VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SOHAN R (1BM23CS336)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019 Sep

2024-Jan 2025

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**OBJECT ORIENTED JAVA PROGRAMMING**" carried out by **SOHAN R (1BM23CS336),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of **Object-Oriented Java Programming Lab - (23CS3PCOOJ)** work prescribed for the said degree.

Dr. Nandhini Vineeth

Associate Professor,
Department of CSE,
BMSCE, Bengaluru

Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

INDEX

Github Link : https://github.com/SohanR-1BM23CS336/1BM23CS336-SOHAN-R---OOJ-LAB

LABORATORY PROGRAM – 1

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

OBSERVATION :

Teacher Sign / Remarks

Q.1. Develop a Java Program that prints all real solutions to the quadratic equation $ax^2 + bx + c = 0$. Read in a, b, c and use the quadratic formula. If the discriminate $b^2 - 4ac$ is negative, display a message stating that there are no real solution.

Ans:

```
import java.util.Scanner;
class quadratic {
    float d;
    Scanner sc = new Scanner(System.in);

    void check() {
        System.out.println("enter the values of a, b and c");
        int a = sc.nextInt();
        int b = sc.nextInt();
        int c = sc.nextInt();
        if (a==0){
            System.out.println("invalid equation");}
        else {
            d = b*b - 4*a*c;
            System.out.println(d);
            System.out.println(" the solutions are");
            if (d>0){
                ~~System.out.println(" the solutions are ");~~
                System.out.println("roots are unique ");
                double r1 = (-b + Math.sqrt(d))/(2*a);
                double r2 = (-b + Math.sqrt(d))/(2*a);
                System.out.println(r1+" "+ r2);
            }
            if (d==0){
                System.out.println("roots are equal ");
                double r=-b/(2*a);
                System.out.println(r);}
```

1

```
                if (d<0){
                    System.out.println("roots are imaginary");
                    double r1 = math.sqrt(-d)/(2*a);
                    double r2 = (-b)/(2*a);
                    System.out.println(r2+" + i "+r1+" "+r2+"-i"+r1);
                }
            }
        }
    }

public class Main{
    public static void main(String args[]){
        quadratic q1 = new quadratic();
        q1.check();
    }
}
```

Observation :-

enter the values of a, b and c

1
-3
2

1.0
the solution are
roots are unique
2.0 1.0

Q.2. Deve
with
Inclu
meth

Ans.
impo
publ

}

CODE :

```
import java.util.Scanner;

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

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

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

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

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



        if (discriminant > 0) {

            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            System.out.println("The equation has two real and distinct
solutions:");
            System.out.println("Root 1: " + root1);
            System.out.println("Root 2: " + root2);
```

```
        } else if (discriminant == 0) {


            double root = -b / (2 * a);
            System.out.println("The equation has one real solution (a double root):");
            System.out.println("Root: " + root);
        } else {


            System.out.println("The equation has no real solutions.");
        }
        scanner.close();
    }
}
```

OUTPUT :

```
Enter the coefficient a: 1
Enter the coefficient b: -3
Enter the coefficient c: 2
The equation has two real and distinct solutions:
Root 1: 2.0
Root 2: 1.0
PS D:\3rd sem\OOJ JAVA\Git-hub> java QuadraticEquationSolver
Enter the coefficient a: 1
Enter the coefficient b: 2
Enter the coefficient c: 1
The equation has one real solution (a double root):
Root: -1.0
```

LABORATORY PROGRAM – 2

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

OBSERVATION :

Q.2. Develop a Java Program ~~that prints~~ 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.

Ans.

```
import java.util.Scanner;
public class cgpa{
    public static void main(String[] args){
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of subjects: ");
        int numSubjects = scanner.nextInt();
        double[] gradePoints = new double[numSubjects];
        int[] credits = new int[numSubjects];
        int totalCredits = 0;
        double total = 0;

        for(int i=0; i< numSubjects; i++){
            System.out.print("Enter the grade points for subject"+
                                (i+1)+":");
            gradePoints[i] = scanner.nextDouble();
            System.out.print("Enter the credits for subject"+(i+1)+":");
            credits[i]= scanner.nextInt();
            total += gradePoints[i]*credits[i];
            totalCredits += credits[i];
        }
        double cgpa= total / totalCredits;
        System.out.printf("Your CGPA is:"+ cgpa);
    }
}
```

