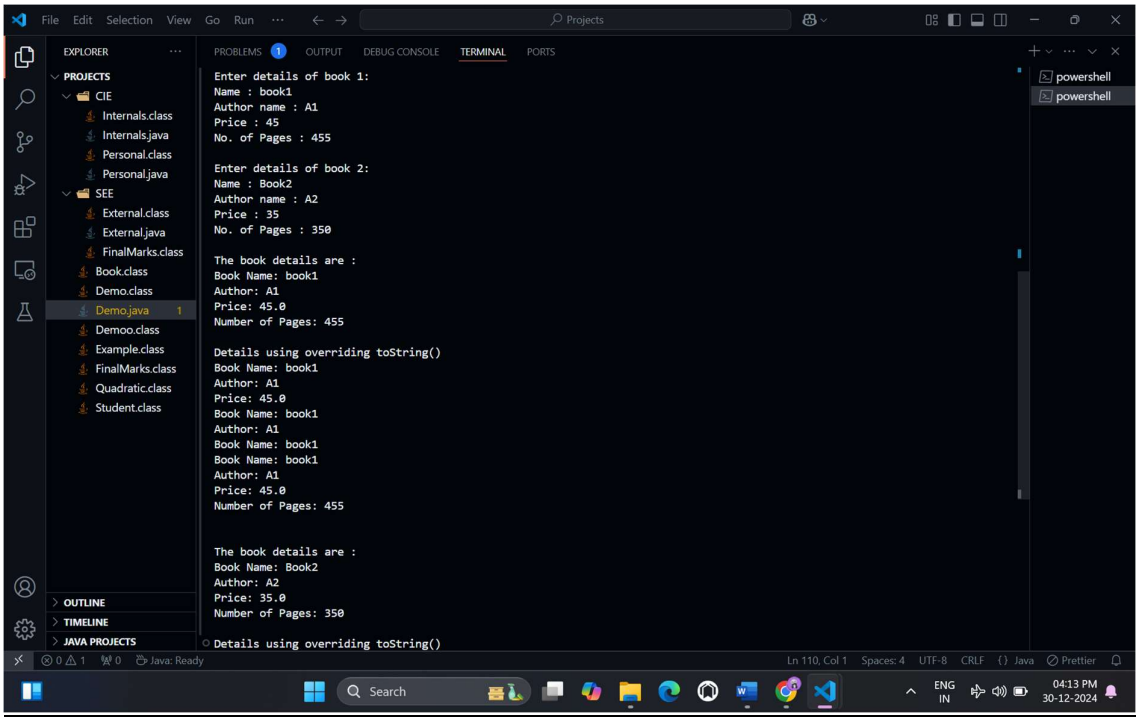
```
import java.util.Scanner;

class Book
{
   String name, author;
   double price;
   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 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 num_pages)
{
   this.num_pages = num_pages;
}

public String getName()
{
   return this.name;
}

public String getAuthor()
{
   return this.author;
}

public double getPrice()
{
   return this.price;
}

public int getNumPages()
```

```
    {
        return this.num_pages;
    }

    public void display()
    {
        System.out.println("Book Name: " + name);
        System.out.println("Author: " + author);
        System.out.println("Price: " + price);
        System.out.println("Number of Pages: " + num_pages);
    }

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

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

        System.out.print("Enter the number of books : ");
        n = sc.nextInt();
        sc.nextLine(); // Consume newline

        Book books[] = new Book[n]; // Creating array of n books
```

```
        for (int i = 0; i < n; i++)
        {
            System.out.println("\nEnter details of book " + (i + 1) + ":");
            System.out.print("Name : ");
            String name = sc.nextLine();
            System.out.print("Author name : ");
            String author = sc.nextLine();
            System.out.print("Price : ");
            double price = sc.nextDouble();
            System.out.print("No. of Pages : ");
            int num_pages = sc.nextInt();
            sc.nextLine(); // Consume newline

            books[i] = new Book(name, author, price, num_pages);
        }

        for (int i = 0; i < n; i++)
        {
            System.out.println("\nThe book details are : ");
            books[i].display();
            System.out.println("\nDetails using overriding toString()");
            System.out.println(books[i]);
        }
    }
}
```

CODE



Create a class book which contains four members : name, author, price, num_pages. Include method to set the values for the members. Include methods to set and get the details of the object. Display complete details of the book. Develop a java program to create n books.

NOTE:
1) use normal display method
2) use override toString method.

```
import java.util.Scanner;
class Book
{
    String name, author;
    double price;
    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 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 num_pages)
{
    this num_pages = num_pages;
}
public String GetName()
{
    return this.name;
}
public String getAuthor()
{
    return this.author;
}
public double getPrice()
{
    return this.price;
}
```

```
public int getNumPages()
{
    return this.num_pages;
}
public void display()
{
    System.out.println("Book Name : " +name);
    System.out.println(" Author name : " + name);
    System.out.println("Price : " + price);
    System.out.println("Pages : " + num_pages);
}
public String toString()
{
    return "Book Name: " + name + "\n Author : " +author
        + "\n Price : " + price + "\n num pages : " +
            num_pages ";
    }
}
public class Demo
{
    public static void main (String args[])
    {
        int n;
        Scanner sc = new Scanner (System.in);
        System.out.print("Enter number of books ");
        n= sc.nextInt();
        sc.nextLine(); // consumes newline
```

```
Book books[] = new Book[n];
for (int i=0; i<n; i++)
{
 System.out.println("Enter details of book"+(i+1));
 System.out.print("Name :");
 String name = sc.nextLine();
 System.out.print("Author name :");
 String author = sc.nextLine();
 System.out.print("Price :");
 double price = sc.nextDouble();
 System.out.print("No of pages :")
 int num_pages = sc.nextInt();
 sc.nextLine(); // consume new line
 books[i] = new Book(name, author, price, num_pages);
}
for (int i=0; i<n; i++)
{
  System.out.println("Book details are :");
  books[i].display();
  System.out.println("Using overrride ");
  System.out.println(books[i]);
}
}
}
```

OUTPUT

```
Enter number of books : 2

Enter details of book 1
Name : Java
Author name : author1
Price : 300
No of pages : 400

Enter details of book 2
Name : C programming
Author name : author2
price : 350
No of pages : 450

The book details are :
Book Name : Java
Author : author1
Price : 300.00
Number of pages : 400

Details using overriding toString()
Book Name : Java
Author : author1
Price : 300.00
Number of pages : 400.
```

The book details are

Book name : C programming

Author : author 2

Price : 350.00

Number of page : 400

Details using overriding to string().

Book name : C programming

Author : author 2

Price : 350.0

Number of page : 450.

