VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

OBJECT ORIENTED JAVA PROGRAMMING(23CS3PCOOJ)

Submitted by

SHREYAS T S(1BM23CS322)

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 **SHREYAS T S (1BM23CS322),** 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. Nandhini Vineeth Associate Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

LAB 1

Develop a Java program that prints all real solutions to thequadratic 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 QuadraticEquationSolver {
   public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the co-efficient of a: ");
        double a = scanner.nextDouble();
        System.out.print("Enter the co-efficient of b: ");
        double b = scanner.nextDouble();
        System.out.print("Enter the co-efficient of c: ");
        double c = scanner.nextDouble();
        double D = b * b - 4 * a * c;
        if (D > 0) {
            double root1 = (-b + Math.sqrt(D)) / (2 * a);
            double root2 = (-b - Math.sqrt(D)) / (2 * a);
            System.out.println("The equation has two real solutions:");
            System.out.println("Root 1: " + root1);
            System.out.println("Root 2: " + root2);
        } else if (D == 0) {
            double root = -b / (2 * a);
            System.out.println("The equation has one real solution: " + root);
        } else {
            System.out.println("The equation has no real solutions.");
        }
        scanner.close();
    }
}
```

```
C:\Users\shrey\OneDrive\Desktop\java programs>javac QuadraticEquationSolver.java

C:\Users\shrey\OneDrive\Desktop\java programs>java QuadraticEquationSolver
Enter the co-efficient of a: 2
Enter the co-efficient of b: 5
Enter the co-efficient of c: 3
The equation has two real solutions:
Root 1: -1.0
Root 2: -1.5

C:\Users\shrey\OneDrive\Desktop\java programs>java QuadraticEquationSolver
Enter the co-efficient of a: 2
Enter the co-efficient of b: 4
Enter the co-efficient of c: 2
The equation has one real solution: -1.0

C:\Users\shrey\OneDrive\Desktop\java programs>java QuadraticEquationSolver
Enter the co-efficient of a: 4
Enter the co-efficient of b: 3
Enter the co-efficient of c: 6
The equation has no real solutions.

C:\Users\shrey\OneDrive\Desktop\java programs>
```

Date
Page

```
import java.util.Scanner;

class QuadraticSolver{
public static void main(String[] args)
{ Scanner sc = new Scanner(System.in);
  System.out.print("Enter co-efficient a: ");
  double a = sc.nextDouble();
  System.out.print("Enter co-efficient b: ");
  double b = sc.nextDouble();
  System.out.print("Enter co-efficient c: ");
  double c = sc.nextDouble();
  double D = (b*b) - (4*a*c);
  if (D>0){
    double root1 = (-b + Math.sqrt(D)) / (2*a);
    double root2 = (-b - Math.sqrt(D)) / (2*a);
    System.out.println("The eqn has two real and
    distinct roots: ");
    System.out.println("Root 1: " + root1);
    System.out.println("Root 2: " + root2); }
  else if (D==0){
    double root = -b / (2*a);
    System.out.println("The eqn has one real root: ");
    System.out.println("Root: " + root); }
  else{
    System.out.println("The equation has no real solution,
    discriminant is negative");
  }
}
```

2) Develop a java program that prints all real solutions to quadratic eqn $ax^2+bx+c=0$. Read a,b,c and use the quadratic formula if the discriminant b^2-4ac is negative, display a message stating that there are no real solutions.

output:

enter co-efficient a:2

enter co-efficient b:5

enter co-efficient c:3

the equation has two real and distinct roots:

Root1: -1.0

Root2: -1.5

enter co-efficient a:2

enter co-efficient b:4

enter co-efficient c:2

the equation has one real root:

Root: -1.0

enter co-efficient a:4

enter co-efficient a:4

enter co-efficient b:3

enter co-efficient c:6

the equat" has no real and D is negative.

24/10/24

LAB 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.Include all the 8 programs as instructed in the classroom.

```
import java.util.Scanner;

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

  void acceptDetails() {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter USN: ");
      usn = sc.nextLine();
      System.out.print("Enter Name: ");
      name = sc.nextLine();
      System.out.print("Enter number of subjects: ");
      numSubjects = sc.nextInt();
      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) + ": ");
          marks[i] = sc.nextInt();
      }
  }

  void displayDetails() {
      System.out.println("\nStudent Details:");
      System.out.println("USN: " + usn);
      System.out.println("Name: " + name);
      System.out.println("Subject-wise details:");
      for (int i = 0; i < numSubjects; i++) {
          System.out.println("Subject " + (i + 1) + ": Credits = " + credits[i] + ", Marks = " + marks[i]);
      }
  }
```

```
    double calculateSGPA() {
        int totalCredits = 0;
        double totalGradePoints = 0;
        for (int i = 0; i < numSubjects; i++) {
            int gradePoint = getGradePoint(marks[i]);
            totalGradePoints += gradePoint * credits[i];
            totalCredits += credits[i];
        }
        return totalGradePoints / totalCredits;
    }
    int getGradePoint(int marks) {
        if (marks >= 90) return 10;
        if (marks >= 80) return 9;
        if (marks >= 70) return 8;
        if (marks >= 60) return 7;
        if (marks >= 50) return 6;
        if (marks >= 40) return 5;
        return 0;
     }
}
public class SGPA_Calculator {
    public static void main(String[] args) {
        Student student = new Student();
        student.acceptDetails();
        student.displayDetails();
        double sgpa = student.calculateSGPA();
        System.out.printf("SGPA:%.2f", sgpa);
    }
}
```

```
C:\Users\shrey\OneDrive\Desktop\java programs>java SGPA_Calculator
Enter USN: 1BM23CS322
Enter Name: shreyas
Enter number of subjects: 3
Enter credits for subject 1: 4
Enter marks for subject 1: 56
Enter credits for subject 2: 4
Enter marks for subject 2: 89
Enter credits for subject 3: 4
Enter marks for subject 3: 97

Student Details:
USN: 1BM23CS322
Name: shreyas
Subject-wise details:
Subject 1: Credits = 4, Marks = 56
Subject 2: Credits = 4, Marks = 89
Subject 3: Credits = 4, Marks = 97
SGPA:8.33
C:\Users\shrey\OneDrive\Desktop\java programs>
```

Date / /
Page

02 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 Student {
    String usn;
    String name;
    int numSubjects;
    int[] Credits;
    int[] marks;

    void studentdetails(){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter USN:");
        usn = sc.nextLine();
        System.out.print("Enter Name:");
        name = sc.nextLine();
        System.out.print("Enter number of subjects");
        numSubjects = sc.nextInt();
        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)+
            ":");
            marks[i] = sc.nextInt();
        }
    }
    void display(){
        System.out.println("usn:"+usn);
```

