VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SNEHA S BHAIRAPPA (1BM23CS333)

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 **SNEHA S BHAIRAPPA (1BM23CS333),** 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	**Dr. Kavitha Sooda**
Associate Professor,	Professor and Head,
Department of CSE,	Department of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

INDEX

Sl. No.	Date	Experiment Title	Page No.
1	26/09/2024	Quadratic Equation Solver	4
2	03/10/2024	Student SGPA Calculator	8
3	19/10/2024	Book &Bookstore Classes	12
4	24/10/2024	Area of Different Shapes	19
5	07/11/2024	Bank Management System	27
6	14/11/2024	CIE &SEE Packages	39
7	21/11/2024	Exception Handling	48
8	28/11/2024	Thread Program	54
9	19/12/2024	Division of Two Numbers(open ended)	58
10	19/12/2024	Interprocess Communication and Deadlock(open ended)	65

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.

Q: Java program to solve a quadratic equation. 26/09/2024

```
import java.util.Scanner;
public class QuadraticEquationSolver{
    public static void main(String[]args){
        Scanner scanner = new Scanner(System.in);
    // Input coefficients
    System.out.println("Enter coefficient a:");
    double a = scanner.nextDouble();
    System.out.println("Enter coefficient b:");
    double b = scanner.nextDouble();
    System.out.println("Enter coefficient c:");
    double c = scanner.nextDouble();
  // Calculate the discriminant
  double discriminant = b*b - 4*a*c;
  // Determine the nature of the roots
  if (discriminant > 0){
      double root1 = (-b + Math.sqrt(discriminant))/(2*a);
      double root2 = (-b - Math.sqrt(discriminant))/(2*a);
      System.out.println("The roots are real and different");
      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 roots are real & same");
      System.out.println("Root: " +root);
  }
```

```
else {
    double realpart = -b / (2*a);
    double imaginarypart = Math.sqrt(-discriminant) / (2*a);
    System.out.println("The roots are complex and different");
    System.out.println("Root 1: " + realPart + " + " + imaginarypart
                                    + "i");
    System.out.println("Root 2: " + realpart + " -" + imaginarypart + "i");
    }
    scanner.close();
    }
}
```

Output:

```
Enter coefficient a: 1
Enter coefficient b: 1
Enter coefficient c: 1
The roots are complex and different.
Root 1: -0.5 + (0.8660254037844386)i
Root 2: -0.5 - (0.8660254037844386)i
```

24/10/24

import

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

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

        // Calculate the discriminant
        double discriminant = b * b - 4 * a * c;

        // Determine the nature of the roots
        if (discriminant > 0) {
            // Two distinct real roots
            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            System.out.println("The roots are real and different.");
            System.out.println("Root 1: " + root1);
            System.out.println("Root 2: " + root2);
        } else if (discriminant == 0) {
            // One real root
            double root = -b / (2 * a);
            System.out.println("The roots are real and the same.");
            System.out.println("Root: " + root);
        } else {
            // Complex roots
            double realPart = -b / (2 * a);
            double imaginaryPart = Math.sqrt(-discriminant) / (2 * a);
            System.out.println("The roots are complex and different.");
            System.out.println("Root 1: " + realPart + " + " + "("+imaginaryPart +")"+ "i");
            System.out.println("Root 2: " + realPart + " - " + "("+ imaginaryPart + ")"+ "i");
        }

        scanner.close();
    }
}
```

```
C:\1BM23CS333>javac Book.java

C:\1BM23CS333>java QuadraticEquationSolver
Enter coefficient a: 1
Enter coefficient b: 1
Enter coefficient c: 1
The roots are complex and different.
Root 1: -0.5 + (0.8660254037844386)i
Root 2: -0.5 - (0.8660254037844386)i

C:\1BM23CS333>
```

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.

Java program to calculate student SGPA. 3/10/2024

```
import java.util.Scanner;
public class SGPA_calc {
    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the no of subjects: ");
        int subj = scanner.nextInt();
        double[] credits = new double[subj];
        double[] grades = new double[subj];

        for (int i=0; i<subj; i++) {
        s.o.p("Enter credit hours for subject "+(i+1)+": ");
        credits[i] = scanner.nextDouble();
        s.o.p("Enter grade for subject "+(i+1)+"(on a scale of 0
                to 10): ");
        grades[i] = scanner.nextDouble();
    }
    double totalcredits = 0;
    double weightedgradesum = 0;
    for (int i=0; i<subj; i++) {
        weightedgradesum += grades[i] * credits[i];
        totalcredits += credits[i];
    }
    double sgpa = weightedgradesum / totalcredits;
    s.o.p("Your SGPA is: %.2f%n", sgpa);
    scanner.close();
}
```

Output:

```
Enter the number of subjects : 4
Enter credit for subject 1 : 4
Enter grade for subject 1 : 10
Enter credit for subject 2 : 3
Enter grade for subject 2 : 9
Enter credit for subject 3 : 2
Enter grade for subject 3 : 9
Enter credit for subject 4 : 1
Enter grade for subject 4 : 10
Your SGPA is : 9.50
```

24/10/24

```
import java.util.Scanner;

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

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

        // Arrays to store credits and grades
        double[] credits = new double[numSubjects];
        double[] grades = new double[numSubjects];

        // Input credits and grades
        for (int i = 0; i < numSubjects; i++) {
            System.out.print("Enter credit hours for subject " + (i + 1) + ": ");
            credits[i] = scanner.nextDouble();
            System.out.print("Enter grade for subject " + (i + 1) + " (on a scale of 0 to 10): ");
            grades[i] = scanner.nextDouble();
        }

        // Calculate SGPA
        double totalCredits = 0;
        double weightedGradesSum = 0;

        for (int i = 0; i < numSubjects; i++) {
            weightedGradesSum += grades[i] * credits[i];
            totalCredits += credits[i];
        }

        double sgpa = weightedGradesSum / totalCredits;
```

```
        // Display the SGPA
        System.out.printf("Your SGPA is: %.2f%n", sgpa);

        scanner.close();
    }
}
```

```
C:\1BM23CS333>javac Book.java

C:\1BM23CS333>java SGPA_Calculator
Enter the number of subjects: 4
Enter credit hours for subject 1: 4
Enter grade for subject 1 (on a scale of 0 to 10): 9
Enter credit hours for subject 2: 3
Enter grade for subject 2 (on a scale of 0 to 10): 10
Enter credit hours for subject 3: 4
Enter grade for subject 3 (on a scale of 0 to 10): 10
Enter credit hours for subject 4: 2
Enter grade for subject 4 (on a scale of 0 to 10): 9
Your SGPA is: 9.54

C:\1BM23CS333>
```

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.

Java program for the Book class & Bookstore class 19/10/2024.

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
class Book {
    String name;
    String author;
    double price;
    int numPages;
    //Constructor
    Book(String name, String author, double price, int numPages){
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }
    void setName(String name){
        this.name = name;
    }
    void setAuthor(String author){
        this.author = author;
    }
    void setPrice(double price){
        this.price = price;
    }
    void setNumPages(int numPages){
        this.numPages = numPages;
    }
```

```
String getName() {
    return name;
}
String getAuthor() {
    return author;
}
double getPrice() {
    return price;
}
int getNumPages() {
    return numPages;
}
// toString method
@Override
public String toString() {
    return "Book{" +
            "Name = '" + name + '\'' +
            ", Author = '" + author + '\'' +
            ", Price = " + price +
            ", Number of Pages = " + numPages +
            '}';
    }
}
class BookStore {
    psvm (String[] args) {
        Scanner sc = new Scanner(System.in);
        List<Book> books = new ArrayList<>();
        S.o.p("Enter the no of books:");
        int n = sc.nextInt();
```

```
        sc.nextLine();
        for (int i=0; i<n; i++) {
            s.o.p("Enter details for book" + (i+1) + ":");
            s.o.p ("Name:");
            String name= sc.nextLine();
            s.o.p ("Author:");
            String author= sc.nextLine();
            s.o.p ("Price:");
            double price = sc.nextDouble();
            s.o.p("Number of pages:");
            int numPages = sc.nextInt();
            sc.nextLine();
            Book book= new Book(name, author, price, numPages);
            books.book add (book);
        }
        s.o.p ("\n Book Details:");
        for (Book book: books) {
            s.o.p (book);
        }
        sc.close();
    }
}
```

Question:

Create a class Book which contains 4 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.