Observation:-

Enter the number of subjects: 5

Enter the grade points for subject 1: 9

Enter the credits for subject 1: 4

Enter the grade points for subject 2: 10

Enter the credit for subject 2: 5

Enter the grade points for subject 3: 10

Enter the ~~grad~~ credit for subject 3: 5

Enter the grade points for subject 4: 9

Enter the credit for subject 4: 3

Enter the grade points for subject 5: 1

Enter the credit for subject 5: 1

Your CGPA is: 9.111111111111111111

CODE :

```
import java.util.Scanner;
class Student {
    String usn;
    String name;
    double[] credits;
    double[] marks;
    int numSubjects
    Student(int numSubjects) {
        this.numSubjects = numSubjects;
        credits = new double[numSubjects];
        marks = new double [numSubjects];
    }
    public void acceptDetails() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter USN: ");
        usn = scanner.nextLine();
        System.out.print("Enter Name: ");
        name = scanner.nextLine();
        for (int i = 0; i < numSubjects; i++) {
            System.out.print("Enter credits for subject " + (i+1) + ": ");
            credits[i] = scanner.nextDouble();
            System.out.print("Enter marks for subject " + (i+1) + ": ");
```

```
            marks[i] = scanner.nextDouble();}}
    public void displayDetails() {
        System.out.println("\nStudent Details:");
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Subjects Details: ");
        for (int i = 0; i < numSubjects; i++) {
            System.out.println("Subject " + (i+1) + ": Credits = " +
credits[i] + ", Marks = " + marks[i]);}   }
    public void calculateSGPA() {
        double totalCredits = 0;
        double totalGradePoints = 0;

        for (int i = 0; i < numSubjects; i++) {
            double gradePoint = getGradePoint(marks[i]);
            totalGradePoints += gradePoint * credits[i];
            totalCredits += credits[i];
        }
        double sgpa = totalGradePoints / totalCredits;
        System.out.println("SGPA: " + sgpa);}
    private double getGradePoint(double marks) {
        if (marks >= 90) return 10;
        else if (marks >= 80) return 9;
        else if (marks >= 70) return 8;
```

```
        else if (marks >= 60) return 7;
        else if (marks >= 50) return 6;
        else if (marks >= 40) return 5;
        else return 0;  }
public class StudentMain {
    public static void main(String[] args) {
        Student student = new Student(3);
        student.acceptDetails();
        student.displayDetails();
        student.calculateSGPA();
    }
}
```

OUTPUT :

```
Enter USN: 123
Enter Name: Sushanth Rai
Enter credits for subject 1: 3
Enter marks for subject 1: 80
Enter credits for subject 2: 4
Enter marks for subject 2: 90
Enter credits for subject 3: 3
Enter marks for subject 3: 80

Student Details:
USN: 123
Name: Sushanth Rai
Subjects Details:
Subject 1: Credits = 3.0, Marks = 80.0
Subject 2: Credits = 4.0, Marks = 90.0
Subject 3: Credits = 3.0, Marks = 80.0
SGPA: 9.4
```

LABORATORY PROGRAM – 3

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

OBSERVATION :

Q.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.

Ans:

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

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

        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages; }
    public String getDetails(){
        return toString();
    }

    public String toString(){
        return "Book Name: " + name + ", Author: " + author + ", Price: " +
                price + ", Pages: " + numPages; }
    }
```

```
public class BookDemo{
    public static void main (String[] args){
        Scanner scanner = new Scanner (System.in);
        System.out.print ("Enter number of books: ");
        int n = scanner.nextInt();
        scanner.nextLine();
        Book[] books = new Book[n];
        for (int i=0 ; i<n ; i++){
            System.out.println ("Enter details for book"+(i+1)+":");
            System.out.print ("Name: ");
            String name = scanner.nextLine();
            System.out.print ("Author: ");
            String author = scanner.nextLine ();
            System.out.print ("Price:" );
            double price = scanner.nextDouble();
            System.out.print ("Number of Pages: ");
            int numPages = scanner.nextInt();
            scanner.nextLine();

            books[i] = new Book(name, author, price, numPages);
            System.out.println(books[i].getDetails()); }
        scanner.close();
    }
}
```

Observation:

Enter ~~the~~ number of books: 2

Enter the details of book 1:

Name: r d sharma

Author: r d sharma

Price: 2000

Number of Pages: 400

Book Name: rd sharma, Author: rd sharma, Price: Rs. 2000, ~~Number of~~ Pages: 400

Enter the details of book 2:

Name: h c verma

Author: h c verma

Price: 200

Number of Pages: 300

Book Name: h c verma, Author: h c verma, Price: Rs 200,

CODE :

```
import java.util.Scanner;
class Book {
    private String name;
    private String author;
    private double price;
    private int numPages;
    public Book(String name, String author, double price, int numPages) {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;}
    public 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(double price) {
        this.price = price;  }
    public int getNumPages() {
        return numPages;  }
```

```
    public void setNumPages(int numPages) {
        this.numPages = numPages;}
    @Override
    public String toString() {
        return "Book Name: " + name + "\nAuthor: " + author + "\nPrice: "
+ price + "\nNumber of Pages: " + numPages;  }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of books: ");
        int n = sc.nextInt();
        sc.nextLine();
        Book[] books = new Book[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for book " + (i + 1) + ":");
            System.out.print("Enter Book Name: ");
            String name = sc.nextLine();
            System.out.print("Enter Author Name: ");
            String author = sc.nextLine();
            System.out.print("Enter Price: ");
            double price = sc.nextDouble();
            System.out.print("Enter Number of Pages: ");
            int numPages = sc.nextInt();
            sc.nextLine();
            books[i] = new Book(name, author, price, numPages);
        }
        System.out.println("\nDetails of all books:");
        for (int i = 0; i < n; i++) {
```

```
            System.out.println("\nBook " + (i + 1) + " Details:");
            System.out.println(books[i].toString());
        }
        sc.close();
    }
}
```

OUTPUT :

```
Enter the number of books: 2
Enter details for book 1:
Enter Book Name: Book 1
Enter Author Name: Author 1
Enter Price: 123
Enter Number of Pages: 13
Enter details for book 2:
Enter Book Name: Book
Enter Author Name: Author 2
Enter Price: 321
Enter Number of Pages: 35

Details of all books:

Book 1 Details:
Book Name: Book 1
Author: Author 1
Price: 123.0
Number of Pages: 13

Book 2 Details:
Book Name: Book
Author: Author 2
Price: 321.0
Number of Pages: 35
```

LABORATORY PROGRAM – 4

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

OBSERVATION :

Q.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.

