VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SAMIR CHAUDHARY(1BM23CS294)

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 **SAMIR CHAUDHARY(1BM23CS294),** 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

		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.	
6	14/11/2024	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.	35-42
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	43-48

		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.	
8	13/12/2024	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.	49-52
9	20/12/2024	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.	53-55
10	20/12/2024	Demonstrate Inter process Communication and deadlock	56-60

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.

[OOJ] [26/09]

LAB PROGRAM – ONE

(1) Develop a java program that prints all real solutions to the quadratic equation $ax^2+bx+c=$ 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 solutions.

=>

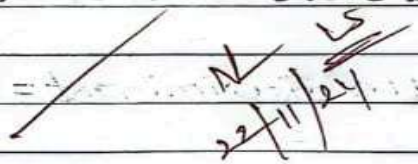
```
import java.util.Scanner;
Class QuadraticEquation {
    public static void main (String[] args) {
        Scanner scanner = new Scanner(System.in);

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

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

        if (discriminant > 0) {
            double root1 = (-b + Math.sqrt(discriminant
                    ))/(2*a);
            double root2 = (-b - Math.sqrt(discriminant)
                    )/(2*a);
            System.out.println("The real solutions
                are: " + root1 + "and " + root2);
        }
        else if (discriminant == 0) {
            double root = -b / (2*a);
            System.out.println("The real solution
                is: " + root);
        }
```

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

Output:-

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

case 2:- Enter coefficient a : 3
Enter coefficient b : 2
Enter coefficient c : 1
These are no real solutions.



Code:

```
import java.util.Scanner;

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

        // Read input values for a, b, and c
        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 (b^2 - 4ac)
        double discriminant = b * b - 4 * a * c;

        // Check the value of the discriminant
        if (discriminant > 0) {
            // Two real solutions
            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            System.out.println("The real solutions are: " + root1 + " and " +
root2);
        }
        else if (discriminant == 0) {
            // One real solution
            double root = -b / (2 * a);
            System.out.println("The real solution is: " + root);
        }
        else {
            // No real solutions
            System.out.println("There are no real solutions.");
        }


    }
}
```

Output:

```
C:\java\bin\1BM23CS294>java QuadraticEquation
Enter coefficient a: 1
Enter coefficient b: -3
Enter coefficient c: 2
The real solutions are: 2.0 and 1.0

C:\java\bin\1BM23CS294>java QuadraticEquation
Enter coefficient a: 3
Enter coefficient b: 2
Enter coefficient c: 1
There are no real solutions.
```

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.

[OOJ] [3/10]

LAB PROGRAM – TWO

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

=>

```
import java.util.Scanner;
class student {
  int n;
  int marks[];
  int credits[];
  String usn, name;
  Scanner scanner = new Scanner(System.in);

  public student(){
    System.out.println("Enter the number of
                        subjects: ");
    n = Scanner.nextInt();
    marks = new int[n];
    credits = new int[n];
  }

  void getDetails(){
    System.out.println("Enter the USN: ");
    usn = scanner.nextline();
    System.out.println("Enter the Name: ");
    name = Scanner.nextline();
    System.out.println("Enter the credits
        and marks of "+"n"+"subjects: ");
    for(int i=0; i<n; i++){
      credits[i] = scanner.nextInt();
      marks[i] = Scaner.nextInt();
    }
}
```

```
void displayDetails(){
    System.out.println("Name: "+name+
                       "USN: "+usn);
    System.out.println("The marks and
      credits of "+n+" subjects: ");
    for(int i=0; i<n; i++){
      System.out.println(marks[i]+": "
                         +credits[i]);
    }
    System.out.println("SGPA: "+calculate-
                       SGPA());
}

public double calculateSGPA(){
    double totalCredits=0;
    double totalpoints=0;
    for(int i=0; i<n; i++){
      double points=0;
    if(marks[i]>=90) points=10;
    else if(marks[i]>=80) points=9;
    else if(marks[i]>=70) points=8;
    else if(marks[i]>=60) points=7;
    else if(marks[i]>=50) points=6;
    else if(marks[i]>=40) points=5;
      totalCredits+=credits[i];
      totalPoints+=points*credits[i];
    }
    return totalPoints/totalCredits;
    }
}
```

```
Class Main {
    public static void main (String[] args){
    Student S1 = new Student ();
    S1.getDetails ();
    S1.displayDetails ();
    }
}
```

* Output:-

```
Enter the number of subjects:
3
Enter the USN:
1BM23CS294
Enter the Name:
SAMIR CHAUDHARY
Enter the credits and marks of n subjects:
4   95
3   93
2   91
Name: SAMIR CHAUDHARY USN: 1BM23CS294
The marks and credits of 3 subjects:
95 : 4
93 : 3
91 : 2
SGPA : 10.0
```