Output:

Area of Rectangle : 80
Area of Triangle : 12.0
Area of Circle : 28.274333882308138

Enter the no^r of books : 2
Enter details for book 1 :
Name : abc
Author : xyz
Price : 150
No^r of Pages : 200
Enter details for book 2 :
Name : def
Author : pqr
Price : 130
No^r of Pages : 100
Book Details :
Book { Name='abc', Author='xyz', Price=150.0, No^r of pages=200 }
Book { Name='def', Author='pqr', Price=130.0, No^r of pages=100 }

RS
24/10/24

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

class Book {
    private String name;
    private String author;
    private double price;
    private int numPages;

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

    // Setters
    void setName(String name) {
        this.name = name;
    }

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

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

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

    // Getters
    String getName() {
        return name;
    }

    String getAuthor() {
        return author;
    }

    double getPrice() {
        return price;
    }

    int getNumPages() {
        return numPages;
    }
```

```
    // toString method
    @Override
    public String toString() {
        return "Book{" +
                "Name='" + name + '\'' +
                ", Author='" + author + '\'' +
                ", Price=" + price +
                ", Number of Pages=" + numPages +
                '}';
    }
}

class BookStore {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<Book> books = new ArrayList<>();

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

        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(); // Consume newline

            Book book = new Book(name, author, price, numPages);
            books.add(book);
        }

        System.out.println("\nBook Details:");
        for (Book book : books) {
            System.out.println(book);
        }

        scanner.close();
    }
}
```

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.

Java program for finding area of different shapes 24/10/2024

```
abstract class Shape{
    int dim1;
    int dim2;
    Shape(int dim1, int dim2){
      this.dim1 = dim1;
      this.dim2= dim2;
    abstract void printArea();
}
class Rectangle extends Shape{
    Rectangle(int width, int height){
        super(width, height);
    }
    @Override
    void printArea(){
      int area= dim1 * dim2;
      System.out.println("Area of Rectangle: "+area);
    }
}
class Triangle extends Shape{
    Triangle(int base, int height){
        super(base, height);
    }
    @Override
    void printArea(){
      double area= 0.5* dim1* dim2;
      System.out.println("Area of Triangle:"+area);
    }
}
```

```
class Circle extends Shape{
    Circle (int radius){
        super(radius, 0);
    }
    @Override
    void printArea (){
        double area = Math.PI * dim1 * dim1;
        System.out.println("Area of Circle: "+area);
    }
}
class Main {
    public static void main (String[] args) {
        Shape rectangle = new Rectangle (5,10);
        Shape triangle = new Triangle (4,6);
        Shape circle = new Circle (3);
        rectangle.printArea();
        triangle.printArea();
        circle.printArea();
    }
}
```

Question:

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.

```
class Main {
    public static void main (String[] args){
        Scanner sc = new Scanner(System.in);
        s.o.p("Choose a shape to calculate the area (1: Rectangle,
                    2: Triangle, 3: Circle): ");
        int choice = sc.nextInt();
        Shape shape = null;
        switch (choice){
            case 1:
                s.o.p ("Enter width of the rectangle, ");
                int width = sc.nextInt();
                s.o.p ("Enter height of the rectangle: ");
                int height = sc.nextInt();
                shape = new Rectangle (width, height);
                break;
            Case 2:
                s.o.p("Enter base of the triangle: ");
                int base = sc.nextInt();
                s.o.p("Enter height of the triangle: ");
                int height = sc.nextInt();
                shape = new Triangle (base, height);
                break;
            case 3:
                s.o.p("Enter radius of the circle: ");
                int radius = sc.nextInt();
                shape = new Circle (radius);
                break;
```

```
        default:
            s.o.p("Invalid choice");
            break;
        }
        if (shape != null) {
            shape.printArea();
        }
        scanner.close();
    }
```

Output:

```
choose a shape to calculate the area (1: Rectangle, 2: Triangle,
3: Circle): 1
Enter width of the rectangle: 12
Enter height of the rectangle: 44
Area of Rectangle: 528
```

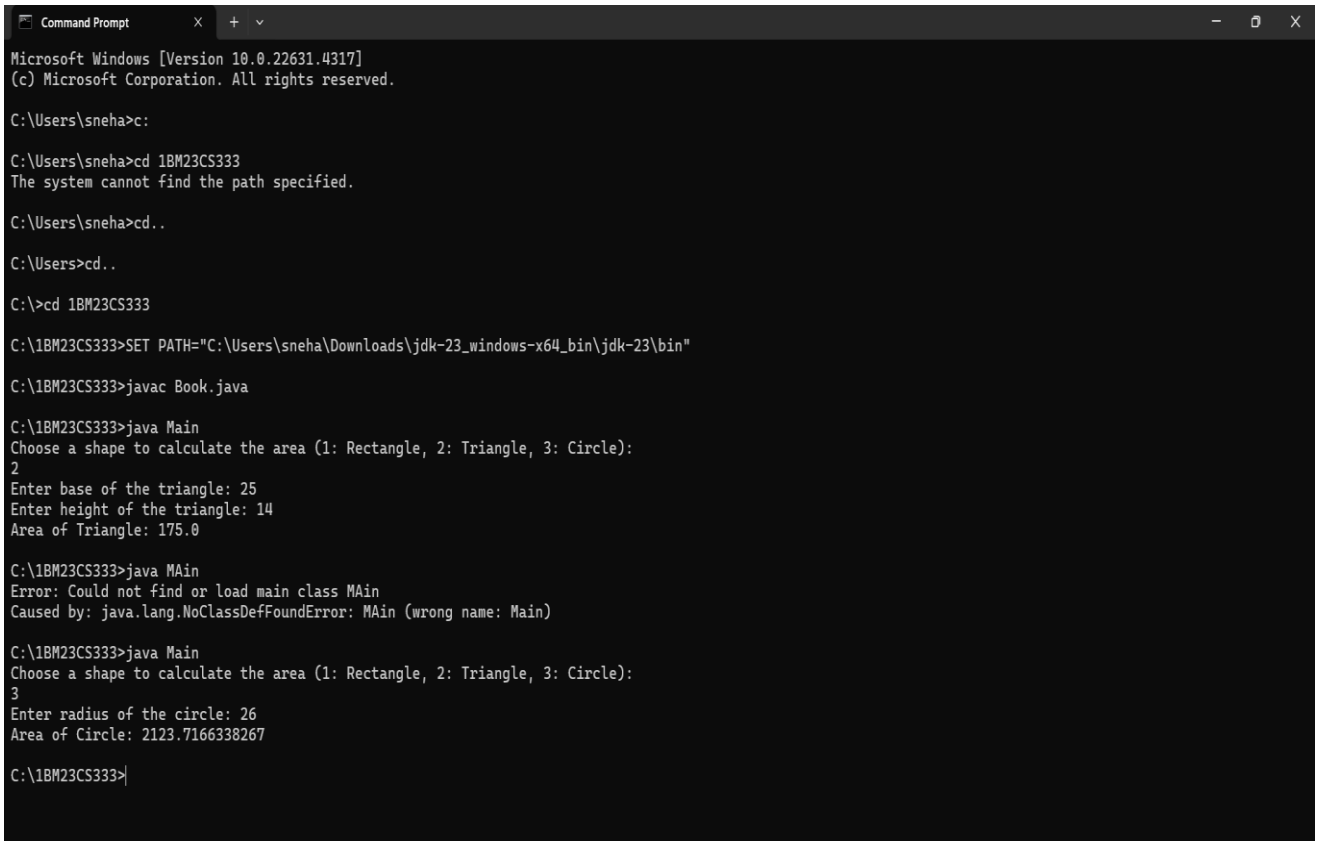
RS
24/10/24

```
import java.util.Scanner;
abstract class Shape {
    protected int dimension1;
    protected int dimension2;

    // Constructor to initialize dimensions
    public Shape(int dimension1, int dimension2) {
        this.dimension1 = dimension1;
        this.dimension2 = dimension2;
    }

    // Abstract method to print the area
    public abstract void printArea();
}

class Rectangle extends Shape {
    public Rectangle(int width, int height) {
        super(width, height);
    }