PROGRAM -4

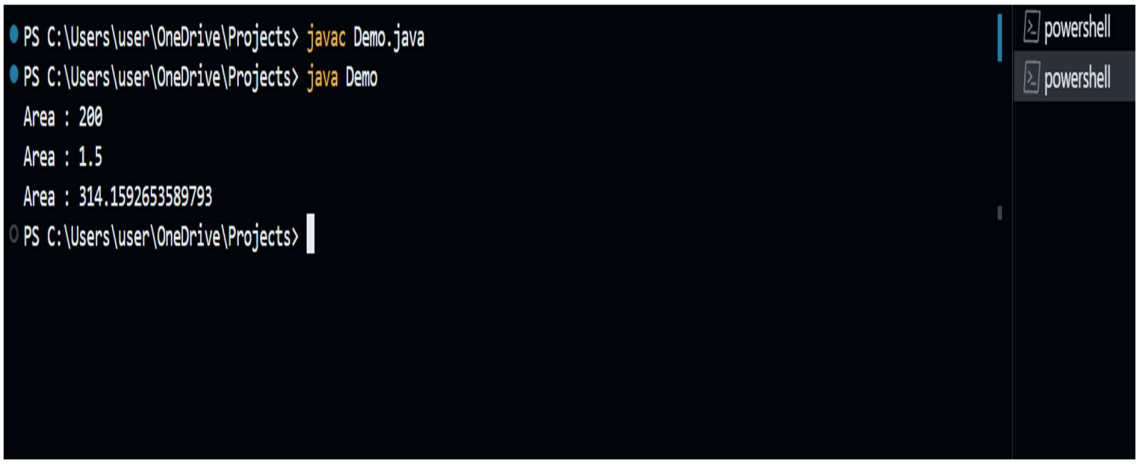
4)Develop a Java program to create an abstract class named Shape that contains two integers and an empty method named printArea( ). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape.Each one of the classes contain only the method printArea(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

```
import java.util.*;


abstract class Shape
{
   int s1, s2;


   public Shape(int a, int b)
   {
      this.s1 = a;
      this.s2 = b;
   }


   public Shape(int a)
   {
      this.s1 = a;
   }


   public abstract void printArea();
}


class Rectangle extends Shape
{
   public Rectangle(int x, int y)
   {
      super(x, y);
   }
```

```
    @Override
    public void printArea()
    {
        System.out.println("Area : " + (s1 * s2));
    }
}

class Triangle extends Shape
{
    public Triangle(int x, int y)
    {
        super(x, y);
    }

    @Override
    public void printArea()
    {
        System.out.println("Area : " + (0.5 * s1 * s2));
    }
}

class Circle extends Shape
{
    public Circle(int x)
    {
        super(x);
    }

    @Override
    public void printArea()
    {
```

```
        System.out.println("Area : " + (Math.PI * s1 * s1));
    }
}

public class Demo
{
    public static void main(String args[])
    {
        Shape obj1 = new Rectangle(10, 20);
        obj1.printArea();

        Shape obj2 = new Triangle(1, 3);
        obj2.printArea();

        Shape obj3 = new Circle(10);
        obj3.printArea();
    }
}
```

CODE:



Q) Java program of shapes

Code:

```
import java.util.*;
abstract class shape
{
    int s1, s2;
    public shape (int a, int b)
    {
        this.s1 = s1;
        this.s2 = s2;
    }
    public shape (int a)
    {
        this.s1 = a;
    }
    public abstract void Area();
}
class Rectangle extends Shape
{
    public Rectangle (int x, int y)
    {
        super (x, y);
    }
    public void Area()
    {
        System.out.println ("Area " + (s1 * s2));
    }
}
```

```
class Triangle extends Shape
{
    public Triangle (int x, int y)
    {
        super (x,y);
    }

    public void printArea()
    {
        Sout (" Area :" +(math.PI *s1*s2));
    }
}

class circle extends Shape
{
    public circle (int x)
    {
        super(x);
    }

    public void printArea()
    {
        System.out.println(" Area :"+ (math.PI.s1*s1));

    }
}
```

```
public class Demo
{
    public static void main (String a[])
    {
        Shape obj1 = new Rectangle (10,20);
        obj1. printArea();
        Shape obj2 = new Triangle (1,3);
        obj2. printArea();
        Shape obj3 = new Circle (10);
        obj3. printArea();
    }
}
```

PROGRAM -5

5)Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed. Create a class Account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks: a) Accept deposit from customer and update the balance. b) Display the balance. c) Compute and deposit interest d) Permit withdrawal and update the balance Check for the minimum balance, impose penalty if necessary and update the balance.

```
import java.util.Scanner;


abstract class Account
{
   String customername;
   int accno;
   double balance;


   public Account(String customername, int accno, double balance)
   {
      this.customername = customername;
      this.accno = accno;
      this.balance = balance;
   }


   public abstract void deposit(double amount);
   public abstract void withdraw(double amount);
   public abstract void displayBalance();


   public double getBalance()
   {
      return this.balance;
   }
```

```
}

class CurrAct extends Account
{
   double minbalance = 500;
   double servicecharge = 20;

   public CurrAct(String customername, int accno, double balance)
   {
      super(customername, accno, balance);
   }

   public void deposit(double amount)
   {
      balance += amount;
      System.out.println("Deposited: " + amount + " Rs");
   }

   public void withdraw(double amount)
   {
      if (balance - amount >= minbalance)
      {
         balance -= amount;
         System.out.println("Withdrawn: " + amount + " Rs");
      }
      else
      {
         balance -= servicecharge;
         System.out.println("Service charge applied");
      }
   }
```

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

class SavAcc extends Account
{
    double interest;

    public SavAcc(String customername, int accno, double balance, double interest)
    {
        super(customername, accno, balance);
        this.interest = interest;
    }

    public void deposit(double amount)
    {
        balance += amount;
        System.out.println("Deposited: " + amount);
    }