Code:

```
import java.util.Scanner;
class Student {
    int n;
    int marks[];
    int credits[];
    String usn,name;
    Scanner sc=new Scanner(System.in);

    public Student () {
        System.out.println("Enter the number of subjects: ");
        n=sc.nextInt();
        marks=new int[n];
        credits=new int[n];
    }
    void getDetails(){
        System.out.println("Enter the USN:");
        usn=sc.nextLine();


        System.out.println("Enter the Name:");
        name=sc.nextLine();
        System.out.println("Enter the credits and marks of"+" n " + "subjects:
");
        for(int i=0;i<n;i++){
            credits[i]=sc.nextInt();
            marks[i]=sc.nextInt();
        }
    }

    void displayDetails(){
        System.out.println("Name: "+name+" USN: "+usn);


        System.out.println("The marks and credits of "+n+" subjects: ");
        for(int i=0;i<n;i++){
            System.out.println(marks[i]+":"+credits[i]);
        }
        System.out.println("SGPA: "+calculateSGPA());
    }
    public double calculateSGPA() {
        double totalCredits = 0;
        double totalPoints = 0;
```

```
        for (int i = 0; i < n; i++) {
            double points = 0;
            if (marks[i] >= 90) points = 10;
            else if (marks[i] >= 80) points = 9;
            else if (marks[i] >= 70) points = 8;
            else if (marks[i] >= 60) points = 7;
            else if (marks[i] >= 50) points = 6;
            else if (marks[i] >= 40) points = 5;
            totalCredits += credits[i];
            totalPoints += points * credits[i];
            }

        return totalPoints/totalCredits;
        }
}

class Main {
    public static void main(String[] args){
    Student s1=new Student ();
    s1.getDetails();
    s1.displayDetails();

    }

}
```

Output:

```
C:\java\bin\1BM23CS294>java Main
Enter the number of subjects:
3
Enter the USN:
1BM23CS294
Enter the Name:
SAMIR CHAUDHARY
Enter the credits and marks of n subjects:
4 95
3 93
2 91
Name: SAMIR CHAUDHARY USN: 1BM23CS294
The marks and credits of 3 subjects:
95:4
93:3
91:2
SGPA: 10.0
```

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.

[OOJ] [19/10]

LAB PROGRAM - THREE

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

=>

```
import java.util.Scanner;
class Book {

    String name;
    String author;
    double price;
    int num_pages;

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

```
String getName(){
return name;
}
String getAuthor(){
return author; }
double getPrice(){
return price;
}
int getNum_pages(){
return num_pages;
}
public string to string(){
return "Book_Details\n"+
"Name:"+name+"\nAuthor:"+author+
"\nPrice:"+price+"\npages:"+numpages;
}
}
Class BookDetails {
public static void main(String[]args){
  Book[] books;
  Scanner scanner = new Scanner(System.in);
  System.out.println("Enter the number
                      of Book");
  int n = sc.nextInt();
  books = new Book[n];
  for(int i=0; i<n; i++){
  System.out.println("Enter the details
                      of book"+(i+1)+":");
  System.out.println("Enter the name of
                      Book:");
  String name = scanner.next();
  System.out.println("Enter the name of
                      Author:");
```

```
String author=scanner.next();
System.out.println("Enter the
        price of Book : ");
double price = sc.nextDouble();
System.out.println("Enter the
        Number of Pages : ");
int num_pages=scanner.nextInt();