    @Override
    public void printArea() {
        int area = dimension1 * dimension2; // width * height
        System.out.println("Area of Rectangle: " + area);
    }
}

class Triangle extends Shape {
    public Triangle(int base, int height) {
        super(base, height);
    }
```

```
    @Override
    public void printArea() {
        double area = 0.5 * dimension1 * dimension2; // 0.5 * base * height
        System.out.println("Area of Triangle: " + area);
    }
}

class Circle extends Shape {
    public Circle(int radius) {
        super(radius, 0); // dimension2 is not used for Circle
    }

    @Override
    public void printArea() {
        double area = Math.PI * dimension1 * dimension1; // π * radius^2
        System.out.println("Area of Circle: " + area);
    }
}
class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Choose a shape to calculate the area (1: Rectangle, 2: Triangle, 3: Circle): ");
        int choice = scanner.nextInt();

        Shape shape = null;

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

            case 2:
                System.out.print("Enter base of the triangle: ");
                int base = scanner.nextInt();
                System.out.print("Enter height of the triangle: ");
```

```
            int triangleHeight = scanner.nextInt();
            shape = new Triangle(base, triangleHeight);
            break;

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

        default:
            System.out.println("Invalid choice.");
            break;
    }

    if (shape != null) {
        shape.printArea();
    }

    scanner.close();
  }
}
```

OUTPUT



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.

Develop a java program to create a class Bank that 07/11/2024
maintains 2 kinds of account for its customers, one called savings account & the other current account. The savings account provides compound interest & withdrawal facilities but no cheque book facility. The current acc provid cheque book facility but no interest. Current acc holders should also maint -in a minimum balance & if the balance falls below this level, a service charg is imposed. Create a class Account that stores customer name, account num -ber and type of account. From this derive the classes CurAcct & SavAcct to make them more specific to their requirements. Include the necessary methods.

```
import java.util.Scanner;

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

    public Accoant ( String customerName, String accountNumber,
    String accType, double initialBalance) {
     this.customerName = customerName;
     this.accountNumber = accountNumber;
     this.accountType = accountType;
     this.balance = initialBalance;
     public void deposit (double amount) {
          balance += amount;
      s.o.p ( " Deposit successful ! Current balance: " + balance);
      }
    public void displayBalance () {
        s.o.p ("Account balance: " + balance);
    }
```

```
public void withdraw (double amount){
        if ( balance >= amount){
                balance -= amount;
        S.o.p ( "Withdrawal successful ! Current balance: "+
                balance); }
         else {
           s.o.p ( "Insufficient balance ! Withdrawal failed!");
         }
public String getAccountType (){
        return accountType ; }

class SavAcct extends Account {
    private static final double interestRate = 0.04;
    public SavAcct (String customerName, String accountNumber,
            double initialBalance) {
      super(customerName, accountNumber, "Savings". initialBal)
     }
    public void computeInterest () {
       double interest = balance + interestrate ;
        balance += interest;
       s.o.p ("Interest of " + interest + " has been added to
              your account. New balance: " + balance);}
    }

class CurAcct extends Account {
    private static final double MinBalance = 500;
    private static final double penalty = 50 ;
```

import

```
acc = new SavAcct(custName, accnum, initialDeposit);
}
else if (accountChoice == 2){
  s.o.p ("Enter initial deposit for current account: ");
  double initialDeposit = sc.nextDouble();
  acc = new CurrAcct (custName, accnum, initialDeposit);
}
else {
  s.o.p ("Invalid choice! Exiting Program.");
  return; }

boolean running = true;
while (running) {
  s.o.p ("In Bank Operations Menu: ");
  s.o.p ("1. Deposit");
  s.o.p ("2. Withdraw");
  s.o.p ("3. Display Balance");
  s.o.p ("4. Compute Interest (savings accounts only)");
  s.o.p ("5. Check and apply min balance penalty (Current
          accounts only)");
  s.o.p ("6. Exit");
  s.o.p ("Enter your choice: ");
  int choice = sc.nextInt();

  switch (choice){
  case 1: s.o.p ("Enter deposit amount: ");
          double depositAmount = sc.nextDouble;
          account.deposit (depositAmount);
          break;
```

```
case 2 : s.o.p ("Enter withdrawal amount: ");
         double withdrawAmount = sc.nextDouble();
         account.withdraw (withdrawAmount);
         break;
case 3: account.displayBalance();
         break;
case 4: if (account instanceof SavAcct){
         ((SavAcct)account).computeInterest();}
        else{
         s.o.p("Interest can only be computed for Savings
                account.");
        }
        break;
case 5; if (account instanceof CurAcct){
         ((CurAcct)account).computeInterest();
        }
        else{
         s.o.p("Interest can only be computed for Savings
                account");
        }
        break;
            ((CurAcct)account).checkMinBal();
        }
        else{
         s.o.p ("Minimum balance check can only be
           applied to Current account");
        }
        break;
case 6: s.o.p("Exiting program.");
         running = false;
         break;
```

```
        default: S.O.p ("Invalid choice! Please select a valid choice");
        }
    }
    sc.close();
    }
}
```

Output:

```
Enter customer name: Abc
Enter account type (1 for Savings, 2 for Current): 2
Enter account number: 45697
Enter initial deposit for Current account: 4000

Bank operations Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest (Savings accounts only)
5. Check and apply minimum balance penalty (Current account only)
6. Exit
Enter your choice: 3
Account balance: 4000.0

Enter your choice: 4
Interest can only be computed for Savings account

Enter your choice: 2
Enter withdrawal amount: 2000
Withdrawal successful! Current balance: 2000.0

Enter withdrawal amount: 1800
Withdrawal successful! Current balance: 200
Enter your choice: 5
Balance is below minimum. A penalty of 50.0 has been charged.
New balance: 150.0.
```

Enter your choice: 1
Enter deposit amount: 3000
Deposit successful! Current balance: 3150

Rs
07/11/24

```
java.util.Scanner;

// Base class for Account
class Account {
    protected String customerName;
    protected String accountNumber;
    protected double balance;
    protected String accountType;

    // Constructor to initialize account details
    public Account(String customerName, String accountNumber, String accountType, double initialBalance) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = initialBalance;
    }

    // Method to deposit money into account
    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposit successful! Current balance: " + balance);
    }

    // Method to display current balance
    public void displayBalance() {
        System.out.println("Account balance: " + balance);
    }

    // Method to withdraw money from the account
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
```

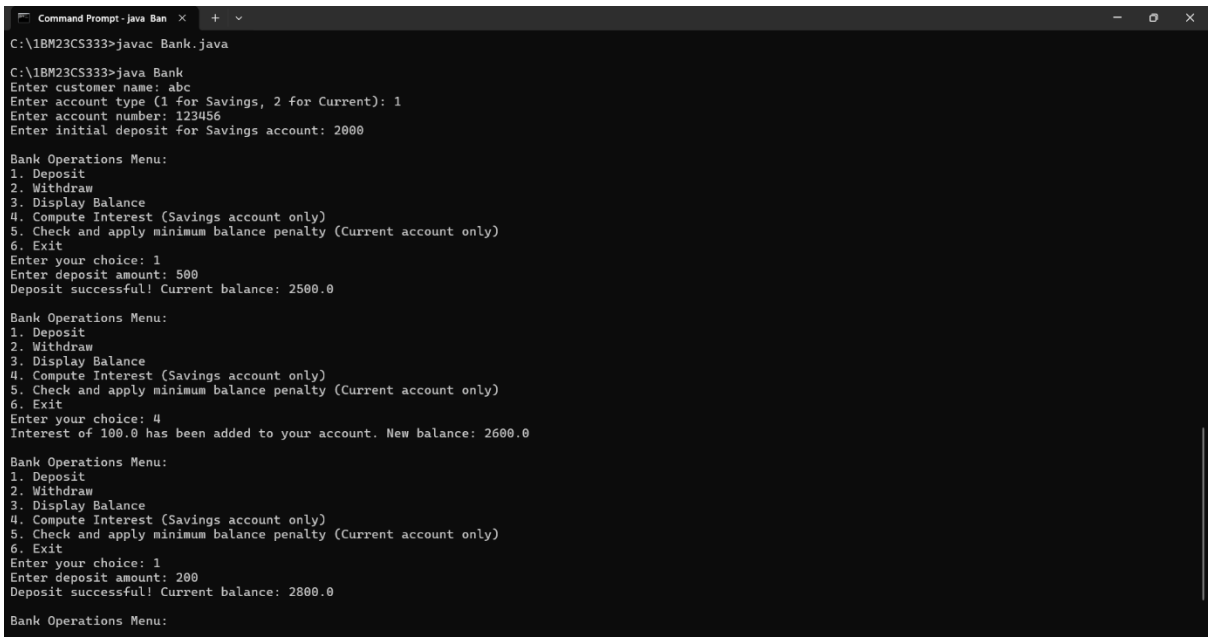
```
            System.out.println("Withdrawal successful! Current balance: " + balance);
        } else {
            System.out.println("Insufficient balance! Withdrawal failed.");
        }
    }