Date / /
Page

System.out.println("Name:" + name);
System.out.println("Subject-wise details:");
for (int i=0; i<numSubjects; i++){
System.out.println("Subject" + (i+1) + ": Credits=" +
credits[i] + " marks:" + marks[i]);
}

double calcSGPA(){
int total_credits=0
int grade_point=
for (int i=0; i<numSubject; i++)
{ totalcredits += credits[i];
if (marks[i]>=90);
totalcredits += 10* credits[i];
if (marks[i]>=80);
totalcredits += 9* credits[i];
if (marks[i]>=70);
total credits += 8*(credits[i]);
if (marks[i]>=60);
total credits += 7* credits[i];
if (marks[i]>=50);
totalcredits += 6* (credits[i]);
if (marks[i]>=40);
totalcredits += 5* credits[i];
else
totalcredits += 0* credits[i];
return totalcredits;

class SGPA_Calc {
public static void main(String[] args)
{ Student student = new Student();
student.studentdetails();
student.display();
Student.SGPA();
double sgpa = student.SGPA();
System.out.printf("SGPA" + sgpa);

Page

output:

Enter USN: 1BM23CS388

Enter Name: Shreyas

Enter number of subject: 3

Enter Credits for subject 1: 4

Enter marks of subject 1: 86

Enter Credits for subject 2: 4

Enter marks for subject 2: 89

Enter Credits for subject 3: 4

Enter marks for subject 3: 99

Student details:

USN: 1BM23CS388

Name: Shreyas

Subject wise details:

Subject 1: Credit=4, Marks=86

Subject 2: Credit=4, Marks=89

Subject 3: Credit=4, Marks=99

SGPA: 8.33

LAB 3

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

```
import java.util.Scanner;
class Book{
 private String name;
 private String author;
 private double price;
 private int num_pages;

 public Book(String name,String author,double price,int num_pages){
  this.name=name;
  this.author=author;
  this.price=price;
  this.num_pages=num_pages;}
 public String getName(){
  return name;}
 public void setName(String name){
  this.name=name;}
 public String getauthor(){
  return  author;}
 public void setauthor(String author){
  this.author=author;}
 public double getprice(){
  return price;}
 public void setprice(){
  this.price=price;}
 public int getnumpages(){
  return num_pages;}
 public void setnumpages(){
  this.num_pages=num_pages;}

 public String toString(){
  return("Book[name:"+name+" author:"+author+" price:"+price+" pages:"+num_pages);}
}
```

```
public class Bookmain{
 public static void main(String[] args){
  Scanner sc=new Scanner(System.in);
  System.out.println("enter no of books");
  int n=sc.nextInt();
  sc.nextLine();
  Book[] books=new Book[n];
  for(int i=0;i<n;i++){
   System.out.println("enter the details of book"+(i+1)+":");
   System.out.print("enter name:");
   String name=sc.nextLine();
   System.out.print("enter author:");
   String author=sc.nextLine();
   System.out.print("enter price:");
   int price=sc.nextInt();
   System.out.print("enter the no of pages:");
   int num_pages=sc.nextInt();
   books[i]=new Book(name,author,price,num_pages);}
  System.out.print("\ndetails of books");
  for(int i=0;i<n;i++){
   System.out.println(books[i].toString());
  }
}
}
```

```
C:\Users\shrey\OneDrive\Desktop\java programs>javac Bookmain.java

C:\Users\shrey\OneDrive\Desktop\java programs>java Bookmain
enter no of books
2
enter the details of book1:
enter name:
java programming
enter author:james gostling
enter price:450
enter the no of pages:350
enter the details of book2:
enter name:python basics
enter author:guido van rosum
enter price:350
enter the no of pages:450

details of booksBook[name: author:james gostling price:450.0 pages:350]
Book[name: author:guido van rosum price:350.0 pages:450]

C:\Users\shrey\OneDrive\Desktop\java programs>
```

Date / /
Page

Q) Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values of the members, include methods to set and get the details of object. include a toString() method that could display the complete details of the book. Develop a java program to create n books objects.

=>

```
import java.util.Scanner;

class Book{
    String name;
    String author;
    double price;
    int num_pages;

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

void setName(String name){
    this.name = name; }
String getAuthor(){
    return author; }
void setAuthor(String author){
    this.author = author; }
String getName(){
    return name; }
double getPrice(){
    return price;
```

```
void setPrice(double price){
        this.price = price; }
int getNumPages(){
        return num_pages; }
void setNumPages(int num_pages){
        this.num_pages = num_pages; }

String toString(){
    return "Book[Name: " + name + ", Author: " + author + ",
    price: $" + price + ", pages: " + num_pages + "]";
    }
}

Public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the no of book");
        int n = sc.nextInt();
        sc.nextLine;
        Book[] books = new Book[n];
        for(int i=0; i<n; i++){
        System.out.println("\nEnter details for Book" + (i+1)
        + ":");
        System.out.print("Enter name:");
        String name = sc.nextLine();
        System.out.print("Enter author:");
        String author = sc.nextLine();
        System.out.print("Enter price:");
        double price = sc.nextDouble();
        System.out.print("Enter number of pages");
        int num_pages = sc.nextInt();
        book[i] = new Book(name, author, price,
                                num_pages);}
```

```
System.out.println("\n Details of Books:");
for (int i=0; i<n; i++){
System.out.println(books[i].toString());
}
}
}
```

output

Enter the number of book: 2

Enter details of Book 1:
Enter name: Java programming
Enter author: James Gosling
Enter price: 450
Enter number of pages: 350