    public void withdraw(double amount)
    {
        if (balance - amount >= 0)
        {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        }
        else
```

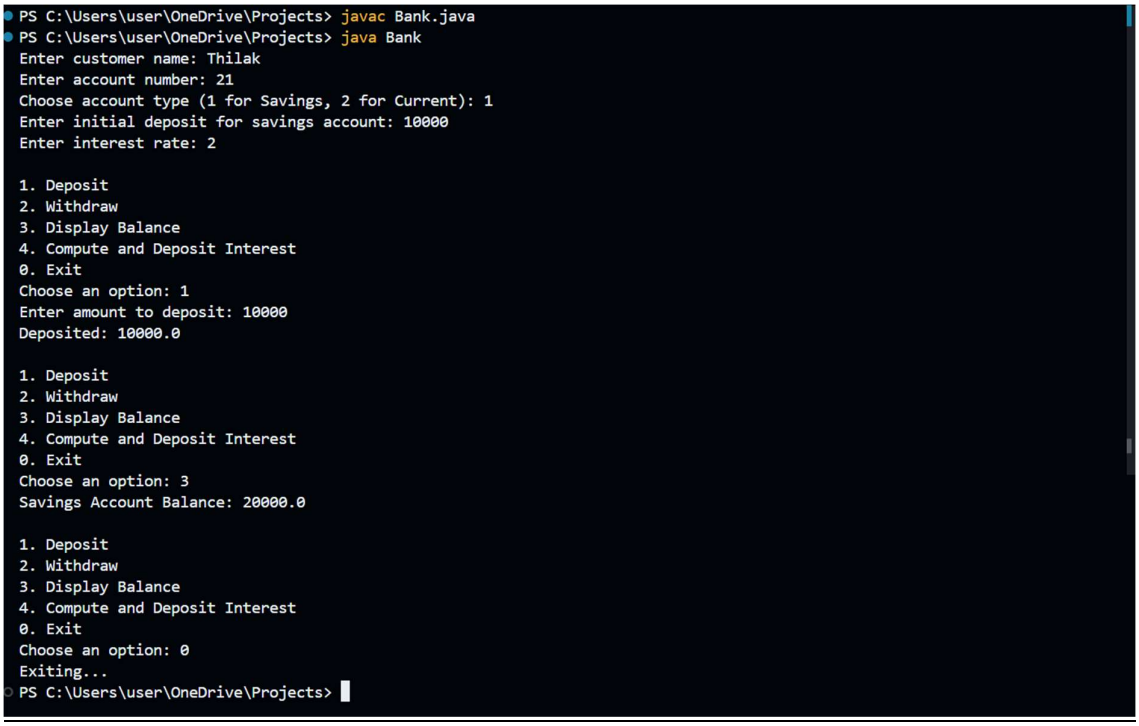
```
        {
            System.out.println("Insufficient balance for withdrawal.");
        }
    }

    public void computeAndDepositInterest()
    {
        double calculatedInterest = balance * this.interest / 100;
        balance += calculatedInterest;
        System.out.println("Interest deposited: " + calculatedInterest);
    }

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

public class Bank
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter customer name: ");
        String customerName = sc.nextLine();
        System.out.print("Enter account number: ");
        String accountNumber = sc.nextLine();
        System.out.print("Choose account type (1 for Savings, 2 for Current): ");
        int accountType = sc.nextInt();
        Account account = null;
```

```
if (accountType == 1)
{
   System.out.print("Enter initial deposit for savings account: ");
   double initialDeposit = sc.nextDouble();
   System.out.print("Enter interest rate: ");
   double interestRate = sc.nextDouble();
   int accno = Integer.parseInt(accountNumber);
   account = new SavAcc(customerName, accno, initialDeposit, interestRate);
}
else
{
   System.out.print("Enter the initial deposit: ");
   double initialDeposit = sc.nextDouble();
   int accno = Integer.parseInt(accountNumber);
   account = new CurrAct(customerName, accno, initialDeposit);
}

while (true)
{
   System.out.println("\n1. Deposit");
   System.out.println("2. Withdraw");
   System.out.println("3. Display Balance");
   if (account instanceof SavAcc)
   {
      System.out.println("4. Compute and Deposit Interest");
   }
   System.out.println("0. Exit");
   System.out.print("Choose an option: ");
   int option = sc.nextInt();

   switch (option)
```

```
{
    case 1:
        System.out.print("Enter amount to deposit: ");
        double depositAmount = sc.nextDouble();
        account.deposit(depositAmount);
        break;
    case 2:
        System.out.print("Enter amount to withdraw: ");
        double withdrawAmount = sc.nextDouble();
        account.withdraw(withdrawAmount);
        break;
    case 3:
        account.displayBalance();
        break;
    case 4:
        if (account instanceof SavAcc)
        {
            ((SavAcc) account).computeAndDepositInterest();
        }
        else
        {
            System.out.println("This option is not available for Current Accounts.");
        }
        break;
    case 0:
        System.out.println("Exiting...");
        sc.close();
        return;
    default:
        System.out.println("Invalid option. Please try again.");
}
```

```
    }
  }
}
```

OUTPUT:


PS C:\Users\user\OneDrive\Projects> javac Bank.java
PS C:\Users\user\OneDrive\Projects> java Bank
Enter customer name: Thilak
Enter account number: 21
Choose account type (1 for Savings, 2 for Current): 1
Enter initial deposit for savings account: 10000
Enter interest rate: 2
1. Deposit
2. Withdraw
3. Display Balance
4. Compute and Deposit Interest
0. Exit
Choose an option: 1
Enter amount to deposit: 10000
Deposited: 10000.0
1. Deposit
2. Withdraw
3. Display Balance
4. Compute and Deposit Interest
0. Exit
Choose an option: 3
Savings Account Balance: 20000.0
1. Deposit
2. Withdraw
3. Display Balance
4. Compute and Deposit Interest
0. Exit
Choose an option: 0
Exiting...
PS C:\Users\user\OneDrive\Projects>

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

Create the class

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

```
Import java.util.Scanner;
abstract class Account
{
    String customername;
    int accno;
    double balance;
```

```
public Account (String customername , int accno,
                double balance),
{
    this.customername = customername;
    this.accno = accno;
    this.balance = balance;
}
public abstract void deposit (double amount);
public abstract void deposit (double amount);
public abstract void displayBalance();

public double getBalance()
{
    return this.balance;
}

class currAct extends Amount
{
    double minbalance = 500;
    double servicecharge = 20;
    public CurrAct (String customername,
                    int accno, double balance)
    {
        super (customername, accno, balance);
    }
```

```
public void deposit (double Amount)
{
    balance += amount;
    Sout.("" Deposited :"+ amount+ "Rs");
}
public void withdraw (double amount)
{
    ib (balance - amount >= minbalance)
    {
        balance -= amount;
        Sout (" withdrawn :" amount);
    }
    else
    {
        balance -= service charge;
        Sout ("Service charge applied");
    }
}
public void displayBalance ()
{
    System.out.println (" current Balance : " + balance);
}
}
```

```
class SavAcc extends Account
{
    double interest;
    public SavAcc(String customername, int accno,
                  double balance, double interest)
    {
        super(customername, accno, balance);
        this.interest = interest;
    }

    public void deposit(double amount)
    {
        balance += amount;
        System.out.println("Deposited : " + amount);
    }
    public void withdraw(double amount)
    {
        if (balance - amount >= 0)
        {
            balance -= amount;
            Sout("withdrawn : " + amount);
        }
        else
        {
            System.out.println("Insufficient balance");
        }
}
```

```
public void computeAndDepositInterest()
{
    double calculatedInterest = balance * this.balance/100;
    balance += calculatedInterest;
    Sout (" Interest deposited " + calculatedInterest);
}
public void displayBalance()
{
    System.out.println("Savings Account balance: " + balance);
}
}
public class Bank
{
    public static void main (String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter customer name: ");
        String customerName = sc.nextLine();
        System.out.print("Enter account accno: ");
        String accno = sc.nextLine();
        System.out.print("Choose account type (1 for
                savings, 2 for current): ");
        int accountType = sc.nextInt();
        Account account = null;
```