    // Method to get account type (for displaying)
    public String getAccountType() {
        return accountType;
    }
}

// Savings account class extends Account
class SavAcct extends Account {
    private static final double interestRate = 0.04; // 4% interest rate for savings account

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

    // Method to compute and deposit interest
    public void computeInterest() {
        double interest = balance * interestRate;
        balance += interest;
        System.out.println("Interest of " + interest + " has been added to your account. New balance: " +
balance);
    }
}

// Current account class extends Account
class CurAcct extends Account {
    private static final double MIN_BALANCE = 500; // Minimum balance required for Current Account
```

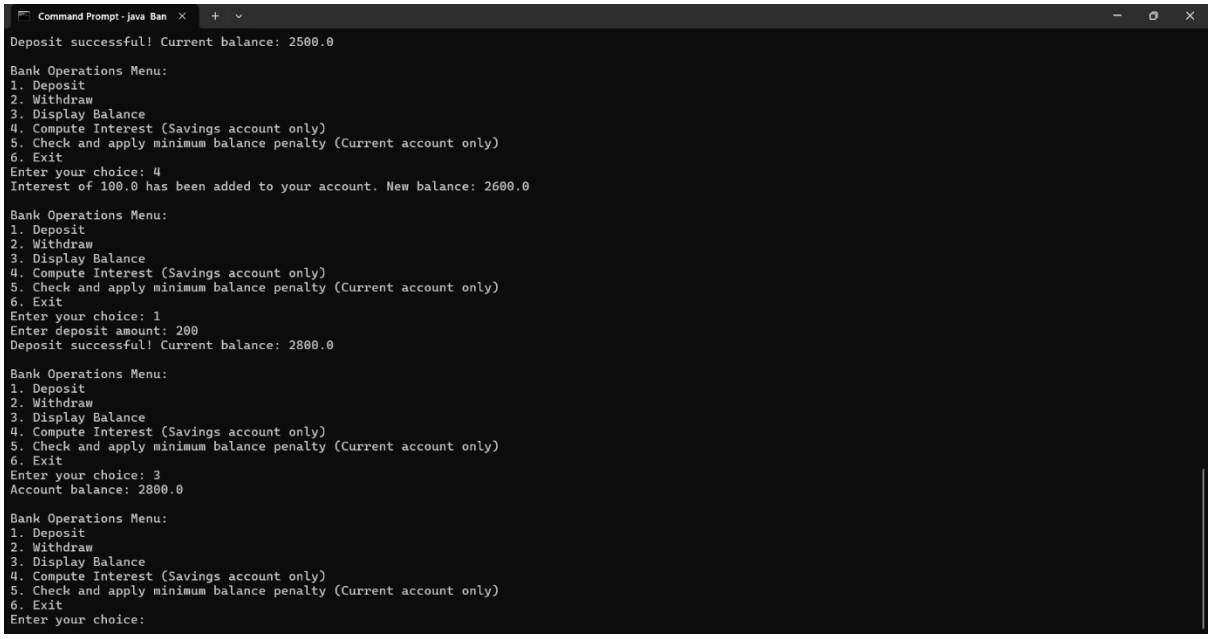
```
    private static final double PENALTY = 50; // Penalty fee for falling below minimum balance

    // Constructor for Current Account
    public CurAcct(String customerName, String accountNumber, double initialBalance) {
        super(customerName, accountNumber, "Current", initialBalance);
    }

    // Method to check and apply penalty for low balance
    public void checkMinimumBalance() {
        if (balance < MIN_BALANCE) {
            balance -= PENALTY;
            System.out.println("Balance is below minimum. A penalty of " + PENALTY + " has been charged. New balance: " + balance);
        }
    }
}

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

        // Create sample accounts (in real applications, these would be dynamic)
        System.out.print("Enter customer name: ");
        String customerName = scanner.nextLine();

        System.out.print("Enter account type (1 for Savings, 2 for Current): ");
        int accountChoice = scanner.nextInt();

        scanner.nextLine(); // Consume newline character after integer input

        System.out.print("Enter account number: ");
        String accountNumber = scanner.nextLine();
```

```
Account account = null;

if (accountChoice == 1) {
    System.out.print("Enter initial deposit for Savings account: ");
    double initialDeposit = scanner.nextDouble();
    account = new SavAcct(customerName, accountNumber, initialDeposit);
} else if (accountChoice == 2) {
    System.out.print("Enter initial deposit for Current account: ");
    double initialDeposit = scanner.nextDouble();
    account = new CurAcct(customerName, accountNumber, initialDeposit);
} else {
    System.out.println("Invalid choice! Exiting program.");
    return;
}

// Menu for operations
boolean running = true;
while (running) {
    System.out.println("\nBank Operations Menu:");
    System.out.println("1. Deposit");
    System.out.println("2. Withdraw");
    System.out.println("3. Display Balance");
    System.out.println("4. Compute Interest (Savings account only)");
    System.out.println("5. Check and apply minimum balance penalty (Current account only)");
    System.out.println("6. Exit");
    System.out.print("Enter your choice: ");
    int choice = scanner.nextInt();

    switch (choice) {
        case 1:
            System.out.print("Enter deposit amount: ");
            double depositAmount = scanner.nextDouble();
```

```
                account.deposit(depositAmount);
                break;
            case 2:
                System.out.print("Enter withdrawal amount: ");
                double withdrawAmount = scanner.nextDouble();
                account.withdraw(withdrawAmount);
                break;
            case 3:
                account.displayBalance();
                break;
            case 4:
                if (account instanceof SavAcct) {
                    ((SavAcct) account).computeInterest();
                } else {
                    System.out.println("Interest can only be computed for Savings account.");
                }
                break;
            case 5:
                if (account instanceof CurAcct) {
                    ((CurAcct) account).checkMinimumBalance();
                } else {
                    System.out.println("Minimum balance check can only be applied to Current account.");
                }
                break;
            case 6:
                System.out.println("Exiting program.");
                running = false;
                break;
            default:
                System.out.println("Invalid choice! Please select a valid option.");
        }
    }
```

```
        scanner.close();
    }
}
```





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.

Create a package CIE which has 2 classes - Student & Internals. The class Student has members like usn, name, sem. The class internals has an array that stores the internal marks scored in 5 courses of the current sem 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 5 courses of the current sem of the student. Import the 2 packages in a file that declares the final marks of n students in all 5 courses.

Date: 14/11/2024.