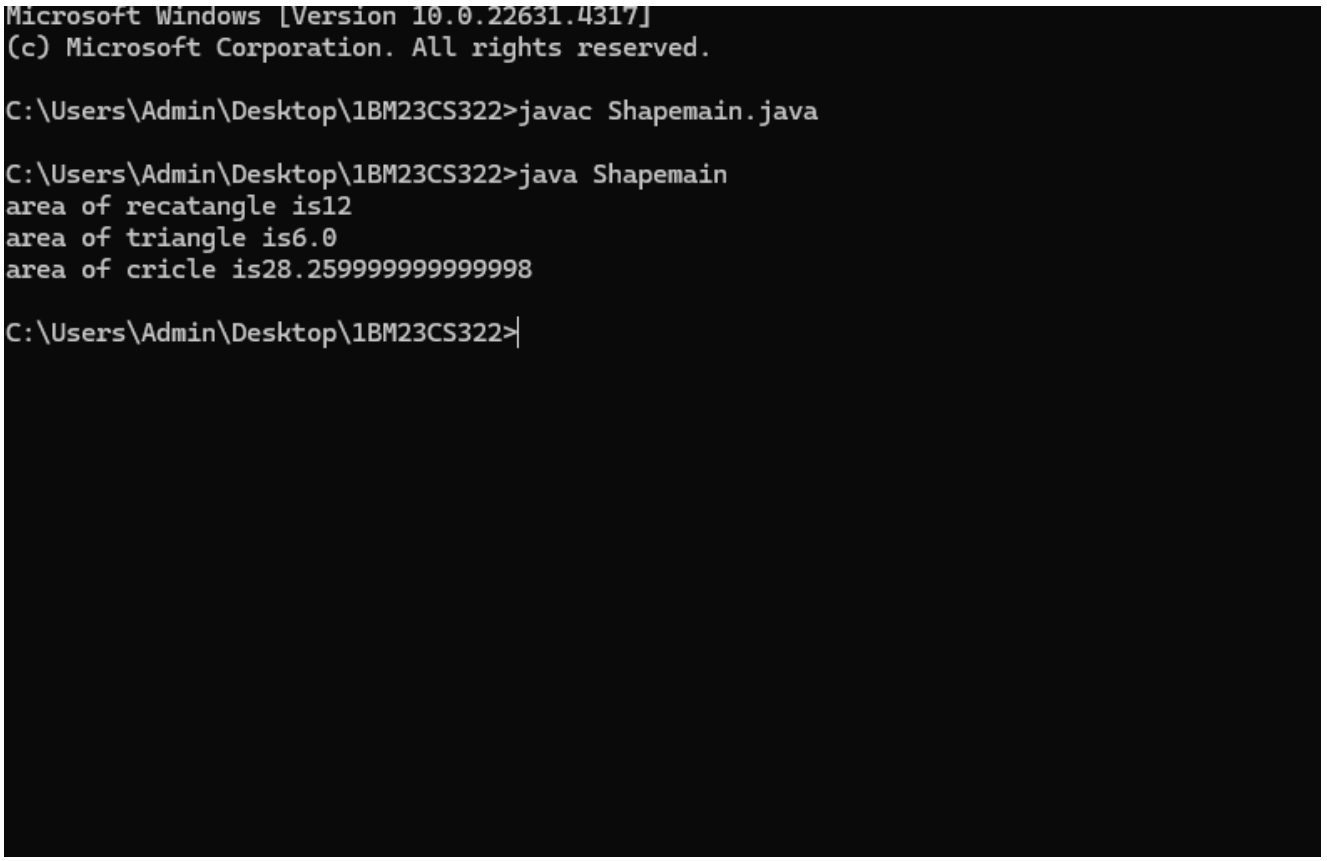
Enter details for Book 2:
Enter name: Python Basics
Enter author: Guido van Rossum
Enter price: 350
Enter number of pages: 450

Details of 2 Books:
Book [Name: Java Programming, Author: James Gosling, price: 450, Pages: 350]
Book [Name: Python Basics, Author: Guido van Rossum, price: 35, Pages: 450]

LAB 4

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

```
import java.util.Scanner;

abstract class Shape{
 int d1,d2;
 abstract void printarea();}

class Rectangle extends Shape{
 Rectangle(int d1,int d2){
  this.d1=d1;
  this.d2=d2;}
 void printarea(){
  System.out.println("area of recatangle is"+(d1*d2));}
}

class Triangle extends Shape{
 Triangle(int d1,int d2){
  this.d1=d1;
  this.d2=d2;}
 void printarea(){
  System.out.println("area of triangle is"+(0.5*d1*d2));}
}

class Circle extends Shape{
 Circle(int d1,int d2){
  this.d1=d1;
  this.d2=1;}
 void printarea(){
  System.out.println("area of cricle is"+(3.14*d1*d1));}}

class Shapemain{
 public static void main(String[] args){
  Rectangle rectangle=new Rectangle(3,4);
  rectangle.printarea();
  Triangle triangle=new Triangle(3,4);
  triangle.printarea();
  Circle circle=new Circle(3,4);
  circle.printarea(); }}
```


Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Admin\Desktop\1BM23CS322>javac Shapemain.java
C:\Users\Admin\Desktop\1BM23CS322>java Shapemain
area of recatangle is12
area of triangle is6.0
area of cricle is28.259999999999998
C:\Users\Admin\Desktop\1BM23CS322>

Lab 4

Date / /
Page

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 of the class contains only one method printarea that print the area.

```
import java.util.Scanner

abstract class Shape {
    int d1, d2;
    abstract void printarea();}

class Rectangle extends Shape {
    Rectangle (int d1, int d2) {
        this.d1=d1;
        this.d2=d2;
    void printarea() {
        System.out.println("area of rectangle =" + (d1*d2));}

class Triangle extends Shape {
    Triangle (int d1, int d2) {
        this.d1=d1;
        this.d2=d2;
    void printarea() {
        System.out.println("area of triangle =" + (0.5*d1*d2));}

class Circle extend Shape {
    Circle (int d1, int d2) {
        this.d1=d1;
    void printarea() {
```

System.out.println("area of circle is"+(3.14*d1*d1));}

class shapemain {
public static void main(String args[]){
Rectangle rectangle = new Rectangle(3,4);
rectangle.printarea();
Triangle triangle = new Triangle(3,4);
triangle.printarea();
Circle circle = new Circle(3,4);
circle.printarea();}}