```
if (accounType==1)
{
    System.out.print(" Enter inihial deposit :");
    double inihialDeposit = sc.nextDouble();
    System.out.print(" Enter interest rate : ");
    double InterestRate = sc.nextDouble();
    int accno = Integer.parseInt(accountNumber);
    account = new SavAcc(customername, accno, inihial
                         Depsit, interestRate);
}
else
{
    System.out.print(" Enter the inihial : ");
    double inihialDeposit = sc.nextDouble();
    int accno = Integer.parseInt(accountNumber);
    account = new CurrAcc(customerName, accno,
                          inihialDeposit);
}
while(true)
{
    System.out.println("1. Deposit ");
    System.out.println("2. Withdraw")
    System.out.println("3. Display Balance");
    if (account instanceof SavAcc)
    {
        System.out.println(" 4. Compute & deposit")
    }
```

PROGRAM -6

6. Create a package CIE which has two classes- Student and Internals. The class Student 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.

Internals.java

```
package CIE;


public class Internals {
    public int[] internalMarks;

    public Internals(int[] marks) {
        if (marks.length == 5) {
            this.internalMarks = marks;
        } else {
            throw new IllegalArgumentException("Marks array must have exactly 5 elements.");
        }
    }
}
```

Personal.java

```
package CIE;


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

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

```
        this.sem = sem;
    }
}
```

External.java

```
package SEE;

import CIE.Personal;
import CIE.Internals;

public class External extends Personal
{
    public Internals internalMarks;
    public int[] externalMarks;

    public External(String usn, String name, int sem, int[] internalMarks, int[] externalMarks)
    {
        super(usn, name, sem);
        this.internalMarks = new Internals(internalMarks);
        if (externalMarks.length == 5)
        {
            this.externalMarks = externalMarks;
        }
        else
        {
            throw new IllegalArgumentException("Marks array must have exactly 5 elements.");
        }
    }

    public int[] calculateFinalMarks()
    {
        int[] finalMarks = new int[5];
```

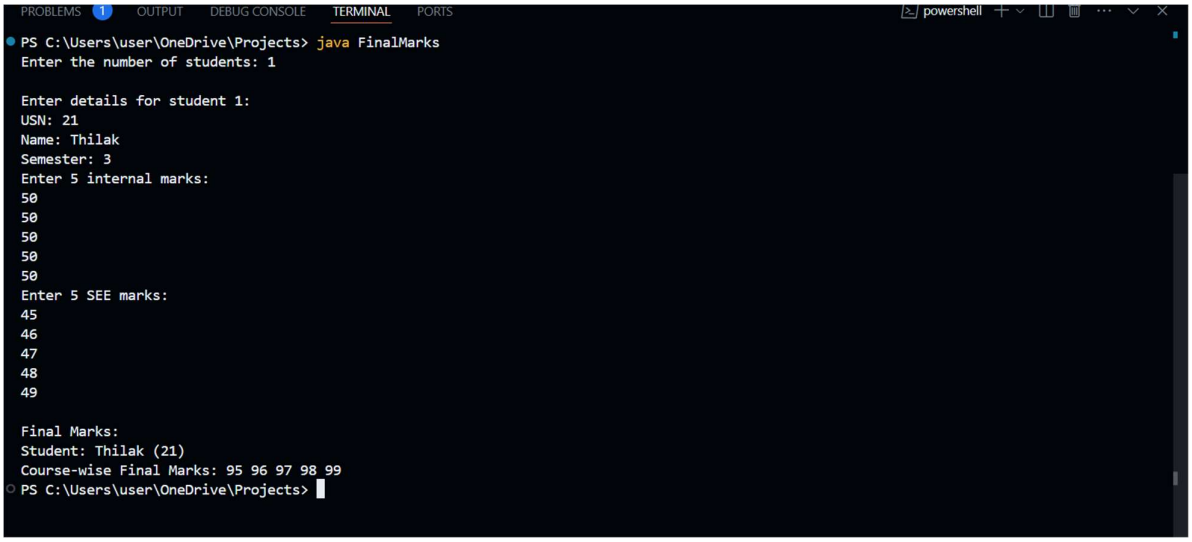
```
        for (int i = 0; i < 5; i++)
        {
            finalMarks[i] = this.internalMarks.internalMarks[i] + this.externalMarks[i];
        }
        return finalMarks;
    }
}
```

FinalMarks.java

```
import CIE.*;
import SEE.*;

import java.util.Scanner;

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];

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

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

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

System.out.println("\nFinal Marks:");
for (External student : students)
{
    System.out.println("Student: " + student.name + " (" + student.usn + ")");
    System.out.print("Course-wise Final Marks: ");
    int[] finalMarks = student.calculateFinalMarks();
    for (int mark : finalMarks)
    {
        System.out.print(mark + " ");
    }
```

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


        scanner.close();
    }
}
```



8) Create a package CIE which has two classes personal and Internals.

The class personal has members usn, name and sem.

The class Internals has an array that stores internal marks scored in five courses of current semester of the student.

Create another package SEE which has one class External derived from Personal.

This class has an array that stores the SEE marks scored five courses of the current semester of the student.

Package CIE

Internals.java

```
package CIE;
public class Internals
{
  public int [] internalmarks
  {
    public Internals [ int[] marks)
    {
      if (marks.length == 5)
      {
        this.internal marks = marks;
      }
  }
```

```
        {
            throw new IllegalArgumentException(" marks
                    array must have 5 elements");
        }
    }
}
```

Personal.java

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

PACKAGE SEE

```
package SEE;
import CIE.Personal;
import CIE.Internals;
public class External extends Personal
{
```

```
public Internals internalmarks;
public int[] externalmarks;

public External(String usn, String name, int sem,
                int[] internalmarks, int[] externalmarks)

{
    super(usn, name, sem);
    this.internalmarks = new Internals(internalmarks);
    if (externalmarks.length == 5)
    {
        this.externalmarks = externalmarks;
    }
    else
    {
        throw new IllegalArgumentException("5 elements only");
    }
}

public int[] calculateFinalMarks()
{
    int[] finalmarks = new int[5];
    for (int i=0; i<5; i++)
    {
        finalmarks[i] = this.internalmark.internalmarks[i]
                        + this.externalmarks[i];
    }
    return finalmarks;
}
```

Main class

```
import CIE.*;
import SEE.*;
import java.util.Scanner;
public class FinalMarks
{
  public static void main(String args[])
  {
    Scanner scanner= new Scanner(System.in);
    System.out.print(" Enter no of students
    int n= scanner.nextInt();
    External[] students= new External[n];
    for (int i=0; i<n; i++)
    {
      System.out.println("Enter details of student");
      System.out.print(" USN: ");
      String usn = scanner.usn();
      System.out.print(" Name: ");
      String name= scanner.next();
      System.out.print("Semester");
      int sem = scanner.nextInt();
      int internalmarks[]= new int[5];
      System.out.println(" Enter 5 internal
                          marks)
```