```
import java.util.Scanner;
abstract class Shape {
    double dim1, dim2;

    Shape(double a, double b){
        dim1 = a;
        dim2 = b;
    }
    abstract double printArea();
}

class Rectangle extends Shape{
    Rectangle (double a, double b){
        super(a,b);
    }
    double printArea(){
        System.out.printf("The area of the Rectangle is: "+(dim1*dim2)
                                                    + "\n");
        return 0;
    }
}

class Triangle extends Shape {
    Triangle (double a, double b){
        super(a,b);
    }
    double printArea(){
        System.out.printf("The area of the Triangle is: "+
                                    ((dim1* dim2)/2) + "\n");
        return 0;
    }
}
```

7

```
class Circle extends Shape {
    Circle (double a, double b){
        super (a,b);
    }
    double printArea(){
        System.out.printf("The area of the Circle is: %f
                                (3.14* dim1* dim1)+"\n");

        return 0;
    }
}

class Area {
    public static void main (String[] args){
        Rectangle r = new Rectangle (4,5);
        Triangle t = new Triangle (10,8);
        Circle c = new Circle (10,10);
        Shape s;
        s=r;
        s.printArea();
        s=t;
        s.printArea();
        s=c;
        s.printArea();
    }
}
```

Observations

The area of the Rectangle is: 20.0

The area of the Triangle is: 40.0

The area of the Circle is: 314.0

CODE :

```
abstract class Shape {
    int d1;
    int d2;
    Shape(int d1, int d2) {
        this.d1 = d1;
        this.d2 = d2;
    }
    abstract void printArea();
}
class Rectangle extends Shape {
    Rectangle(int l, int b) {
        super(l, b);
    }
    void printArea() {
        System.out.println("Rectangle Shape");
        System.out.println("The area is : " + d1 * d2);}}
class Triangle extends Shape {
    Triangle(int b, int h) {
        super(b, h);
    }
    void printArea() {
        System.out.println("Triangle Shape");
        System.out.println("The area is : " + 0.5 * d1 * d2);
    }
}
```

```
class Circle extends Shape {
    Circle(int r) {
        super(r, 0);
    }
    void printArea() {
        System.out.println("Circle Shape");
        System.out.println("The area is : " + (Math.PI * d1 * d1));}}
class ShapeMain {
    public static void main(String[] args) {
        Shape shape;
        shape = new Rectangle(5, 2);
        shape.printArea();
        shape = new Triangle(5, 2);
        shape.printArea();
        shape = new Circle(5);
        shape.printArea();
    }
}
```

OUTPUT :

```
Rectangle Shape
The area is : 10
Triangle Shape
The area is : 5.0
Circle Shape
The area is : 78.53981633974483
```

LABORATORY PROGRAM – 5

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

OBSERVATION :

Q.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 facilities. The current account provides check book facilities 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 accountant 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 the deposit interest.

d) Permit withdrawal and update the balance

Check for the minimum balance, impose penalty if necessary and update the balance.

Ans:

```
import java.util.Scanner;
abstract class Account{
        protected String customerName;
        protected String accountNumber;
        protected String accountType;
        protected double balance;

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

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

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

        public abstract void withdraw (double amount);
        public abstract void checkMinimumBalance(); }

class SavingsAccount extends Account{
        private static final double INTEREST_RATE = 0.05;
        public SavingsAccount (String customerName, String accountNumber,
                double initialBalance){
            super(customerName, accountNumber, "Savings", initialBalance);}

        public void computeAndDepositInterest(){
            double interest = balance * INTEREST_RATE;
            balance += interest;
            System.out.println("Interest computed and added: " + interest);
            displayBalance();
        }
```

```
@Override
public void withdraw (double amount){
    if (amount <= balance){
        balance -= amount;
        System.out.println("Amount withdrawn: " + amount);
    else{
        System.out.println("Insufficient balance for withdrawal.");
        }
        displayBalance();
    }

@Override
public void checkMinimumBalance(){ }}

class CurrentAccount extends Account{
    private static final double MINIMUM_BALANCE = 1000.0;
    private static final double SERVICE_CHARGE = 50.0;

    public CurrentAccount (String customerName, String accountNumber, double
                                    initialBalance){
        super (customerName, accountNumber, "Current", initialBalance);}

@Override
public void withdraw (double amount){
    if (amount <= balance){
        balance -= amount;
        System.out.println("Amount withdrawn: " + amount);
        checkMinimumBalance();
    }
    else{
        System.out.println("Insufficient balance for withdrawal.");}
        displayBalance();
    }
```

```
@Override
public void checkMinimumBalance(){
        if(balance< MINIMUM_BALANCE){
            balance -= SERVICE_CHARGE;
            System.out.println("Balance below minimum. Service charge
                                imposed: " + SERVICE_CHARGE);
        }
    }
}

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

        System.out.println("Enter customer name:");
        String customerName = sc.nextLine();