Output

area of rectangle is 12
area of triangle is 6.0
area of circle is 28.85999

o/p seen.
07/11/24

LAB 5

Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The 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.

```
class Account {
    String customerName;
    String accountNumber;
    double balance;
    public Account(String customerName, String accountNumber, double initialBalance) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.balance = initialBalance;
    }


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


    public void displayBalance() {
        System.out.println("Balance for account " + accountNumber + ": " + balance);
    }


    public boolean withdraw(double amount) {
        return false;
    }


    public void calculateInterest() {
    }
}


class SavingsAccount extends Account {
    private static final double INTEREST_RATE = 0.04;
```

```
class SavingsAccount extends Account {
    private static final double INTEREST_RATE = 0.04;


    public SavingsAccount(String customerName, String accountNumber, double initialBalance) {
        super(customerName, accountNumber, initialBalance);
    }


    public boolean withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println(amount + " withdrawn. New balance: " + balance);
            return true;
        } else {
            System.out.println("Insufficient balance for withdrawal.");
            return false;
        }
    }


    public void calculateInterest() {
        double interest = balance * INTEREST_RATE;
        balance += interest;
        System.out.println("Interest of " + interest + " added to the account. New balance: " + balance);
    }
}
```

```
class Bank {
    private Account[] accounts;
    private int accountCount;


    public Bank(int capacity) {
        accounts = new Account[capacity];
        accountCount = 0;
    }

       public void addAccount(Account account) {
        if (accountCount < accounts.length) {
            accounts[accountCount++] = account;
            System.out.println("Account " + account.accountNumber + " added.");
        } else {
            System.out.println("Bank is at full capacity.");
        }
    }

       public void processAccounts() {
        for (int i = 0; i < accountCount; i++) {
            Account account = accounts[i];
            if (account instanceof SavingsAccount) {
                ((SavingsAccount) account).calculateInterest();
            } else if (account instanceof CurrentAccount) {
                ((CurrentAccount) account).withdraw(0);
            }
        }
    }
}
```

```
C:\Users\shrey\OneDrive\Desktop\java programs>javac BankMain.java

C:\Users\shrey\OneDrive\Desktop\java programs>java BankMain
Account SA001 added.
Account CA001 added.
Account SA002 added.
Account CA002 added.
500.0 deposited. New balance: 2000.0
3000.0 withdrawn. New balance: 2000.0
500.0 withdrawn. New balance: 500.0
500.0 withdrawn. New balance: 1500.0
Balance for account SA001: 2000.0
Balance for account CA001: 2000.0
Balance for account SA002: 500.0
Balance for account CA002: 1500.0

C:\Users\shrey\OneDrive\Desktop\java programs>
```

Date / /
Page

Lab program - 6

a) Develop a java program to create a class Bank that maintain two kinds of accounts 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, but 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 these derive the classes current account and Save account to make them more specific to their requirements. More specific to their requirements. Include the necessary methods in order to achieve the following tasks.

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

⇒ class Account {
String customername;
String accountnumber;
double balance;
public Account (String customername, String accountnumber, double initialbalance){
this.customername = customername;
this.accountnumber = accountnumber;
this.balance = initialbalance;

Date
Page

```
public void deposit (double amount){
    balance += amount;
    System.out.println(amount + "deposited. new balance:" + balance);
}
public void displayBalance(){
    System.out.println("Balance for account " + accountNo + ":" + balance);
}
class SavingsAccount extends Account{
    private final double interestRate = 0.05;
    public SavingsAccount (String customerName, String accountNo, double
        initialBalance){
        super(customerName, accountNumber, initialBalance);
    }
    public boolean withdraw (double amount){
        if (amount <= balance){
            balance -= amount;
            System.out.println(balance);
            return true; }
        else{
            System.out.println("Insufficient balance for withdraw");
            return false; }}
    public void CalculateInterest(){
        double interest = balance * interestRate;
        balance += interest;
        System.out.println("Interest " + interest + "added new balance:"
                balance); }
}
class CurrentAccount extends Account{
    private double minBalance = 1000.00;
    private double penalty = 50.00;
    public CurrentAccount (String customerName, String accountNo,
        double initialBalance){ super(customerName, accountNo, initialBalance);
```

Date / /
Page

```
public boolean withdraw (double amount){
    if (balance - amount < minBalance){
        System.out.println ("min balance")
        return false;
    }else{
        balance -= amount;
        System.out.println (account balance());
        return true;
    }}

public void checkBalance (){
    if (balance < minBalance){
        balance -= penalty;
        System.out.println ("balance");
}}

class Bank{
    private Account[] account;
    private int accountCount;
    public Bank (int capacity){
        account = new Account [capacity];
        accountCount=0; }
    Public void addAccount (Account account){
        if (accountCount< account.length){
            account[accountCount++]=account;
            System.out.println (account.accountNumber);}
        else{
            System.out.println ("Bank is at full capacity");}}
    public void processAccounts (){
        for (int i=0; i< accountCount; i++){
            Account account = account[i];
            if (account instanceof SavingsAccount){
                ((SavingsAccount) account).calculateInterest();
            } else if (account instanceof CurrentAccount){
                ((CurrentAccount) account).checkBalance();
        }}}
```

Page

```
public class Main {
    public static void main (String [] args){
        Bank bank = new Bank(3);
        Savingaccount sa1 = new Savingaccount ("a", "SA001", 1500);
        Currentaccount ca1 = new Currentaccount ("b", "CA001", 5000.0);
        Savingaccount sa2 = new Savingaccount ("c", "SA002", 1000);
        Currentaccount ca2 = new Currentaccount ("d", "CA002", 2000);
        bank.addaccount (sa1);
        bank.addaccount (ca1);
        bank.addaccount (sa2);
        bank.addaccount (ca2);
        sa1.deposit (500);
        ca1.withdraw (3000);
        sa2.withdraw (900);
        ca2.withdraw (500);
        bank.processAccount();
        sa1.displaybalance();
        ca1.displaybalance();
        sa2.displaybalance();
        ca2.displaybalance();
}}
```

Output

```
Account SA001 added
Account CA001 added
Account SA002 added
Account CA002 added
500.0 deposited. New balance: 2000.0
3000.0 withdrawn. New balance: 2000.0
500.0 withdrawn. New balance: 500.0
500.0 withdrawn. New balance: 1500.0
Balance for account SA001: 2000.0
Balance for account SA002: 500.0
Balance for account CA002: 1500
```

write o/p properly

o/p seems

of this

21/1/24

25

LAB 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.