```
package CIE;
import java.util.Scanner;
class student
{
  protected string usn;
  protected string name;
  protected int sem;
  public student (string usn, string name, int sem)
  {
    this.usn = usn;
    this.name = name;
    this.sem = sem;
  }
  void displaydetails()
  {
    S.O.P("USN: "+usn);
    S.O.P("Name: "+name);
    S.O.P ("Semester: "+sem);
  }
}
```

```
class Internals
{
  protected int[] internalmarks;
  Internals()
  {
    this.internalmarks = internalmarks, new int[5];
  }
  void inputmarks()
  {
    Scanner sc = new Scanner(System.in);
    S.O.p("Enter internal marks for 5 courses");
    for(int i=0; i<5; i++)
    {
      internalmarks[i] = sc.nextInt();
    }
  }
  void displaymarks()
  {
    S.O.p("Internal Marks:");
    for(i=0; i<5 ; i++)
    {
      S.O.p("Course" + (i+1) + ":" + internalmarks[i]);
    }
  }
}
package SEE;
import CIE.Student;
import CIE.Internals;
import java.util.Scanner;
```

```
public class External extends Student
{
  private int[] externalmarks;
  public External (String usn, String name, int sem)
  {
    super(usn, name, sem);
    this.externalmarks= new int[5];
  }
  public void inputexternalmarks ()
  {
    Scanner sc = new Scanner(System.in);
    S.O.P ("Enter element marks for 5 courses");
    for (int i=0; i<5; i++)
    {
      externalmarks[i] = sc.nextInt();
    }
  }
  public void display extmarks ()
  {
    S.O.P ("External marks : ");
    for (int i=0; i<5; i++)
    {
      S.O.P ("Course" + (i+1) +" : "+ externalmarks[i]);
    }
  }
  public void displayfinalmarks ()
  {
    displaydetails ();
    Internals internals = new Internals ();
    Internals.inputmarks ();
    internals.displaymarks ();
    displayextmarks ();
```

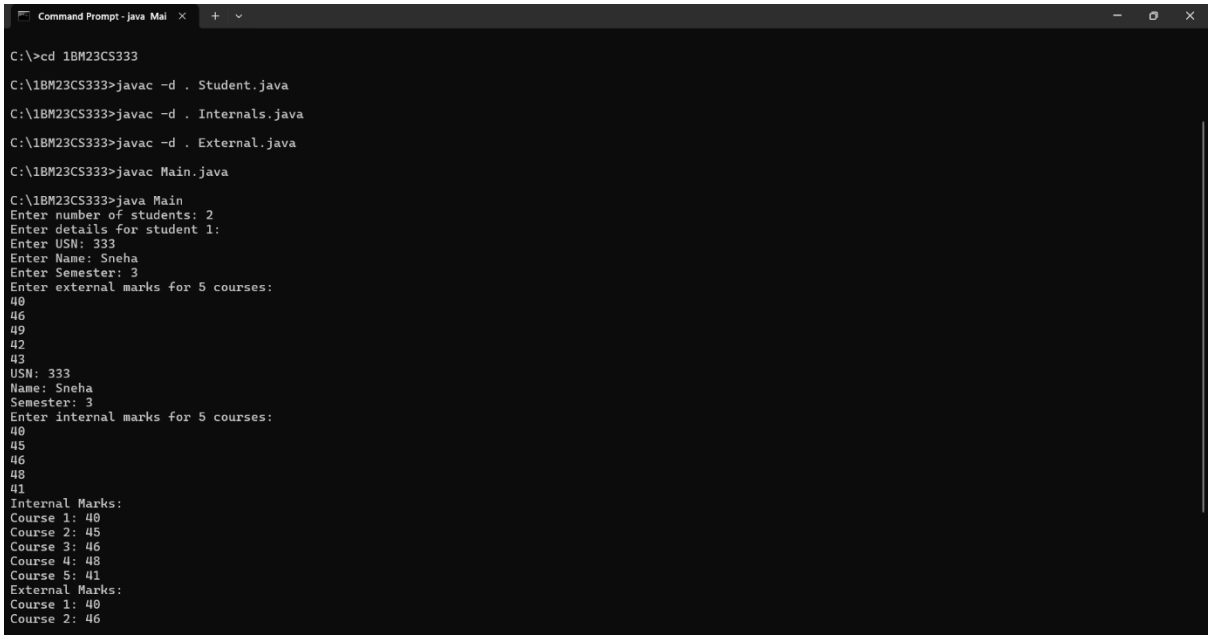
```
        S.O.p("\n Final Marks:");
        for(int i=0; i<5; i++)
        {
            int finalmark = internals.internalmarks[i] + externalmarks[i];
            S.O.p("Course" + (i+1) + ":" + finalmark);
        }
    }
}
import SEE.External;
import java.util.Scanner;
public class Main
{
    public static void main (String[] args)
    {
        Scanner sc = new Scanner (System.in);
        S.O.p("Enter no of students:");
        int n = sc.nextInt();
        for(int i=0; i<n; i++)
        {
            sc.nextLine();
            S.O.p("Enter details for student" + (i+1) + ":");
            S.O.p("Enter USN:");
            String usn = sc.nextLine();
            S.O.p("Enter Name:");
            String name = sc.nextLine();
            S.O.p("Enter semester:");
            int sem = sc.nextInt();
            External student = new External(usn, name, sem);
            student.inputmarks();
            student.displayfinalmarks(); } } }
```

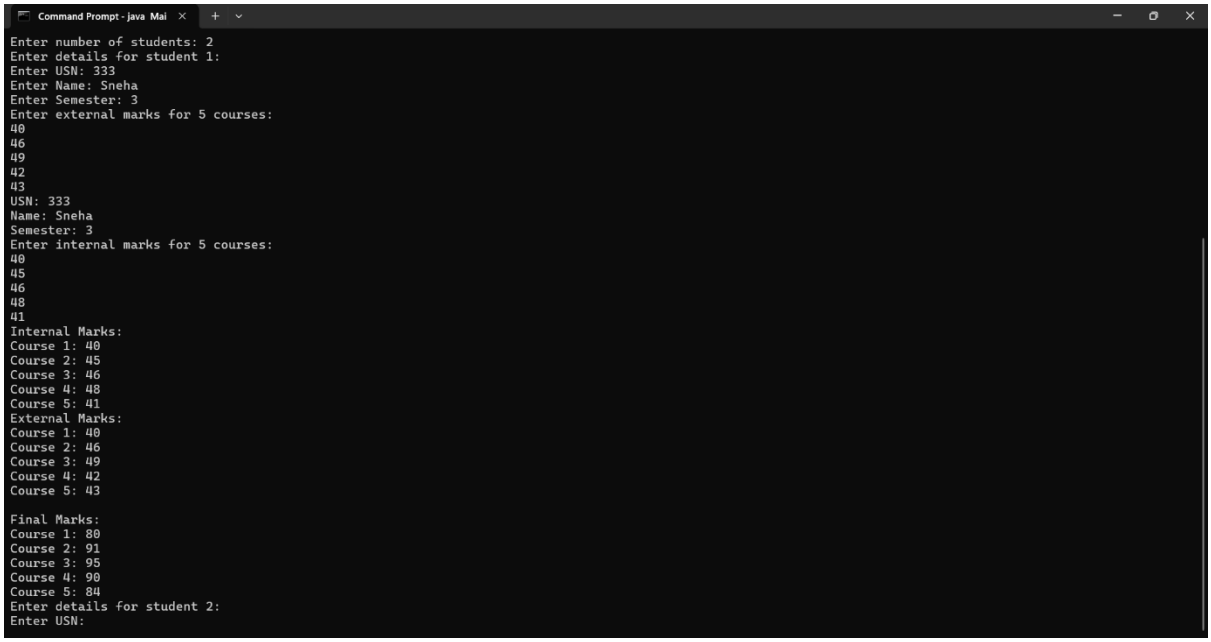
```
// Package CIE
package CIE;

import java.util.Scanner;

public class Student {
    protected String usn;
    protected String name;
    protected int sem;

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

    public void displayPersonalDetails() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Semester: " + sem);
    }
}

package CIE;.

import java.util.Scanner;

public class Internals {
    protected int[] internalMarks;

    public Internals() {
        this.internalMarks = new int[5]; // 5 courses
```

```
    }

    public void inputInternalMarks() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter internal marks for 5 courses:");
        for (int i = 0; i < 5; i++) {
            internalMarks[i] = sc.nextInt();
        }
    }

    public void displayInternalMarks() {
        System.out.println("Internal Marks:");
        for (int i = 0; i < 5; i++) {
            System.out.println("Course " + (i+1) + ": " + internalMarks[i]);
        }
    }
}

// Package SEE
package SEE;

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

public class External extends Student {
    private int[] externalMarks;

    public External(String usn, String name, int sem) {
        super(usn, name, sem);
        this.externalMarks = new int[5]; // 5 courses
    }
```