        System.out.println("Enter account number: ");
        String accountNumber = sc.nextLine();

        System.out.println("Enter account type (savings/current): ");
        String accountType = sc.nextLine();

        System.out.println("Enter initial balance: ");
        double initialBalance = sc.nextDouble();

        Account account;
        if(accountType.equalsIgnoreCase("Savings")){
            account = new SavingsAccount(customerName, accountNumber,
                                initialBalance);}
        else if(accountType.equalsIgnoreCase("Current")){
            account = new CurrentAccount(customerName, accountNumber,
                                initialBalance);}
        else{
            System.out.println("Invalid account type.");
            sc.close();
            return;
        }
        int choice;
```

```
do{
    System.out.println("\n Bank Menu: ");
    System.out.println("1. Deposit");
    System.out.println("2. Display Balance");
    System.out.println("2. Display Balance");
    if (account instanceof SavingsAccount){
        System.out.println("3. Compute and Deposit Interest");}

    System.out.println("4. Withdraw");
    System.out.println("5. Exit");
    System.out.print("Enter your choice: ");
    choice = sc.nextInt();

    switch (choice){
        case 1: System.out.print("Enter deposit amount: ");
                double depositAmount = sc.nextDouble();
                account.deposit(depositAmount);
                break;
        case 2: account.displayBalance();
                break;
        case 3: if (account instanceof SavingsAccount){
                ((SavingsAccount) account).computeAndDepositInterest();
                }
                else{
                    System.out.println("Interest calculation not
                                        available for current accounts.");}
                break;
        case 4: System.out.print("Enter withdrawal amount: ");
                double withdrawalAmount = sc.nextDouble();
                account.withdraw(withdrawalAmount);
                break;
        case 5: System.out.println("Exiting the program.");
                break;
        default: System.out.println("Invalid choice. Try again.");
    }while (choice != 5);
    sc.close();
}
```

(14)

Observations

Enter customer
Snehanth C
Enter account
10000045
Enter account
Savings
Enter initial
10000

Bank Menu
1. Deposit
2. Display
3. compute
4. Withdr
5. Exit
Enter yo
Enter dep
Amount de
Current T

2
current

3
Interest
Current

4
Enter
Amount
Current

5
Exiting

Observations

Enter customer name:
Snehanth C
Enter account number:
10000045
Enter account type (Savings/Current):
Savings
Enter initial balance:
10000

Bank Menu:
1. Deposit
2. Display Balance
3. Compute and Deposit Interest
4. Withdraw
5. Exit
Enter your choice: 1
Enter deposit amount: 2000
Amount deposited: 2000.0
Current Balance: 12000.0

*2
Current balance: 12000.0

*3
Interest computed and added: 600.0
Current balance: 12600.0

*4
Enter withdrawal amount: 3000
Amount withdrawn: 3000.0
Current balance: 9600.0

*5
Exiting the program.

Enter customer name:
Shashank
Enter account number:
1000056
Enter account type (Savings/Current):
Current
Enter initial balance:
10000

Bank Menu:
1 Deposit
2. Display Balance
3. Withdraw
5. Exit
Enter your choice: 1
Amount deposited: 2000.0
Current Balance: 12000.0

**2
~~Display~~ Current balance: 12000.0

*3
Invalid choice. Try again

*4.
Enter withdrawal amount: 5000
Amount withdrawn: 5000.0
Current balance: 7000.0

*5
Exiting the Program.

CODE :

```
class Account{
   String customer_name;
   long account_no;
   String typeofAccount;
   double balance;
   Account(String customer_name,long account_no,String
typeAccount,double balance){
      this.customer_name=customer_name;
      this.account_no=account_no;
      this.typeofAccount=typeAccount;
      this.balance=balance;    }
   void deposites(double amount){
      this.balance+=amount;
      System.out.println("Amount of "+amount+" has been debited"); }
   void displayBalance(){
      System.out.println("The Balance Of The "+account_no+" and Name
"+customer_name+" is :"+balance);    }
   void withdraw(double amount){
      if(balance<amount){
         System.out.println("Insufficient Balance");    }else{
         this.balance-=amount;
         System.out.println("Amount of "+amount+" succesfully
withdrwn");}}}
class SavingAccount extends Account{
   double compound_interest=0.04;
   SavingAccount(String customername,long account_no){
```

```
    super(customername,account_no,"SAVINGS",0);}
  void compoundInterest(){
    double interest=balance*0.04;
    balance+=interest;
    System.out.println("Intereset deposited");}  }
  class CurrentAccount extends Account{
    boolean chequebook=true;
    double minimum_Balance=5000;
    double service_charge=50;
    CurrentAccount(String customername,long account_no){
      super(customername, account_no,"CURRENT", 5000);  }
    void withdraw(int amount){
      if(balance>amount){
        this.balance-=amount;
        System.out.println("Amounte of "+amount+" withdrawed
Succesfully");
        imposePenalty();    }else{
        System.out.println("Insuffescient  Balance");}}
    void imposePenalty(){
      if(balance<minimum_Balance){
        balance-=service_charge;
        System.out.println("Penalty Added");}    }  }
  public class Bank {
     public static void main(String[] args) {
      SavingAccount savingAccount=new
SavingAccount("sushanth",123456789 );
      savingAccount.displayBalance();
```