C: > Users > shrey > OneDrive > Desktop > java packages > CIE > Student.java

```java
package CIE;

public class Student {
    String usn;
    String name;
    int sem;
    public Student(String usn, String name, int sem) {
        this.usn = usn;
        this.name = name;
        this.sem = sem;}
    public String getUsn() {
        return usn;}
    public void setUsn(String usn) {
        this.usn = usn;}
    public String getName() {
        return name;}
    public void setName(String name) {
        this.name = name;}
    public int getSem() {
        return sem;}
    public void setSem(int sem) {
        this.sem = sem;}
}

```

C: > Users > shrey > OneDrive > Desktop > java packages > CIE > Internals.java

```java
package CIE;

public class Internals extends Student {
    protected int[] internalMarks = new int[5];
    public Internals(String usn, String name, int sem, int[] internalMarks) {
        super(usn, name, sem);
        this.internalMarks = internalMarks;
    }
    public void displayInternalMarks() {
        System.out.println("Internal marks for " + name + ": ");
        for (int i = 0; i < internalMarks.length; i++) {
            System.out.println("Course " + (i + 1) + ": " + internalMarks[i]);}
    }
}

```

C: > Users > shrey > OneDrive > Desktop > java packages > SEE > External.java

```java

import CIE.Internals;
public class External extends Internals {
    protected int[] externalMarks = new int[5];
    public External(String usn, String name, int sem, int[] internalMarks, int[] externalMarks) {
        super(usn, name, sem, internalMarks);
        this.externalMarks = externalMarks;
    }
    public void displayExternalMarks() {
        System.out.println("External marks for " + name + ": ");
        for (int i = 0; i < externalMarks.length; i++) {
            System.out.println("Course " + (i + 1) + ": "  + externalMarks[i]);}
    }
}

```

C: > Users > shrey > OneDrive > Desktop > java packages > Marksmain_.java

```java
import CIE.Internals;
import SEE.External;
public class Marksmain_ {
    public static void main(String[] args){
        int[] internalMarks1 = {85, 90, 78, 88, 92};
        int[] externalMarks1 = {75, 80, 65, 78, 85};
        int[] internalMarks2 = {80, 85, 75, 87, 90};
        int[] externalMarks2 = {70, 74, 68, 77, 82};
        int[] internalMarks3 = {92, 95, 88, 90, 91};
        int[] externalMarks3 = {80, 85, 78, 85, 90};
        External student1 = new External("1BM23CS001", "A", 3, internalMarks1, externalMarks1);
        External student2 = new External("1BM23CS002", "B", 3, internalMarks2, externalMarks2);
        External student3 = new External("1BM23CS003", "C", 3, internalMarks3, externalMarks3);
        student1.displayInternalMarks();
        student1.displayExternalMarks();
        student2.displayInternalMarks();
        student2.displayExternalMarks();
        student3.displayInternalMarks();
        student3.displayExternalMarks();
    }
}

```

```
C:\Users\shrey\OneDrive\Desktop\java packages>javac Marksmain_.java

C:\Users\shrey\OneDrive\Desktop\java packages>java Marksmain_
Internal marks for A:
Course 1: 85
Course 2: 90
Course 3: 78
Course 4: 88
Course 5: 92
External marks for A:
Course 1: 75
Course 2: 80
Course 3: 65
Course 4: 78
Course 5: 85
Internal marks for B:
Course 1: 80
Course 2: 85
Course 3: 75
Course 4: 87
Course 5: 90
External marks for B:
Course 1: 70
Course 2: 74
Course 3: 68
Course 4: 77
Course 5: 82
Internal marks for C:
Course 1: 92
Course 2: 95
Course 3: 88
Course 4: 90
Course 5: 91
External marks for C:
Course 1: 80
Course 2: 85
Course 3: 78
Course 4: 85
Course 5: 90

C:\Users\shrey\OneDrive\Desktop\java packages>
```

```
C:\Users\shrey\OneDrive\Desktop\java packages>javac -d . Student.java

C:\Users\shrey\OneDrive\Desktop\java packages>javac -d . Internal.java

C:\Users\shrey\OneDrive\Desktop\java packages>javac -d . External.java
```

Date / /
Page

lab-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 the courses of the current semester of the student. Import the two packages in a file that declares the final marks of a student in all five courses.

```
package CIE;
public class student{
    String usn;
    String name;
    int sem;
    public student(String usn, String name, int sem){
        this.usn = usn;
        this.name = name;
        this.sem = sem;
    public String getusn(){
        return usn;}
    public void setusn(String usn){
        this.usn = usn;}
    public void setName(String name){
        this.name = name;}
    public int getSem(){
        return sem;}
    public void setSem(int sem){
        this.sem = sem;}}
```

Date / /
Page

```
package CIE;
public class Internals extends Student {
    protected int[] internalMarks = new int[5];
    public Internals(String usn, String name, int sem, int[] internalMarks) {
        super(usn, name, sem);
        this.internalMarks = internalMarks;
    }
    public void displayInternalMarks() {
        System.out.println("Internal marks for " + name + ":");
        for (int i = 0; i < internalMarks.length; i++) {
            System.out.println("Course" + (i+1) + ": " + internalMarks[i]);
        }
    }
}
package SEE;
import CIE.Internals;
public class External extends Internals {
    protected int[] externalMarks = new int[5];
    public External(String usn, String name, int sem, int[] internalMarks,
                    int[] externalMarks) {
        super(usn, name, sem, internalMarks);
        this.externalMarks = externalMarks;
    }
    public void displayExternalMarks() {
        System.out.println("External marks for " + name + ":");
        for (int i = 0; i < externalMarks.length; i++) {
            System.out.println("Course" + (i+1) + ": " + externalMarks[i]);
        }
    }
}
import CIE.Internals;
import SEE.External;
public class MarksEntry {
    public static void main(String[] args) {
        int[] internalMarks1 = {85, 90, 78, 88, 92};
        int[] externalMarks1 = {75, 80, 65, 78, 85};
        int[] internalMarks2 = {80, 85, 70, 89, 94};
        int[] externalMarks2 = {70, 96, 88, 72, 89};
```

```
int[] internalMarks = {99, 95, 88, 90, 93};
int[] externalMarks = {80, 85, 78, 87, 90};
External student1 = new External("1BM23CS001", 'A', 3, internal
                                  marks, externalMarks);
External student2 = new External("1BM23CS002", 'B', 3, internalmarks,
                                  externalMarks2);
External student3 = new External("1BM23CS003", 'C', 3, internalMarks,
                                  externalMarks3);
student1.displayInternalMarks();
student1.displayExternalMarks();
student2.displayInternalMarks();
student2.displayExternalMarks();
student3.displayInternalMarks();
student3.displayExternalMarks(); }
}
```

Output:

Internal marks for A:

Course 1: 85

Course 2: 90

Course 3: 95

Course 4: 88

Course 5: 92

External mark for A:

Course 1: 75

Course 2: 80

Course 3: 65

Course 4: 78

Course 5: 85

Internal marks for B:

Course 1: 80

Course 2: 85

Course 3: 87

Course 4: 82

Course 5: 90

LAB 7

Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor 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.