```
    public void inputExternalMarks() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter external marks for 5 courses:");
        for (int i = 0; i < 5; i++) {
            externalMarks[i] = sc.nextInt();
        }
    }

    public void displayExternalMarks() {
        System.out.println("External Marks:");
        for (int i = 0; i < 5; i++) {
            System.out.println("Course " + (i+1) + ": " + externalMarks[i]);
        }
    }

    public void displayFinalMarks() {
        displayPersonalDetails(); // Display personal details from the Student class
        Internals internals = new Internals();
        internals.inputInternalMarks();
        internals.displayInternalMarks();
        displayExternalMarks();

        System.out.println("\nFinal Marks:");
        for (int i = 0; i < 5; i++) {
            int finalMark = internals.internalMarks[i] + externalMarks[i];
            System.out.println("Course " + (i+1) + ": " + finalMark);
        }
    }
}

// Main program
```

```
import SEE.External;
import java.util.Scanner;

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

        for (int i = 0; i < n; i++) {
            sc.nextLine(); // Clear buffer
            System.out.println("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();

            External student = new External(usn, name, sem);
            student.inputExternalMarks();
            student.displayFinalMarks();
        }

        sc.close();
    }
}
```





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.

Lab-7 21/11/2024

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

Code:

```
import java.util.Scanner;
class WrongAgeException extends Exception
{
  public WrongAgeException(String message)
  {
    super(message);
  }
}
class Father
{
  protected int age;
  public Father(int age) throws WrongAgeException
  {
    if (age<0)
    {
      throw new WrongAgeException("Father's age cannot be negative");
    }
    this.age = age;
  }
}
```

```
class Son extends Father
{
  private 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;
  }
}
public class ExceptionInheritanceDemo
{
  public static void main( String [] args)
  {
    Scanner sc = new Scanner (System.in);
    try
    {
      S.O.P ("Enter Father's age: ");
      int fatherAge = sc.nextInt();
      S.O.P ("Enter Son's age: ");
      int sonAge = sc.nextInt();
      Son son= new Son (fatherAge, sonAge);
```

```
        s.o.p ("Father's age :" + fatherAge);
        s.o.p ("Son's age: " + sonAge);
        }
      catch (Exception e)
      {
        s.o.p ("An unexpected error occured: "+ e.getMessage());
      }
      finally
      {  sc.close ();
      }
    }
}
```

Output:

Enter Father's age : 35
Enter Son's age : 36
Exception: Son's age cannot be greater than & equal to Father's age!

Enter Father's age : -90
Enter Son's age : 23
Exception: Father's age cannot be negative!

Enter Father's age: 95
Enter Son's age : 46
Father's Age : 95
Son's Age : 46

Enter Father's age: 56
Enter Son's age : -23
Exception: Son's age cannot be negative!

21/11/24

```
import java.util.Scanner;

// Custom Exception Class
class WrongAgeException extends Exception {
    public WrongAgeException(String message) {
        super(message);
    }
}

// Base Class: Father
class Father {
    protected int age;

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

// Derived Class: Son
class Son extends Father {
    private int sonAge;

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

```
        }
        this.sonAge = sonAge;
    }
}

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

        try {
            // Taking user input for Father's age
            System.out.print("Enter Father's age: ");
            int fatherAge = scanner.nextInt();

            // Taking user input for Son's age
            System.out.print("Enter Son's age: ");
            int sonAge = scanner.nextInt();

            // Creating Son object which validates both ages
            Son son = new Son(fatherAge, sonAge);
            System.out.println("Father's Age: " + fatherAge);
            System.out.println("Son's Age: " + sonAge);

        } catch (WrongAgeException e) {
            // Handling the custom exception
            System.out.println("Exception: " + e.getMessage());
        } catch (Exception e) {
            // Handling any other unforeseen exceptions
            System.out.println("An unexpected error occurred: " + e.getMessage());
        } finally {
            scanner.close();
        }
```

```
    }
}
```

```
C:\1BM23CS333>javac Java.java

C:\1BM23CS333>java ExceptionInheritanceDemo
Enter Father's age: 45
Enter Son's age: 23
Father's Age: 45
Son's Age: 23

C:\1BM23CS333>java ExceptionInheritanceDemo
Enter Father's age: 45
Enter Son's age: 46
Exception: Son's age cannot be greater than or equal to Father's age!

C:\1BM23CS333>java ExceptionInheritanceDemo
Enter Father's age: -46
Enter Son's age: 23
Exception: Father's age cannot be negative!

C:\1BM23CS333>java ExceptionInheritanceDemo
Enter Father's age: 45
Enter Son's age: -26
Exception: Son's age cannot be negative!

C:\1BM23CS333>
```

Program 8

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

Lab-8 28/11/2024

Write a program which creates 2 threads, one thread displaying "BMS College of Engineering" once every 10 seconds & another thread displaying "CSE" once every 2 seconds

```
class DisplayMessage extends Thread
{
  private String msg;
  private int interval;
  public DisplayMessage (String msg, int interval)
  { this.msg = msg;
    this.interval = interval;
  }
  public void run()
  {
    try
    { while(true)
      {s.o.p(msg);
       Thread.sleep(interval*1000)
      }
    }
    catch (InterruptedException e)
    {
      s.o.p("Thread interrupted:" + e.getMessage());
    }
  }
}
public class ThreadExample
{
  public static void main(String xx[])
  {
    DisplayMessage f1 = new DisplayMessage("BMSCE", 10);
```

```
        DisplayMessage t2 = new DisplayMessage("CSE", 2);
        t1.start();
        t2.start();
    }
}
```

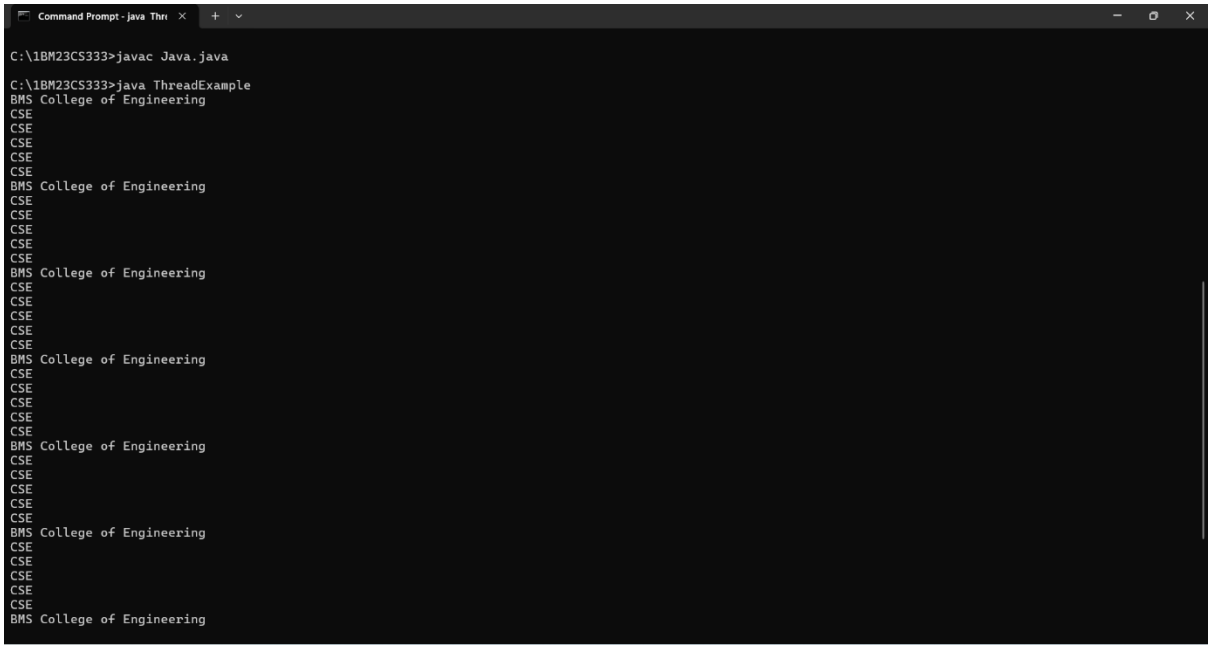
Output:

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

21/12/20

```
class DisplayMessage extends Thread {
    private String message;
    private int interval;

    public DisplayMessage(String message, int interval) {
        this.message = message;
        this.interval = interval;
    }

    @Override
    public void run() {
        try {
            while (true) {
                System.out.println(message);
                Thread.sleep(interval * 1000); // Convert seconds to milliseconds
            }
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted: " + e.getMessage());
        }
    }
}

public class ThreadExample {
    public static void main(String[] args) {
        // Create two threads
        DisplayMessage thread1 = new DisplayMessage("BMS College of Engineering", 10);
        DisplayMessage thread2 = new DisplayMessage("CSE", 2);

        // Start the threads
        thread1.start();
        thread2.start();
    }
```

}



Program 9

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

Lab-9

Write a program that creates a user interface to perform divisions. The user enters 2 nos 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 & Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an ArithmeticException. Display the exception in a message dialog box.

```
import java.awt.*;
import java.awt.event.*;
public class DivisionMain1 extends Frame implements ActionListener
{
  Textfield num1, num2;
  Button dResult;
  Label1
  outResult;
  String out = " ";
  double resultNum;
  int flag = 0;
  public DivisionMain1()
  {
    setLayout(new FlowLayout());
    dResult = new Button("RESULT");
    Label number1 = new Label("Number 1:", Label.RIGHT);
    Label number2 = new Label("Number 2:", Label.RIGHT);
    num1 = new Textfield(5);
    num2 = new Textfield(5);
    outResult = new Label("Result:", Label.RIGHT);
    add(number1);
    add(num1);
```

```
add(number2);
add(num2);
add(dResult);
add(outResult);
num1.addActionListener(this);
num2.addActionListener(this);
dResult.addActionListener(this);
addWindowListener(new WindowAdapter()
   { public void windowClosing(WindowEvent we)
     {
       System.exit(0);
     }
   });
}
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="NumberFormatException!"+e1.repaint();
}
catch (ArithmeticException e2)
{
   flag=1;
   out="Divide by 0 Exception!"+e2.repaint();
}
}
public void paint (Graphics g)
{
   if (flag==0)
   g.drawString(out, outResult.getX()+outResult.getWidth(), out
                Result.getY()+outResult.getHeight()-8);
   else
     g.drawString(out, 100, 200);
     flag=0;
 }
 public static void main (String a[])
 {
    DivisionMain1 dm = new DivisionMain1();
    dm.setSize (new Dimension (800, 400));
    dm.setTitle ("Division of Integers");
    dm.setVisible (true);
 }
}
```

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


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


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


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


            add(num
            ber1);
            add(num
            1);
            add(num
            ber2);
            add(num
            2);
            add(dRe
```

```
sult);
add(out
Result);


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(result
                        Num); repaint();


                }
        }
```

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

        }
        public void paint(Graphics g)
        {
            if(flag==0)
            g.drawString(out,outResult.getX()+outResult.getWidth(),outResult.getY()+
            outResult. getHeight()-8);
            else
            g.drawString(out,
            100,200); flag=0;
        }
        public static void main(String[] args)
        {
            DivisionMain1 dm=new
            DivisionMain1(); dm.setSize(new
            Dimension(800,400));
            dm.setTitle("DivisionOfIntegers");
            dm.setVisible(true);
        }
}
```

Program 10

Demonstrate Interprocess communication and deadlock

Lab-10

Demonstrate Interprocess communicatn & deadlock

```
class Q
{
  int n;
  boolean valueSet=false;
  synchronised int get(){
      while (!valueSet)
      try
      {
        S.O.p("\n Consumer waiting \n");
        wait();
      }
      catch (InterruptedException e)
      {
        s.o.p("Interrupted Exception caught");
      }
      s.o.p("Got:"+n);
      valueSet = false;
      s.o.p ("\n Intimate Producer \n");
      notify();
      return n;
  }
synchronised void put (int n)
{
   while(valueSet)
   try
   {
     s.o.p ("\n Producer waiting \n");
     wait();
   }
   catch(InterruptedException e)
  {
    s.o.p ("Interrupted Exception caught");
  }
```

```
    this.n = n;
    valueSet = true;
    s.o.p ("Put:" +n);
    s.o.p ("\n Intimate Consumer \n");
    notify();
    }
}
class Producer implements Runnable {
 Q q;
 Producer (Q q) {
      this.q = q;
      new Thread (this, "Producer").start();
 }
public void run() {
 int i=0;
 while (i<15) {
    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();
 s.o.p ("consumed:" + r);
```

```
        i++;
    }
  }
}
class PCFixed {
  public static void main(String arg[]){
      Q q = new Q();
      new Producer(q);
      new Consumer(q);
      S.O.P("Press Control-C to stop");
    }
}
```

Output:

```
D:\NotePad++\Java>javac DivisionMain1.java
D:\NotePad++\Java> java DivisionMain1
```

Demonstration of deadlock

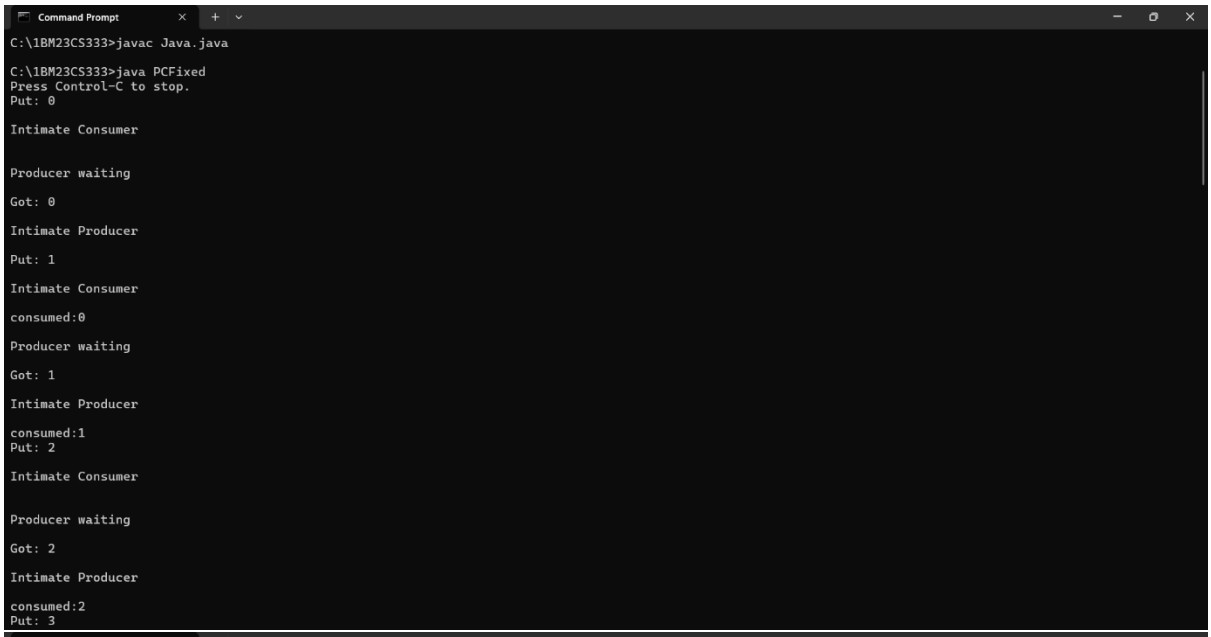
```
class A
{
  synchronized void foo(B b)
  {
    String name = Thread.currentThread().getName();
    S.o.p(name+" entered A.foo");
    try {
      Thread.sleep(1000); }
    catch(Exception e){ S.o.p("A Interrupted"); }
    S.o.p(name+"trying to call B.last()"); b.last(); }

  synchronized void last(){ S.o.p("Inside A.last"); }
}

class B
{
  synchronized void bar(A a){
    String name = Thread.currentThread().getName();
    S.o.p(name+" entered B.bar");
    try{ Thread.sleep(1000); }
    catch(Exception e){ S.o.p("B Interrupted"); }
    S.o.p(name+" trying to call A.last()"); a.last(); }

  synchronized void last(){ S.o.p("Inside A.last"); }
}
class Deadlock implements Runnable{
  A a = new A();
  B b = new B();
  Deadlock(){
    Thread.currentThread().setName("MainThread");
    Thread t = new Thread(this, "RacingThread");
```