```
        savingAccount.withdraw(500);
        savingAccount.deposites(1000);
        savingAccount.deposites(1000);
        savingAccount.compoundInterest();
        savingAccount.withdraw(1000);
        savingAccount.displayBalance();
        System.out.println();
        CurrentAccount currentAccount=new
CurrentAccount("likhith",987654321 );
        currentAccount.displayBalance();
        currentAccount.withdraw(500);
        currentAccount.deposites(1000);
        currentAccount.deposites(1000);
        currentAccount.withdraw(1000);
        currentAccount.displayBalance(); }
    }
```

OUTPUT :

```
The Balance Of The 123456789 and Name sushanth is :0.0
Insufficient Balance
Amount of 1000.0 has been debited
Amount of 1000.0 has been debited
Intereset deposited
Amount of 1000.0 succesfully withdrwn
The Balance Of The 123456789 and Name sushanth is :1080.0

The Balance Of The 987654321 and Name likhith is :5000.0
Amounte of 500 withdrawed Succesfully
Penalty Added
Amount of 1000.0 has been debited
Amount of 1000.0 has been debited
Amounte of 1000 withdrawed Succesfully
The Balance Of The 987654321 and Name likhith is :5450.0
```

LABORATORY PROGRAM – 6

Create a package CIE which has two classes- Student and Internals. The class Personal has members like usn, name, sem. The class internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

OBSERVATION :

Q.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 5 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.

Ans

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

// CIE / Internals.java
package CIE;
public class Internals{
    private int[] InternalMarks = new int[5];
    public void setInternalMarks (int[] marks){
        if (marks.length == 5){
            System.arraycopy(marks, 0, internalMarks, 0, 5);}
        else{
            System.out.println("Please provide exactly 5 marks."); }
        }
public int[] getInternalMarks(){
        return InternalMarks; }
    }
```

```
// SEE / External.java
package SEE;
import CIE.Student;
public class External
    private int[]
    public External
        Super(
    }
    public void
        if (mar
            Syst
        else {
    }
    public int
        ret
    }
}
public int[]
    return
}
}

// Main.java
import CIE.*;
import SEE.*;
import java.
public class M
    public st
        Scan
        Sy
        int
        Ext
        In
```

```
// SEE/External.java
package SEE;
import CIE.Student;
public class External extends Student{
    private int[] seeMarks = new int[5];
    public External(String usn, String name, int sem){
        super(usn, name, sem);
    }
    public void setSeeMarks(int[] marks){
        if (marks.length == 5){
            System.arraycopy(marks, 0, seeMarks, 0, 5); }
        else { System.out.println("Please provide exactly 5 marks."); }
    }
    public int[] getSeeMarks(){
        return seeMarks;
    }
}
public int[] getSeeMarks(){
    return seeMarks;
}
}

// Main.java
import CIE.*;
import SEE.*;

import java.util.Scanner;

public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of students: ");
        int n = sc.nextInt();
        External[] students = new External[n];
        Internals[] internals = new Internals[n];
```

```
for (int i=0 ; i<n ; i++){
    sc.nextline();
    System.out.println("\n Enter details for student:"+
                        (i+1));

    System.out.print("Enter USN: ");
    String usn = sc.nextline();

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

    System.out.print("Enter Semester: ");
    int sem = sc.nextInt();

    Students [i] = new External(usn, name, sem);
    internals[i] = new Internals();

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

    System.out.println("Enter SEE marks for 5 courses: ");
    int[] seeMarks = new int[5];
    for (int j=0 ; j<5 ; j++){
        seeMarks[j] = sc.nextInt(); }
    students[i].setSeeMarks(seeMarks); }

    System.out.println("\n Final Marks of Students: ");
    for (int i=0 ; i<n ; i++){
        System.out.println("\n Student" + (i+1) + "(" +
            students[i].usn + ", " + student[i].name + "):");
```

```
int[] internalMarks = internals[i].getInternalMarks();
int[] seeMarks = students[i].name + ":");

int[] internalMarks = internals[i].getInternalMarks();
int[] seeMarks = students[i].getSeeMarks();

for (int j=0; j<5; j++){
    int finalMarks = internalMarks[j] + (seeMarks[j]/2);
    System.out.println("Course" + (j+1) + "Final Marks: " +
                        finalMarks); }
        }

    Sc.close();
    }
}
```

Observations

~~No. of students~~, ~~Student Details~~, ~~USN: 1BM21CS001~~, ~~Name: Alice~~, ~~Semester~~, ~~Internal Marks (out of 50 for~~

Enter the number of students : 2
Enter details for student 1 :
Enter USN: 1BM21CS001
Enter Name: Alice
Enter Semester: 5
Enter internal marks for 5 courses:
40 45 35 50 48
Enter SEE marks for 5 courses:
80 90 70 100 96
Enter details for student 2
Enter USN: 1BM21CS002
Enter Name: Bob
Enter Semester: 5

Enter internal marks for 5 courses
30 25 40 35 28
Enter SEE marks for 5 courses:
60 50 80 70 58

Final Marks of Students:
Student 1 (1BM21CS001, Alice):
Course 1 Final Marks : 80
Course 2 Final Marks : 90
Course 3 Final Marks : 70
Course 4 Final Marks : 100
Course 5 Final Marks : 96