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

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

class Father {
    private int age;
    public Father(int age) throws WrongAgeException {
        if (age < 0) {
            throw new WrongAgeException("Wrong age");
        }
        this.age = age;
    }

}

class Son extends Father {
    private int sonAge;
    public Son(int fatherAge, int sonAge) throws WrongAgeException, SonAgeException {
        super(fatherAge);
        if (sonAge >= fatherAge) {
            throw new SonAgeException("Son's age cannot be greater than or equal to father's age");
        }
        else if(sonAge <=0 ){
            throw new SonAgeException("Son's age cannot be less than zero");
        }
        this.sonAge = sonAge;
    }

}
public class FatherSon{
    public static void main(String[] args) {
        while(true){
            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();
            try {
                Son son = new Son(fatherAge, sonAge);
                System.out.println("Accepted Succesfully");
            }
            catch (WrongAgeException e) {
                System.out.println(e.getMessage());
            }
            catch (SonAgeException e) {
                System.out.println(e.getMessage());
            }
            System.out.println("Would you like to re-enter details (Yes/no)");
            String input = sc.next();
            if (input.equalsIgnoreCase("no")) {
                break;
            }
        }
    }
}
```

```
C:\Users\Admin\Desktop\1BM23CS322>javac FatherSon.java

C:\Users\Admin\Desktop\1BM23CS322>java FatherSon
Enter Father's Age: 45
Enter Son's Age: -12
Son's age cannot be less than zero
Would you like to re-enter details (Yes/no)
yes
Enter Father's Age: -12
Enter Son's Age: 23
Wrong age
Would you like to re-enter details (Yes/no)
yes
Enter Father's Age: 23
Enter Son's Age: 45
Son's age cannot be greater than or equal to father's age
Would you like to re-enter details (Yes/no)
```

Lab- Program - 07

Date / /
Page

a) 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 & throws the exception WrongAge() when the input age < 0. In Son class, implement a constructor that takes both Father & Son's age & throws an exception if Son's age >= Father's age.

⇒

```
import java.util.Scanner;
class WrongAgeException extends Exception{
    public WrongException(String message){
        super(message);
    }
}

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

class Father{
    private int age;
    public Father(int age) throws WrongAgeException{
        if (age<0){
            throw new WrongAgeException("wrong age");
        }
        this.age=age;
    }
    public int getAge(){
        return age;
    }
}
```

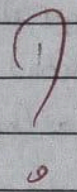
Date / /
Page

```
class son extends father{
    private int sonAge;
    public son(int fatherAge, int sonAge) throws WrongAge
                Exception, SonAgeException{
        super(fatherAge);
        if (sonAge >= fatherAge){
            throw new SonAgeException("Son's age cannot be
                                       greater or equal to father's age");}
        this.sonAge= sonAge;
    }
    public int getSonAge(){
        return sonAge;
    }
}

public class fatherSon{
    public static void main(String[] args){
        while(true){
            Scanner sc=new Scanner(System.in);
            System.out.print("Enter father's Age:");
            int fatherAge = sc.nextInt();
            System.out.print("Enter Son's Age:");
            int sonAge=sc.nextInt();
            try{
                son Son = new Son(fatherAge, sonAge);
                System.out.println("Accepted Successfully");
            }
            Catch (WrongAgeException e){
                System.out.println(e.getMessage());}
            Catch (SonAgeException e){
                System.out.println(e.getMessage());}
            System.out.println("Would you like to re-enter data (Y/N)");
            String input = sc.next();
```

Date / /
Page

- if (input.equalsIgnoreCase("n")) {
  break;
}
}

output:



output: Enter father's Age: 48
Enter Son's Age: -18
Son's age Cannot be less than zero
Could you like to re-enter detail (yes/no)
yes
Enter father's Age: -12
Enter Son's Age: 23
Wrong age
Could you like to re-enter details (yes/no)
yes
Enter father's Age: 23
Enter Son's Age: 45
Son's Age Cannot be greater to or equal to father's age.

Obj sent
11/11/24

LAB 8

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

```
class Thread1{
  public static void main(String[] args){
    Thread threadBMS = new Thread(new DisplayBMS());
    Thread threadCSE = new Thread(new DisplayCSE());
    threadBMS.start();
    threadCSE.start();
}
}
class DisplayBMS implements Runnable{
  public void run(){
    try{
      while(true){
        System.out.println( "BMS college of engineering" ) ;
        Thread.sleep(10000);
}
}
    catch(InterruptedException e){
       System.out.println("interrupted"+e.getMessage());
}
}
}
class DisplayCSE implements Runnable{
   public void run(){
     try{
       while(true){
         System.out.println( "CSE" );
         Thread.sleep(20000);
}
}
     catch(InterruptedException e){
        System.out.println("Interupted"+e.getMessage());
}
}
}
```

```
BMS college of engineering
CSE
BMS college of engineering
CSE
BMS college of engineering
BMS college of engineering
CSE
BMS college of engineering
BMS college of engineering
CSE
```

Page

lab-8

a) write a program which create two threads, displaying "BMS College of engineering" once ten seconds and another displaying "CSE" every two seconds

```
=> class Thread{
    public static void main(String args[]){
      Thread threadBMS = new thread(new DisplayBMS());
      Thread threadCSE = new thread(new DisplayCSE());
      threadBMS.start();
      threadCSE.start();
    }
}

class DisplayBMS implements Runnable{
    public void run(){
      try{
        while(true){
          System.out.println("BMS College of Engineering");
          Thread.sleep(10000);
        }
      }
      Catch (InterruptedException e){
        System.out.println("Interrupted"+e.getMessage());
      }
}
class DisplayCSE implements Runnable{
    public void run(){
      try{
        while(true){
          System.out.println("CSE");
          Thread.sleep(2000);
        }
      }
      Catch(InterruptedException e){
        System.out.println("Interrupted"+e.getMessage());
      }
}
```

output:

BMS College of Engineering

CSE

BMS College of Engineering

BMS College of Engineering

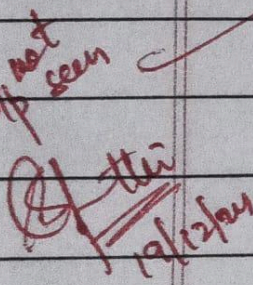
CSE

BMS College of Engineering

BMS College of Engineering

CSE

op not seen



LAB 9

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

ers > shrey > OneDrive > Desktop > java programs > J Main.java

```
import java.awt.*;
import java.awt.event.*;
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("",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(){
```

```
        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        });
    }
    public void actionPerformed(ActionEvent e)
    {
        int n1,n2;
        try
        {
            if (e.getSource() == dResult)
            {
                n1=Integer.parseInt(num1.getText());
                n2=Integer.parseInt(num2.getText());
                if(n2==0)
                {throw new ArithmeticException();}
                out=n1+"/"+n2+" ";
                resultNum=n1/n2;
                out+=resultNum;
            }
        }
        catch(NumberFormatException e1)
        {
            flag=1;
            out="Number Format Exception!"+e1;
        }
```