```
for( int i=0 ; i<5 ; i++)
{
    internalmarks [i] =  scanner.nextInt();
}
sout (" Enter  5  SEE  marks");
for (int i=0 ; i<5 ; i++)
{
    external[i]=  sc.nextInt();
}
students[i]= new External( usn, name, sem, internal,
                                          External);

System.out. println(" Final marks ");
for ( External student : students )
{
  System.out. println(" student : " + student.name);
  System.out. println(" Course wise final marks");
  int [] final = student. calculate Final marks ();
   for (int mark : finalmarks)
   {
        System.out. print ( mark);
   }
   System.out. println();
}
}
```

PROGRAM -7

7)Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called "Father" and derived class called "Son" which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age=father's age

```
import java.util.Scanner;


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


class Father
{
   int age;


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


class Son extends Father
{
```

```
    int sonAge;

    Son(int fatherAge, int sonAge) throws Main
    {
        super(fatherAge);
        if (sonAge >= fatherAge)
        {
            throw new Main("WrongAge: Son's age cannot be greater than or equal to Father's age");
        }
        this.sonAge = sonAge;
    }
}

public class Main
{
    public static void main(String[] args)
    {
        try
        {
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter Father's age: ");
            int fatherAge = sc.nextInt();
            System.out.print("Enter Son's age: ");
            int sonAge = sc.nextInt();
            Son obj = new Son(fatherAge, sonAge);
            System.out.println("Father's age: " + fatherAge + ", Son's age: " + sonAge);
        }
        catch (Main e)
        {
            System.out.println(e.getMessage());
        }
```

```
    }
}
```

OUTPUT:

```
PS C:\Users\user\OneDrive\Projects> java Mainn
Enter Father's age: 50
Enter Son's age: 60
WrongAge: Son's age cannot be greater than or equal to Father's age
PS C:\Users\user\OneDrive\Projects>
```

Q) Write a program that demonstrate handling of exceptions in inheritance tree. Create a base class Father and derived class called Son which extends base class. In Father class implement a constructor which take the age and throws the exception wrongage()

```
import java.util.Scanner;
class main extends Exception
{
    Main (String message)
    {
        super(message);
    }
}

class Father {
    int age;
    if (age<0
    Father (int age) throws main {
        if (age<0)
        {
            throw new Main ("Error");
        }
        this.age=age;
    }
```

```
class Son extends Father
{
    int sonAge;
    Son (int fatherAge, int sonAge) throws main
    {
        super (fatherAge);
        if (sonAge >= fatherAge)
            throw new main ("Error");
        this.sonAge = sonAge;
    }
}
public class main
{
    public static void main (String args[])
    {
        try
        {
            Scanner sc = new Scanner (System.in);
            Sout ("Enter father age");
            int fatherAge = sc.nextInt();
            Sout ("Son's age");
            int sonAge = sc.nextInt();
            Son obj = new Son (fatherage, sonAge);
        }
        catch (main e)
        {
            Sout (e.System.out.getMessage());
        }
    }
```

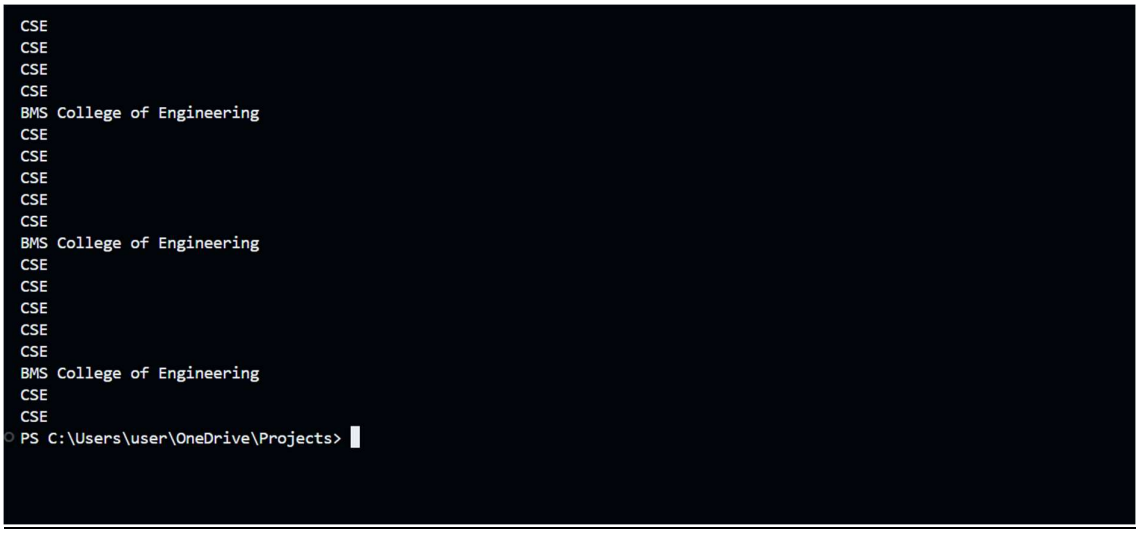
PROGRAM -8

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

```
class DisplayBMS extends Thread
{
   public void run()
   {
      try
      {
         while (true)
         {
            System.out.println("BMS College of Engineering");
            Thread.sleep(10000);
         }
      }
      catch (InterruptedException e)
      {
         e.printStackTrace();
      }
   }
}


class DisplayCSE extends Thread
{
   public void run()
   {
      try
      {
         while (true)
         {
            System.out.println("CSE");
```

```
            Thread.sleep(2000);
          }
        }
        catch (InterruptedException e)
        {
          e.printStackTrace();
        }
    }
}