Student 2 (1BM21CS002, Bob):
Course 1 Final Marks : 60
Course 2 Final Marks : 50
Course 3 Final Marks : 80
Course 4 Final Marks : 70
Course 5 Final Marks : 58

File : CIE/Student.java

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

File : CIE/Internal.java

```
package cie;
public class Internals {
    public int[] internalMarks = new int[5];
    public Internals(int[] marks) {
        if (marks.length == 5) {
            System.arraycopy(marks, 0, internalMarks, 0, 5);
        } else {
            System.out.println("Error: Please provide marks for exactly 5
courses.");
        }
    }
}
```

File : SEE/External.java

```
package see;
import cie.Student;
public class External extends Student {
    public int[] externalMarks = new int[5];
    public External(String usn, String name, int sem, int[] marks) {
        super(usn, name, sem);
        if (marks.length == 5) {
            System.arraycopy(marks, 0, externalMarks, 0, 5);
        } else {
            System.out.println("Error: Please provide marks for exactly 5
courses.");
        }
    }
}
```

File : FinalMarrks.java

```
import cie.*;
import see.*;
import java.util.Scanner;
public class FinalMarks {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of students: ");
        int n = sc.nextInt();
        Student[] students = new Student[n];
        Internals[] internals = new Internals[n];
```

```
        External[] externals = new External[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for student " + (i + 1) + ":");
            System.out.print("USN: ");
            String usn = sc.next();
            System.out.print("Name: ");
            String name = sc.next();
            System.out.print("Semester: ");
            int sem = sc.nextInt();
            System.out.println("Enter internal marks for 5 courses:");
            int[] internalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                internalMarks[j] = sc.nextInt();
            }
            System.out.println("Enter SEE marks for 5 courses:");
            int[] externalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                externalMarks[j] = sc.nextInt();
            }
            students[i] = new Student(usn, name, sem);
            internals[i] = new Internals(internalMarks);
            externals[i] = new External(usn, name, sem, externalMarks);
        }
        System.out.println("\nFinal Marks of Students:");
        for (int i = 0; i < n; i++) {
            System.out.println("Student: " + students[i].name + " (USN: " +
students[i].usn + ")");
```

```
            for (int j = 0; j < 5; j++) {
                int finalMarks = internals[i].internalMarks[j] +
externals[i].externalMarks[j] / 2;
                System.out.println("Course " + (j + 1) + ": " + finalMarks);
            }
        }
        sc.close();
    }
}
```

OUTPUT :

```
Enter details for student 1:
USN: 1RV23CS001
Name: John
Semester: 5
Enter internal marks for 5 courses:
18 19 20 17 16
Enter SEE marks for 5 courses:
70 60 80 90 50
Final Marks of Students:
Student: John (USN: 1RV23CS001)
Course 1: 53
Course 2: 49
Course 3: 60
Course 4: 62
Course 5: 41
```

LABORATORY PROGRAM – 7

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

OBSERVATION :

Q.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 is < 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.

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

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

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

```
public class ExceptionHandlingDemo {
    public static void main (String[] args) {
        ~~try~~ {try{
            System.out.println("Creating a Father object...");
            Father father = new Father(40);
            System.out.println("Father created with age: " + father.getAge());

            System.out.println("Creating a Son object...");
            Son son = new Son(40,20);
            System.out.println("Son created with age: " + son.sonAge);}
        ~~catch~~ catch(WrongAgeException e) {
                System.out.println("Exception caught: " + e.getMessage());
            }
        ~~try~~ {
            System.out.println("\n Testing invalid scenarios...");
            try {
                Father invalidFather = new Father(-10); }
            catch (WrongAgeException e) {
                System.out.println("Exception caught: " + e.getMessage());}
            try { Son invalidSon = new Son(30,40); }
            catch (WrongAgeException e){
                System.out.println("Exception caught: " + e.getMessage());
            }
        }
    }
```

Observation

Creating a Father object...
Father created with age: 40
Creating a Son object...
Son created with age: 20

Testing invalid scenarios...
Exception caught: Father's age cannot be negative.
Exception caught: Son's age cannot be greater than or equal to Father's age.

CODE :

```
class WrongAgeException extends Exception {
   public WrongAgeException(String message) {
      super(message); } }
class Father {
   int fatherAge;
   public Father(int age) throws WrongAgeException {
      if (age < 0) {
         throw new WrongAgeException("Father's age cannot be
negative!");
      }
      this.fatherAge = age;
      System.out.println("Father's Age: " + fatherAge); } }
class Son extends Father {
   int sonAge;
   public Son(int fatherAge, int sonAge) throws WrongAgeException {
      super(fatherAge);
      if (sonAge < 0) {
         throw new WrongAgeException("Son's age cannot be negative!");
      }
      if (sonAge >= fatherAge) {
         throw new WrongAgeException("Son's age cannot be greater than
or equal to father's age!");
      }
      this.sonAge = sonAge;
      System.out.println("Son's Age: " + sonAge); } }
public class ExceptionMain {
```

```
public static void main(String[] args) {
    java.util.Scanner sc = new java.util.Scanner(System.in);
    try {
        System.out.print("Enter Father's Age: ");
        int fatherAge = sc.nextInt();
        System.out.print("Enter Son's Age: ");
        int sonAge = sc.nextInt();
        Son son = new Son(fatherAge, sonAge);
    } catch (WrongAgeException e) {
        System.out.println("Exception: " + e.getMessage());
    } catch (Exception e) {
        System.out.println("Unexpected Exception: " + e);} } }
```