```
        catch(ArithmeticException e1)
        {
            flag=1;
            out="Divide by 0 Exception!"+e1;
        }
        outResult.setText(out);
        invalidate();
        validate();
    }
    public void paint(Graphics g)
    {
        if(flag==0)
        {g.drawString(out,dResult.getX()+dResult.getWidth(),dResult.getY()+outResult.getHeight()-8);}
        else
        {g.drawString(out,100,200); flag=0;}
    }

}
public class Main
{
    public static void main(String args[])
    {
        DivisionMain1 obj=new DivisionMain1();
        obj.setSize(new Dimension(800,400));
        obj.setTitle("DivisionOfIntegers");
        obj.setVisible(true);
    }
}
```


DivisionOfIntegers
Number 1:
45
Number 2:
9
Result
45/9 5.0

Date / /
Page

Lab-9

a) write a program that creates a user interface to perform integer divisions. The user enters two numbers in text fields, Num1 & Num2. The division of Num1 & Num2 is displayed in the result field when the Divide button is clicked. If Num1 were zero, the program would throw an Arithmetic Exception. Display the exception in message dialog box.

=> import java.awt.*;
import java.awt.event.*;

public class DivisionMain extends Frame implements ActionListener
{
    TextField num1, num2;
    ~~num1, num2;~~
    Button dResult;
    Label outResult;
    String str = "";
    double resultNum;
    int flag = 0;
    public DivisionMain(){
        setLayout(new FlowLayout());
        dResult = new Button("Result");
        Label number1 = new Label("Number 1:", Label.RIGHT);
        ~~Label number2 =~~ ~~Label number2;~~
        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);
    }
});
}
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();
            out = n1 + " " + n2 + " ";
            resultNum = n1/n2;
            out += String.valueOf(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();
    }
}
```

Date / /
Page

```
public void paint(Graphics g){
    if(flag==0)
    g.drawString(out, outResult.getX()+outResult.getWidth(),
    outResult.getY()+outResult.getHeight()-8);
    else
    g.drawString(out,100,200);
    flag=0;
}

public class Main
{
    public static void main(String args[])
    {
        DivisionMain1 obj = new DivisionMain();
        obj.setSize(new Dimension(800,400));
        obj.setTitle("Division of Integers");
        obj.setVisible(true);
    }
}
```