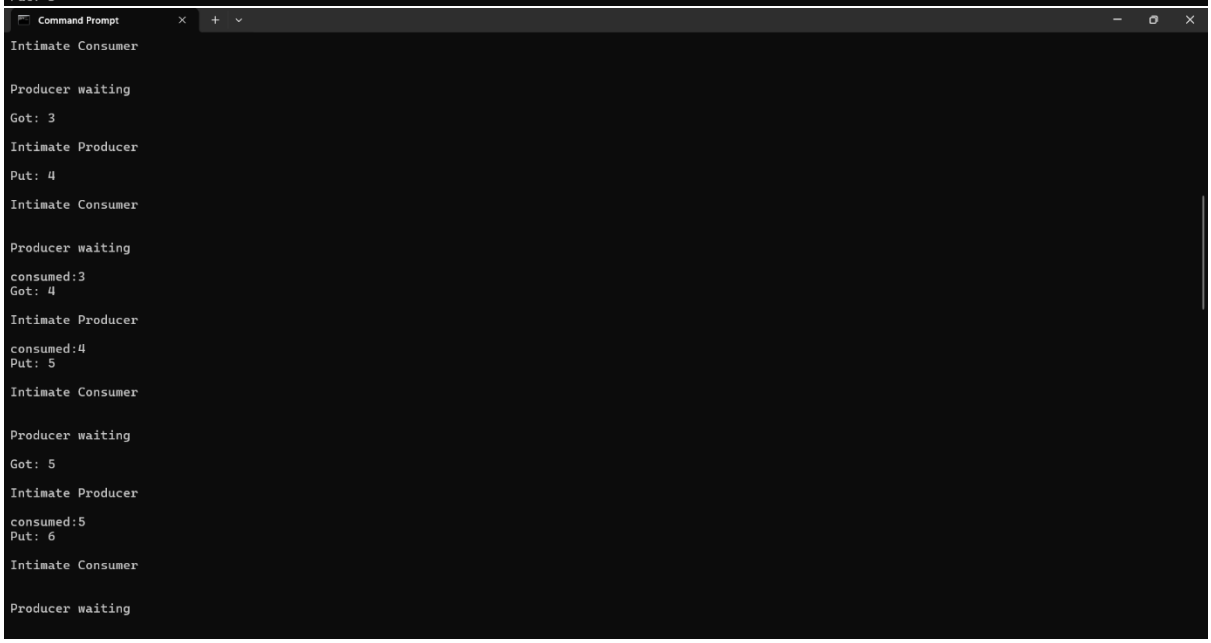
```
        t.start();
        a.foo(b);
        s.o.p ("Back in main thread");
    }
public void run() { b.bar(a);
        s.o.p ("Back in other thread");
    }
public static void main (String a[])
    { new Deadlock(); }
}
```

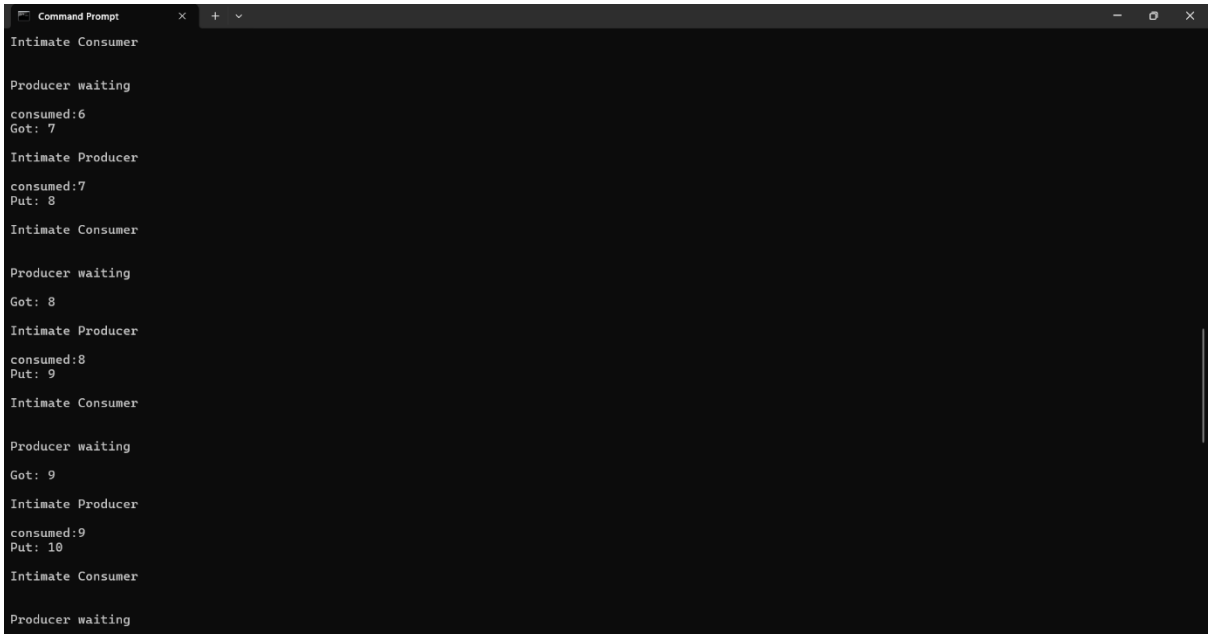
```
class Q {
int n;
boolean valueSet = false;

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("\nIntimate 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("\nIntimate Consumer\n");
notify();
}
```

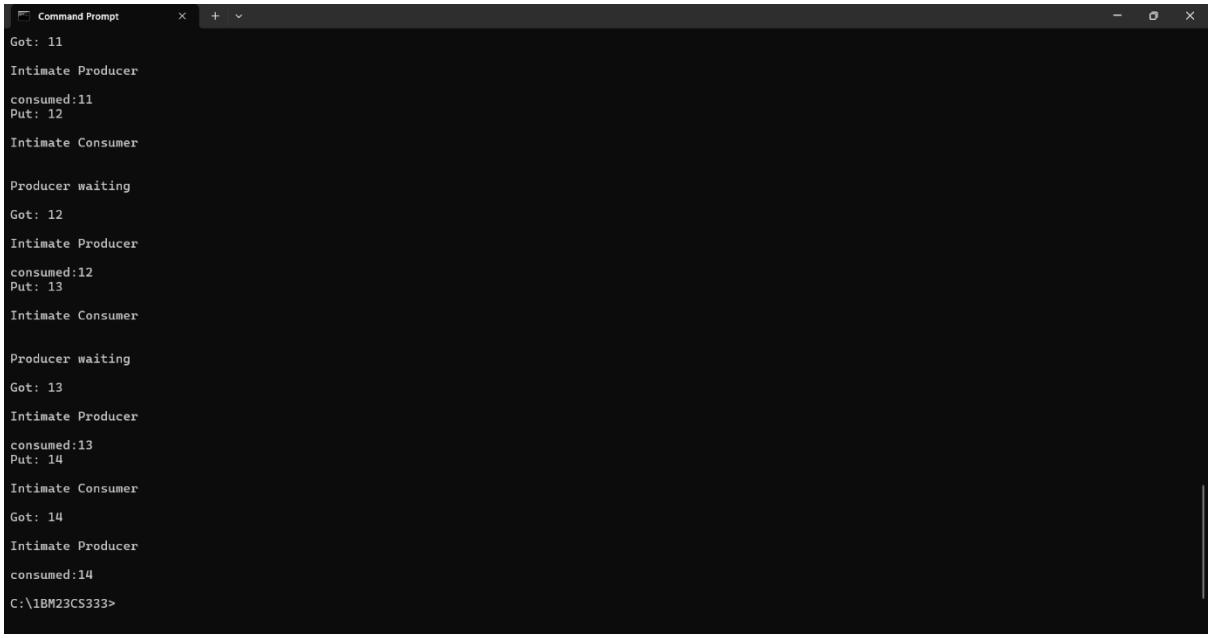
```
}

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

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

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.");
}
}
```






Command Prompt
Intimate Consumer
Producer waiting
consumed:6
Got: 7
Intimate Producer
consumed:7
Put: 8
Intimate Consumer
Producer waiting
Got: 8
Intimate Producer
consumed:8
Put: 9
Intimate Consumer
Producer waiting
Got: 9
Intimate Producer
consumed:9
Put: 10
Intimate Consumer
Producer waiting


Command Prompt
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:\1BM23CS333>