OUTPUT :

```
PS D:\3rd sem\OOJ JAVA\Git-hub> java ExceptionMain
Enter Father's Age: 40
Enter Son's Age: -10
Father's Age: 40
Exception: Son's age cannot be negative!
PS D:\3rd sem\OOJ JAVA\Git-hub> java ExceptionMain
Enter Father's Age: 40
Enter Son's Age: 50
Father's Age: 40
Exception: Son's age cannot be greater than or equal to father's age!
PS D:\3rd sem\OOJ JAVA\Git-hub> java ExceptionMain
Enter Father's Age: -40
Enter Son's Age: 20
Exception: Father's age cannot be negative!
```

LABORATORY PROGRAM – 8

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

OBSERVATION :

Q. 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.

Ans:

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

class CSEDisplay extends Thread {
    public void run() {
        while (true) {
            System.out.println("CSE");
            try {
                Thread.sleep(2000); }
            catch (InterruptedException e) {
                e.printStackTrace(); }
        }
    }
}

public class MultiThreadingExample {
    public static void main(String[] args) {
        BMSDisplay bmsThread = new BMSDisplay();
        CSEDisplay cseThread = new CSEDisplay();
        bmsThread.start();
        cseThread.start();
    }
}
```

Observations

BMS College of Engineering

CSE
CSE
CSE
CSE
CSE

BMS College of Engineering

CSE
CSE
CSE
CSE
CSE

BMS College of Engineering.

CODE :

```
class BmsThread extends Thread{
   public void run(){
      try{
         while (true) {
            System.out.println("BMS college of Engineering");
            Thread.sleep(10000);
         }
      }catch(InterruptedException e){
         System.out.println(e);
      }
   }
}
 class CseThread implements Runnable{
   Thread t;

   public void run(){
      try{
         while (true) {
            System.out.println("Cse");
            Thread.sleep(2000);
         }
      }catch(InterruptedException e){
         System.out.println(e);
      }
   }
```

```
}
public class DemoThread  {
   public static void main(String[] args) {
      BmsThread b=new BmsThread();
      Runnable cse=new CseThread();
      Thread t1=new Thread(cse);
      b.start();
      t1.start();;
   }
}
```

OUTPUT :

```
BMS college of Engineering
Cse
Cse
Cse
Cse
Cse
BMS college of Engineering
Cse
Cse
Cse
Cse
Cse
BMS college of Engineering
Cse
Cse
Cse
Cse
Cse
```

LABORATORY PROGRAM – 9

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

OBSERVATION :

~~Q.9 Write a program which creates two threads, one thread displaying~~

Q.9. Write a program that creates a user interface to perform integer division. The user enters two numbers in text format, N1 and N2. 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 a NumberFormatException. If Num2 were zero, the program would throw an ArithmeticException displaying the exception in a message dialog box.

A

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

public class DivisionApp {
    public static void main(String[] args){
        JFrame frame = new JFrame("Integer Division");
        JTextField num1Field = new JTextField();
        JTextField num2Field = new JTextField();
        JTextField resultField = new JTextField();
        JButton divideButton = new JButton("Divide");

        num1Field.setBounds(50,50,100,30);
        num2Field.setBounds(50,100,100,30);
        resultField.setBounds(50,150,100,30);
        resultField.setEditable(false);
        divideButton.setBounds(50,200,100,30);

        divideButton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
            try{
                int num1 = Integer.parseInt(num1Field.getText());
                int num2 = Integer.parseInt(num2Field.getText());
                int result = num1/num2;
                resultField.setText(String.valueOf(result));}
```

```
catch (NumberFormatException ex){
    JOptionPane.showMessageDialog(frame,"Please enter
        valid integers", "Error", JOptionPane.ERROR_
        MESSAGE);}

catch (ArithmeticException ex) {
    JOptionPane.showMessageDialog(frame,"Cannot
        divide by zero ", "Error",JOptionPane.ERROR_
        MESSAGE);

        }

    }

    });

frame.add(num1Field);
frame.add(num2Field);
frame.add(resultField);
frame.add(divideButton);
frame.setSize(300,300);
frame.setLayout(null);
frame.setDefaultCloseOperation(
    JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);
}
}
```

Observat

Divisio

Num & 1:

Observation:

Division of Integers.

Num 1: 100 Num 2: 20 Result Result: 100/20 = 5.0

CODE :