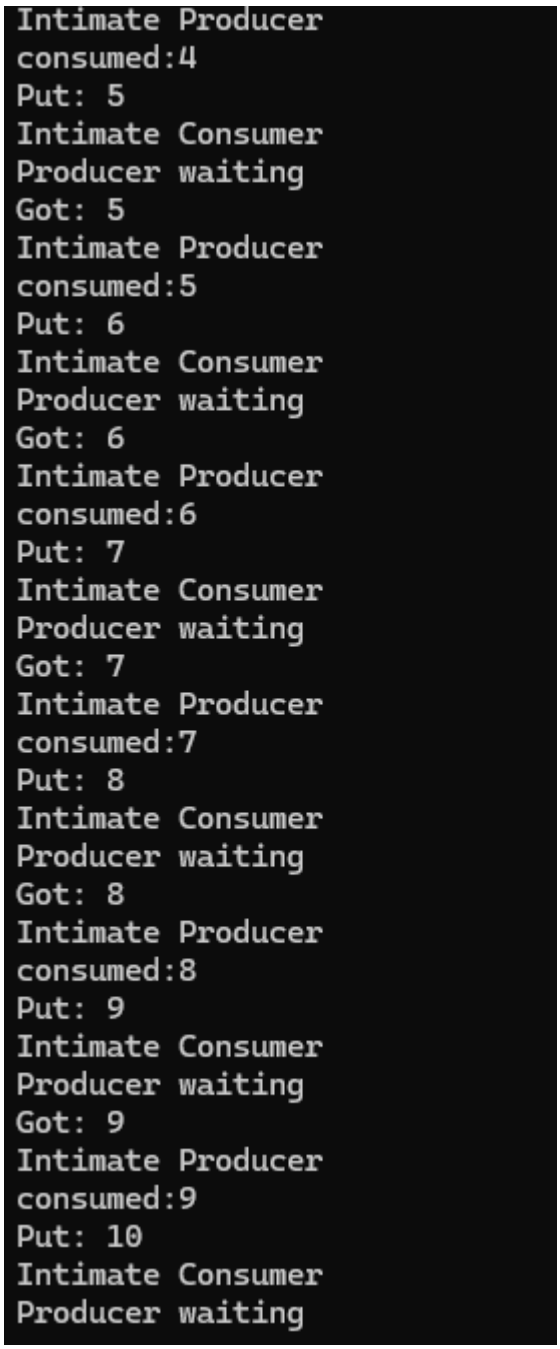
Lab 10

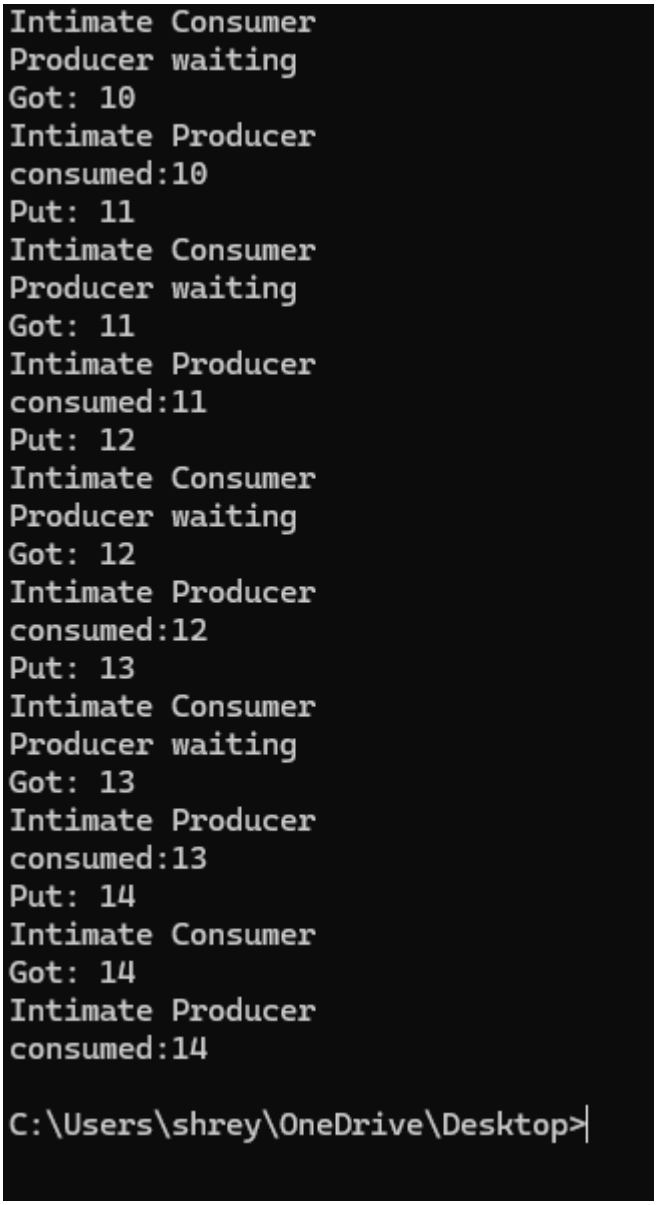
Demonstrate Interprocess communication and deadlock

```
class Q {
int n;
boolean valueSet = false;
synchronized int get() {
while(!valueSet)
try {
System.out.println("Consumer waiting");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
System.out.println("Got: " + n);
valueSet = false;
System.out.println("Intimate Producer");
notify();
return n;
}
synchronized void put(int n) {
while(valueSet)
try {
System.out.println("Producer waiting");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("Intimate Consumer");
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++;
}
}
}
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.");
}
}
```

```
C:\Users\shrey\OneDrive\Desktop>javac PCFixed.java

C:\Users\shrey\OneDrive\Desktop>java PCFixed
Press Control-C to stop.
Put: 0
Intimate Consumer
Producer waiting
Got: 0
Intimate Producer
Put: 1
consumed:0
Intimate Consumer
Producer waiting
Got: 1
Intimate Producer
consumed:1
Put: 2
Intimate Consumer
Producer waiting
Got: 2
Intimate Producer
consumed:2
Put: 3
Intimate Consumer
Producer waiting
Got: 3
Intimate Producer
consumed:3
Put: 4
Intimate Consumer
Producer waiting
Got: 4
```


Intimate Producer
consumed:4
Put: 5
Intimate Consumer
Producer waiting
Got: 5
Intimate Producer
consumed:5
Put: 6
Intimate Consumer
Producer waiting
Got: 6
Intimate Producer
consumed:6
Put: 7
Intimate Consumer
Producer waiting
Got: 7
Intimate Producer
consumed:7
Put: 8
Intimate Consumer
Producer waiting
Got: 8
Intimate Producer
consumed:8
Put: 9
Intimate Consumer
Producer waiting
Got: 9
Intimate Producer
consumed:9
Put: 10
Intimate Consumer
Producer waiting


Intimate Consumer
Producer waiting
Got: 10
Intimate Producer
consumed:10
Put: 11
Intimate Consumer
Producer waiting
Got: 11
Intimate Producer
consumed:11
Put: 12
Intimate Consumer
Producer waiting
Got: 12
Intimate Producer
consumed:12
Put: 13
Intimate Consumer
Producer waiting
Got: 13
Intimate Producer
consumed:13
Put: 14
Intimate Consumer
Got: 14
Intimate Producer
consumed:14
C:\Users\shrey\OneDrive\Desktop>

Q:) Demonstrate interprocess communication.

=> Class Q{
int n;
boolean valueset = false;
Synchronized int get(){
while(!valueset)
try{
System.out.println("Consumer waiting");
wait();}
Catch (InterruptedException e){
System.out.println("InterruptedException caught");
}
System.out.println("Got: "+n);
value set = false;
System.out.println("Intimat producer");
notify();
return n;
}
Synchronized void put(int n){
while(valueset)
try{
System.out.println("producer waiting");
wait();
}Catch(InterruptedException e){
System.out.println("Interrupted Caught");
}
this.n=n;
valueset = true;
System.out.println("put:"+n);
System.out.println("Intimat Consumer");
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++;
}
}
}
class PCFixed {
public static void main (String args[]) {
Q q = new Q();
new Producer(q);
new Producer(q);
System.out.println("Press Control-C to stop.");
}
}
```

output press control c to stop
put:0
intimate consumer
producer waiting
Got:0
intimate producer
put:1
Consumed:0
Intimate Consumer
producer waiting
Got:1
intimate producer
Consumed:1
put:0
intimate consumer
producer waiting
Got:0
Intimate Producer
Consumed:0
put:3
Intimate Consumer
producer waiting
Got:3
Intimate producer
Consumed:3
put:4
intimate Consumer
producer waiting
Got:4
intimate producer
Consumed:4
put:5

Page

Intimate Product
Consumed: 5
put: 6
Intimate Consumer
producer waiting
Got: 6
Intimate producer
Consumed: 6
put: 7
Intimate Consumer
producer waiting
Got: 7
Intimate producer
Consumed: 7
put: 8
Intimate Consumer
producer waiting
Got: 8
Intimate producer
Consumed: 8
put: 9
Intimate consumer
producer waiting
Got: 9
intimate producer
Consumed: 9
put: 10
Intimate Consumer
producer waiting
Got: 10
Intimate producer
Consumed: 10

Date / /
Page

put:11
Intimat Consume
Producer waiting
Consumed: 11
put:12
Intimate Consumer
producer waiting
Got:12
Intimat producer
Consumed:12
put:13
Intimat Consumer
producer waiting
Got:13
Intimate producer
Consumed:13
put:14
Intimat Consumed
Got:14
Intimate producer
Consumed:14

```
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);
        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();
    }
}
```

```
C:\Users\shrey\OneDrive\Desktop\java programs>javac Deadlock.java

C:\Users\shrey\OneDrive\Desktop\java programs>java Deadlock
RacingThread entered B.bar
MainThread entered A.foo
RacingThread trying to call A.last()
MainThread trying to call B.last()
```

Date / /
Page

2) 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");}
      }
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 A.last");}
}
class Deadlock implements Runnable
{ A a = new A();
  B b = new B();
```

Date / /
Page

```
Deadlock(){
    Thread.currentThread().setName("MainThread");
    Thread t = new Thread(this, "RacingThread");
    t.start();
    a.foo(b);
    System.out.println("Back in main thread");
}

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

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

public static void main(String[] args)
    DeadlockMain dm = new DeadlockMain();
    dm.setSize(new Dimension(800, 400));
    dm.setTitle("Deadlock of Integers");
    dm.setVisible(true);
}
```

=> output:

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