public class Main
{
    public static void main(String[] args)
    {
        DisplayBMS thread1 = new DisplayBMS();
        DisplayCSE thread2 = new DisplayCSE();
        thread1.start();
        thread2.start();
    }
}
```

OUTPUT:



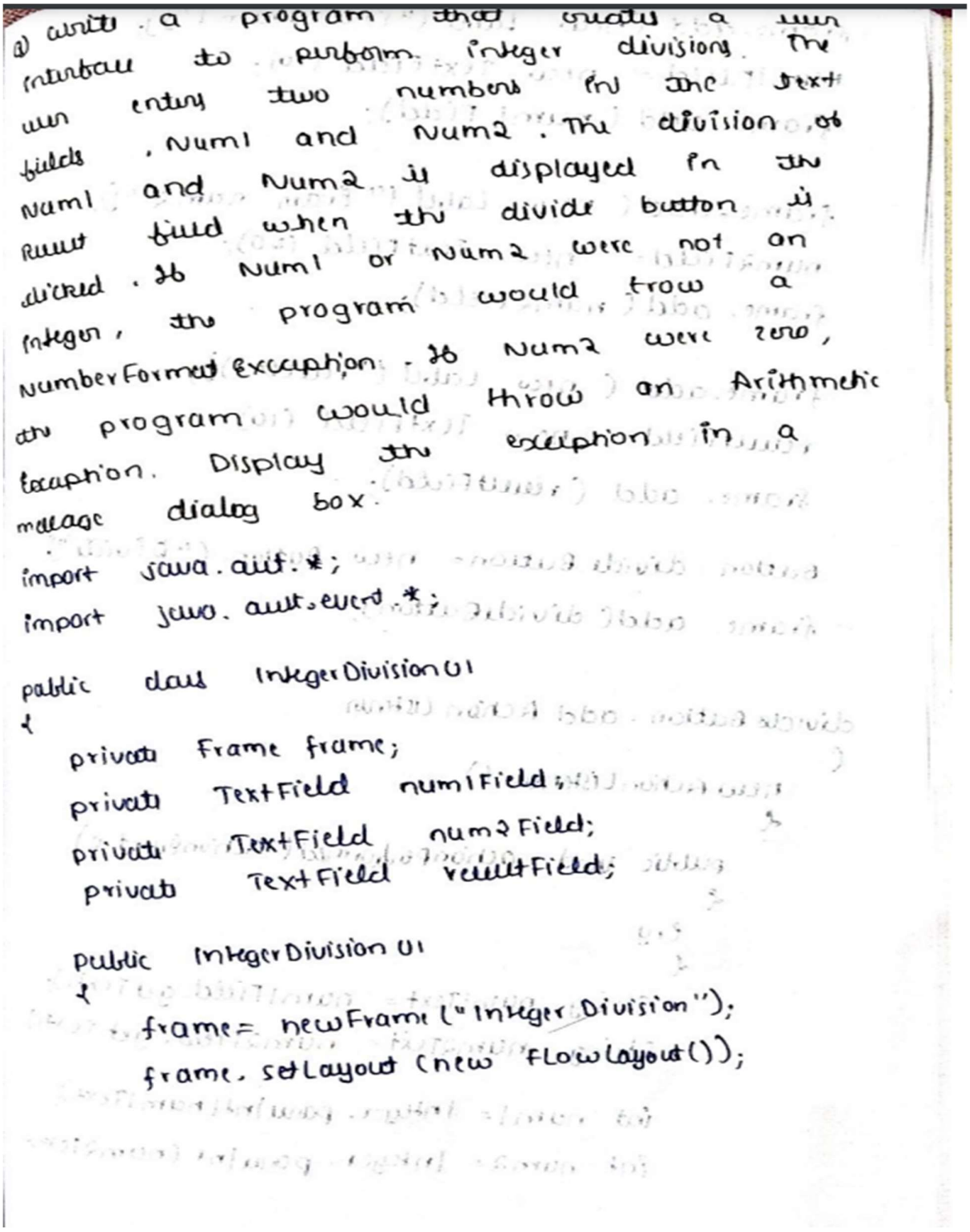
② Write a java program to create two threads one displaying "BMS College of Engineering" one every 10 seconds and another displaying CSE every 2s

```
class DisplayBMS extends Thread
{
    Public void run()
    {
        Try
        {
            while(true)
            {
                Sout("BMS college");
                Thread.sleep(10000);
            }
        }
        Catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

```
class DisplayCSE extends Thread
{
    public void run()
    {
        Try
        {
            while(true)
            {
                Sout("CSE");
                Thread sleep(2000);
            }
        }
        Catch(Exception e)
        {
            Sout(e.printStackTrace);
        }
    }
}

public class Main
{
    psum(String args[])
    {
        DisplayBMS thread1 = new DisplayBMS();
        DisplayCSE thread2 = new DisplayCSE();
        Thread1.start()
        thread2.start()
    }
}
```

PROGRAM -9

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



```
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 javax.swing.event.*;

public class IntegerDivisionUI
{
    private Frame frame;
    private TextField num1Field;
    private TextField num2Field;
    private TextField resultField;

    public IntegerDivisionUI
    {
        frame= new Frame("Integer Division");
        frame.setLayout(new FlowLayout());
```

```
frame.add (new Label ("Enter Num 1"));
num1Field = new TextField (10);
frame. add (num1 Field);

frame.add ( new Label ("Enter num 2"));
num2Field = new TextField (20);
frame. add ( num2Field);

frame.add ( new Label ("result"));
resultField = new TextField (10);
frame. add (resultField);

Button divideButton = new Button ("Divide");
frame. add( divideButton);

divideButton . addActionListener
(
   new ActionListener ()
   {
      public void actionPerformed (ActionEvent e)
      {
        try
        {
          String num1Text = num1Field.getText();
          String num2Text = num2Field.getText();

          int num1 = Integer. parseInt(num1Text);
          int num2 = Integer. parseInt (num2Text);
```

```
int result = num1/num2;
resultField.setText(String.valueOf(result));
}
catch(ArithmeticException ex)
{
showErrorDialog("Enter integers");
}
catch (ArithmeticException ex)
{
showErrorDialog("Can't divide by zero");
}
}
});
frame.setSize(300,200);
frame.setVisible(true);
frame.addWindowListener(new WindowAdapter()
{
public void windowClosing(WindowEvent we)
{
System.exit(0);
}
});
}
```

```
private void showErrorDialog(String message)
{
    Dialog errorDialog = new Dialog(frame, "Error", hu
    errorDialog.setSize(250,100);
    errorDialog.setLayout(new FlowLayout());

    Label errorLabel = new Label(message);
    errorDialog.add(errorLabel);

    Button okButton = new Button("Ok");
    okButton.addActionListener(new ActionListner()
      {
        public void ActionPerformed(ActionEvent e)
        {
            errorDialog.setVisible(false);
        }
      });
    errorDialog.add(okButton);
    errorDialog.setVisible(true);
}
public static void main(String args[]);
{
    new IntegerDivisionUI();
}
```

CODE:

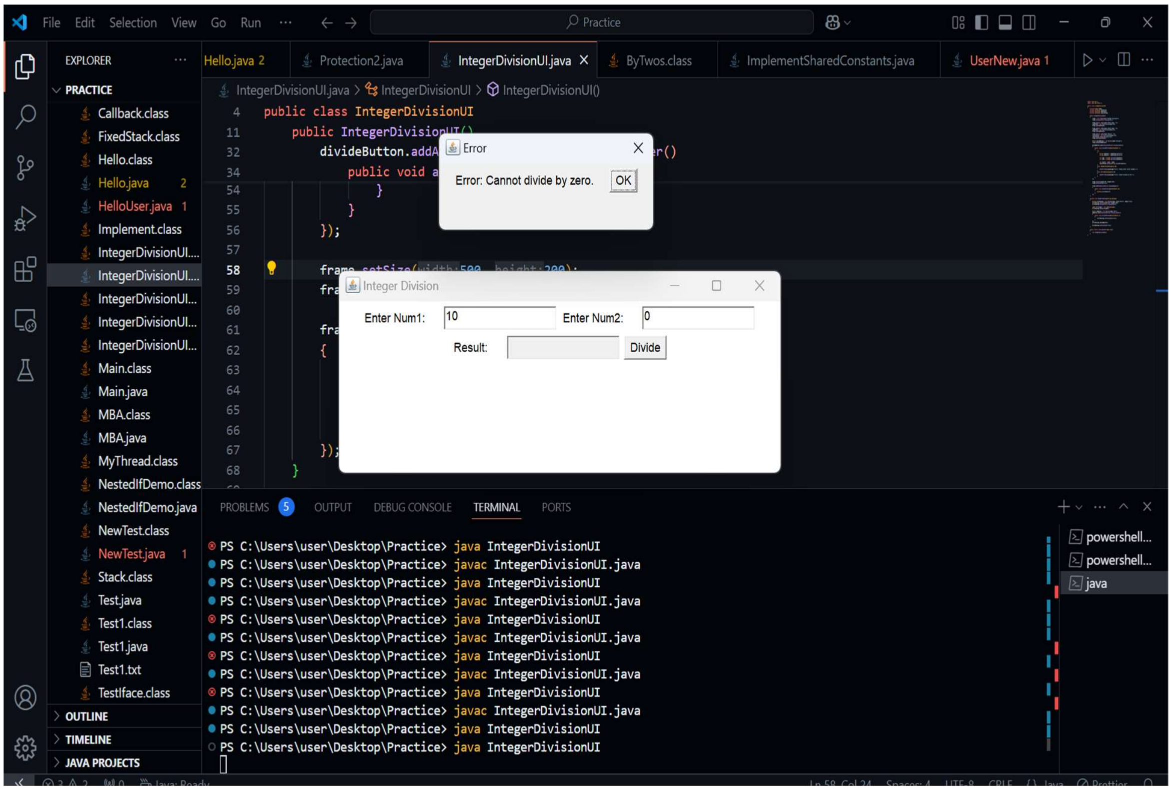
```
import java.awt.*;
import java.awt.event.*;

public class IntegerDivisionUI
{
   private Frame frame;
   private TextField num1Field;
   private TextField num2Field;
   private TextField resultField;

   public IntegerDivisionUI()
   {
      frame = new Frame("Integer Division");
      frame.setLayout(new FlowLayout());

      frame.add(new Label("Enter Num1: "));
      num1Field = new TextField(10);
      frame.add(num1Field);

      frame.add(new Label("Enter Num2: "));
      num2Field = new TextField(10);
      frame.add(num2Field);

      frame.add(new Label("Result: "));
      resultField = new TextField(10);
      resultField.setEditable(false);
      frame.add(resultField);

      Button divideButton = new Button("Divide");
      frame.add(divideButton);
```

```
divideButton.addActionListener(new ActionListener()
{
   public void actionPerformed(ActionEvent e)
   {
      try
      {
         String num1Text = num1Field.getText();
         String num2Text = num2Field.getText();

         int num1 = Integer.parseInt(num1Text);
         int num2 = Integer.parseInt(num2Text);

         int result = num1 / num2;
         resultField.setText(String.valueOf(result));
      }
      catch (NumberFormatException ex)
      {
         showErrorDialog("Error: Please enter valid integers.");
      }
      catch (ArithmeticException ex)
      {
         showErrorDialog("Error: Cannot divide by zero.");
      }
   }
});

frame.setSize(300, 200);
frame.setVisible(true);

frame.addWindowListener(new WindowAdapter()
```

```
    {
        public void windowClosing(WindowEvent we)
        {
            System.exit(0);
        }
    });
}

private void showErrorDialog(String message)
{
    Dialog errorDialog = new Dialog(frame, "Error", true);
    errorDialog.setSize(250, 100);
    errorDialog.setLayout(new FlowLayout());

    Label errorLabel = new Label(message);
    errorDialog.add(errorLabel);

    Button okButton = new Button("OK");
    okButton.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            errorDialog.setVisible(false);
        }
    });
    errorDialog.add(okButton);

    errorDialog.setVisible(true);
}

public static void main(String[] args)
```

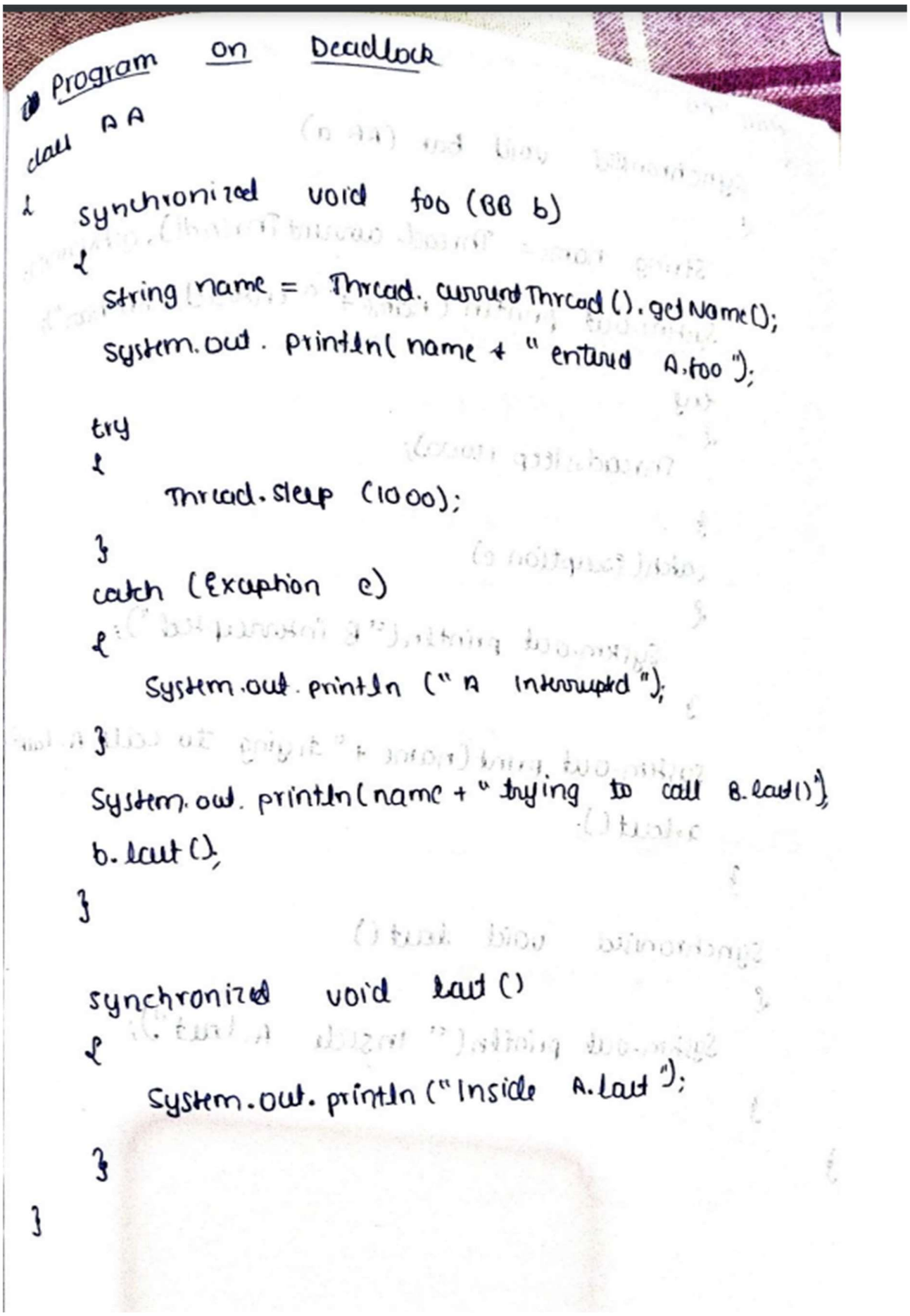
```
    {
        new IntegerDivisionUI();
    }
}
```