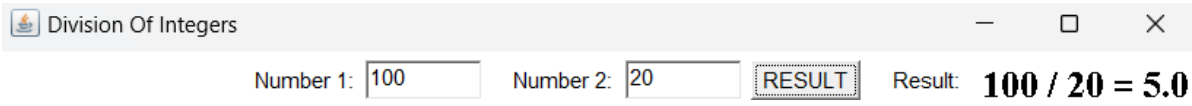
```
import java.awt.*;
import java.awt.event.*;

public class Maindemo extends Frame implements ActionListener {
   TextField num1, num2;
   Button dResult;
   Label outResult;
   String out = "";
   double resultNum;
   int flag = 0;
   public Maindemo() {
      setLayout(new FlowLayout());
      dResult = new Button("RESULT");
      Label number1 = new Label("Number 1:", Label.RIGHT);
      Label number2 = new Label("Number 2:", Label.RIGHT);
      num1 = new TextField(5);
      num2 = new TextField(5);
      outResult = new Label("Result:", Label.RIGHT);
      add(number1);
      add(num1);
      add(number2);
      add(num2);
      add(dResult);
```

```
    add(outResult);
    num1.addActionListener(this);
    num2.addActionListener(this);
    dResult.addActionListener(this);
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent we) {
            System.exit(0);
        }        });
}
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("Division by zero"):}
            resultNum = (double) n1 / n2;
            out = n1 + " / " + n2 + " = " + resultNum;
        }
    } catch (NumberFormatException e1) {
        flag = 1;
        out = "Number Format Exception! Please enter valid integers.";
    } catch (ArithmeticException e2) {
        flag = 1;
        out = "Divide by 0 Exception! " + e2.getMessage();}
```

```
        repaint();}
    public void paint(Graphics g) {
        Font customFont = new Font("Serif", Font.BOLD, 20);
    g.setFont(customFont);
        if (flag == 0) {
            g.drawString(out, outResult.getX() + outResult.getWidth() + 10,
outResult.getY() +20);
        } else {
            g.drawString(out, 100, 200);
            flag = 0;
        } }
    public static void main(String[] args) {
        Maindemo dm = new Maindemo();
        dm.setSize(new Dimension(800, 400));
        dm.setTitle("Division Of Integers");
        dm.setVisible(true);
    }
}
```

OUTPUT :


Division Of Integers
Number 1:
100
Number 2:
20
RESULT
Result:
100 / 20 = 5.0

LABORATORY PROGRAM – 10

Demonstrate Inter process Communication and deadlock.

OBSERVATION :

Q.10. Demonstrate Inter-process Communication & deadlock.

Ans: a)

```
import java.io.PipedInputStream;
import java.io.PipedOutputStream;

public class IPCExample {
    public static void main (String[] args) {
        try {
            PipedInputStream input = new PipedInputStream();
            PipedOutputStream output = new PipedOutputStream
                                                (input);

            Thread sender = new Thread (() -> {
                try {
                    output.write("Hello from Sender", getBytes
                                                ());
                    output.close(); }
                catch (Exception e) {
                    e.printStackTrace(); }
            });

            Thread reciever = new Thread (() -> {
                try {
                    int data;
                    StringBuilder message = new StringBuilder();
                    while ((data = input.read()) != -1) {
                        message.append ((char) data);
                    }
                    input.close();
                    System.out.println("Received: " + message);
                catch (Exception e) {
                    e.printStackTrace(); }
            });
```


b) public c

```
            sender.start();
            receiver.start();

            sender.join();
            receiver.join();
        }
        catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

b)

```
public class DeadlockExample {
    public static void main (String[] args){
        final Object resource1 = "Resource 1";
        final Object resource2 = "Resource 2";

        Thread thread1 = new Thread(()-> {
            synchronized (resource1){
                System.out.println("Thread1: Locked
                        resource 1");
                try { Thread.sleep(100); } catch
                catch (InterruptedException e){}
                synchronized(resource2){
                    System.out.println("Thread 1: Locked
                        resources 2");
                }
            }
        })
        Thread thread2 = new Thread(()-> {
            synchronized (resource2){
                System.out.println("Thread 2: Locked resource
                        2");
```

CODE_:

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

```
        System.out.println("Put: " + n);
        System.out.println("\nNotifying Consumer\n");
        notify();  } }
class Producer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start(); }
    public void run() {
        int i = 0;
        while (i < 15) {
            q.put(i++); }
        System.out.println("Producer finished."); } }
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++; }
        System.out.println("Consumer  finished.");
    } }
```

```
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 :

```
Press Control-C to stop.
Put: 0

Notifying Consumer


Producer waiting

Got: 0

Notifying Producer

Put: 1
Consumed: 0

Notifying Consumer


Producer waiting

Got: 1

Notifying Producer

Consumed: 1
Put: 2

Notifying Consumer


Producer waiting

Got: 2

Notifying Producer
```

```
try { Thread.sleep(100); } catch (InterruptedException
        e){}
synchronized (resource1){
        System.out.println("Thread 2: Locked
                                resource 1");
        }
    }
  });
  thread1.start();
  thread2.start();
 }
}
```

Observation:-

10a) Press Control-c to Stop

Put : 0

Notifying Consumer

Producer Waiting

Got : 0

Notifying Procedure

Put : 1

Consumed : 0

Notifying Consumer

~~Producer~~

Producer waiting

Got : 1

Notifying Procedurer

Consumed : 1

Put : 2

Notifying Consumer

Producer waiting

Got : 2

Notifying Producer

10.b) MainThre

RacingTh

RacingTh

MainThre

10. b) MainThread entered A.foo

RacingThread entered B.bar

RacingThread trying to call A.last()

MainThread trying to call B.last()

CODE :

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

```
        }
        System.out.println(name + " trying to call A.last()");
        a.last();
    }
    synchronized void last() {
        System.out.println("Inside B.last");
    }
}
class Deadlock implements Runnable {
    A a = new A();
    B b = new B();

    Deadlock() {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b);
        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();
    }
```

}

OUTPUT :

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