OUTPUT:



PROGRAM -10

10. Demonstrate Inter process Communication and deadlock



```
// Program on 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 ");
    }
}
```

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

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

```
public class Deadlock implements Runnable
{
    AA a = new AA();
    BB b = new BB();

    Deadlock()
    {
     Thread.currentThread().setName("MainThread");
     Thread t = new Thread(this, "Racing Thread");
     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);
        System.out.println(" Back in other thread");
    }

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

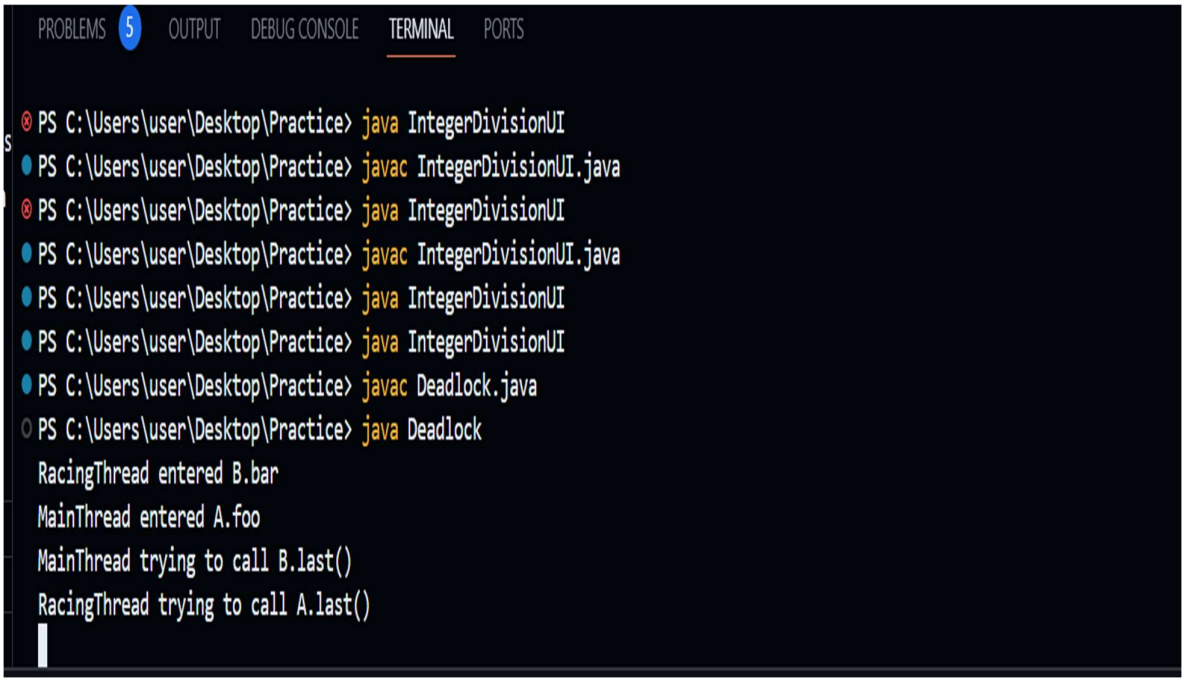
CODE FOR DEADLOCK:

```
class AA
{
   synchronized void foo(BB 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 BB
{
   synchronized void bar(AA 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");
    }
}

public class Deadlock implements Runnable
{
    AA a = new AA();
    BB b = new BB();

    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:



INTERTHREAD COMMUNICATION:

```
Q) Demonstrate Interprocess communication.

class Q
{
    int n;
    boolean valueset = false;

    synchronized int get()
    {
        while (!valueset)
        {
            try
            {
                wait();
            }
            catch (InterruptedException e)
            {
                System.out.println("Interrupted");
            }
        }
        System.out.println("Got :" + n);
        valueset = false;
        notify();
        return n;

    synchronized void put(int n)
    {
        while (valueset)
        {
            try
            {
                wait();
```

```
        }
        catch (InterruptedException e)
        {
            System.out.println("InterruptedException caught");
        }
    }
    this.n = n;
    valueSet = true;
    System.out.println("Put: " + n);
    notify();
  }
}

class Producer implements Runnable
{
    Q q;
    Producer(Q q)
    {
        this.q = q;
        new Thread(this, "Producer").start();
    }

    public void run()
    {
        int i = 0;
        while (true)
        {
            q.put(i++);
        }
    }
}
```

```
class Consumer implements Runnable
{
    Q q;
    Consumer (Q q)
    {
        this.q = q;
        new Thread (this, "consumer").start();
    }

    public void run()
    {
        while (true);
        {
            q.get();
        }
    }
}
class PCFixed
{
    public static void main (String args[])
    {
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Press control c:");
    }
}
```

CODE:

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

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

   synchronized void put(int n)
   {
      while (valueSet)
      {
         try
         {
```

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

class Producer implements Runnable
{
    Q q;

    Producer(Q q)
    {
        this.q = q;
        new Thread(this, "Producer").start();
    }

    public void run()
    {
        int i = 0;
        while (true)
        {
            q.put(i++);
        }
```

```
    }
}

class Consumer implements Runnable
{
    Q q;

    Consumer(Q q)
    {
        this.q = q;
        new Thread(this, "Consumer").start();
    }

    public void run()
    {
        while (true)
        {
            q.get();
        }
    }
}

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:


Got: 20719
Put: 20720
Got: 20720
Put: 20721
Got: 20721
Put: 20722
Got: 20722
Put: 20723
Got: 20723
Put: 20724
Got: 20724
Put: 20725
Got: 20725
Put: 20726
Got: 20726
Put: 20727
Got: 20727
Put: 20728
Got: 20728
Put: 20729
Got: 20729
Put: 20730
Got: 20730
Put: 20731
Got: 20731
Put: 20732
Got: 20732
Put: 20733
Got: 20733
Put: 20734
Got: 20734
Put: 20735
PS C:\Users\user\OneDrive\Projects>
powershell
powershell
Run: Main