book[i]=new Book(name, author,
                price, num_pages);
}
for(int i=0; i<n; i++){
System.out.println("The Details of
        book "+(i+1)+" : ");
System.out.println(books[i].to
                String());
}
}
}
```

* Output:

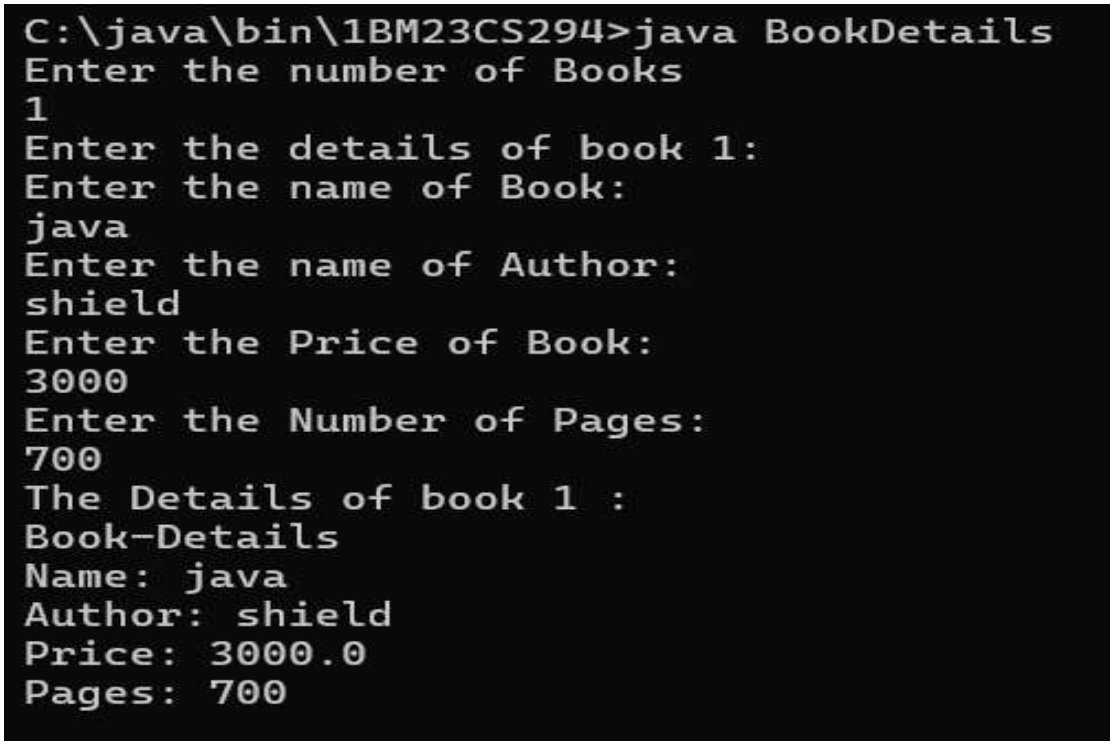
```
Enter the number of Books
1
Enter the details of book 1:
Enter the name of Book:
java
Enter the name of Author:
Shield
Enter the price of book:
3000
Enter the Number of pages:
700
The details of book 1:
Book Details
Name: java, Author: Shield, Price: 3000,
                pages: 700
```

Code:

```
import java.util.Scanner;
class Book{

String name;
String author;
double price;
int num_pages;

Book(String name, String author, double price, int num_pages){
this.name=name;
this.author=author;
this.price=price;
this.num_pages=num_pages;
}
void setDetails(String name, String author, double price, int num_pages){
this.name=name;
this.author=author;
this.price=price;
this.num_pages=num_pages;
}
String getName(){
return name;
}
String getAuthor(){
return author;
}
double getPrice(){
return price;
}
int getNum_pages(){
return num_pages;
}

public String toString(){

return "Book-Details\n"+
    "Name: "+name+"\nAuthor: "+author+"\nPrice: "+price+"\nPages: "+num_pages;
}
}

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

```
Book[] books;
Scanner sc=new Scanner(System.in);
System.out.println("Enter the number of Books");
int n=sc.nextInt();
books=new Book[n];

for(int i=0;i<n;i++){
System.out.println("Enter the details of book "+(i+1)+": ");

System.out.println("Enter the name of Book: ");
String name=sc.next();

System.out.println("Enter the name of Author: ");
String author=sc.next();

System.out.println("Enter the Price of Book: ");
double price=sc.nextDouble();

System.out.println("Enter the Number of Pages: ");
int num_pages=sc.nextInt();

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

}
for(int i=0;i<n;i++){
System.out.println("The Details of book "+(i+1)+" : ");

System.out.println(books[i].toString());

}

}
}
```

Output:


C:\java\bin\1BM23CS294>java BookDetails
Enter the number of Books
1
Enter the details of book 1:
Enter the name of Book:
java
Enter the name of Author:
shield
Enter the Price of Book:
3000
Enter the Number of Pages:
700
The Details of book 1 :
Book-Details
Name: java
Author: shield
Price: 3000.0
Pages: 700

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.

[24110]

LAB PROGRAM-4

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

=>

```
import java.util.Scanner;

abstract class Shape{
  int dimension1, dimension2;

  Shape(int dimension1, int dimension2)
{
    this.dimension1 = dimension1;
    this.dimension2 = dimension2;
  }
  abstract void printArea();
}
Class Rectangle extends Shape{
   Rectangle(int length, int breadth){
     super(length, breadth);
   }
   void printArea(){
     System.out.println("Area of Rectangle: "+
       (dimension1 * dimension2));
   }
}
Class Triangle extends Shape{
     Triangle(int base, int height){
       super(base, height);
     }
```

```
    Void printArea() {
        System.out.println("Area of
Triangle: " + (0.5 * dimension1 * dimension2));
    }
}
Class Circle extends Shape {
    Circle(int radius) {
        super(radius, 0);
    }
    VoidprintArea() {
        System.out.println("Area of Circle: " +
(Math.PI * dimension1 * dimension1));
    }
}
public class ~~Main~~ abstract {
    public static void main(String[] args)
{
    Shape[] shapes = { new Rectangle(5,3),
new Triangle(4,6), new Circle(7)};
        for(Shape shape : shapes)
shape.printArea();
    }
}
```

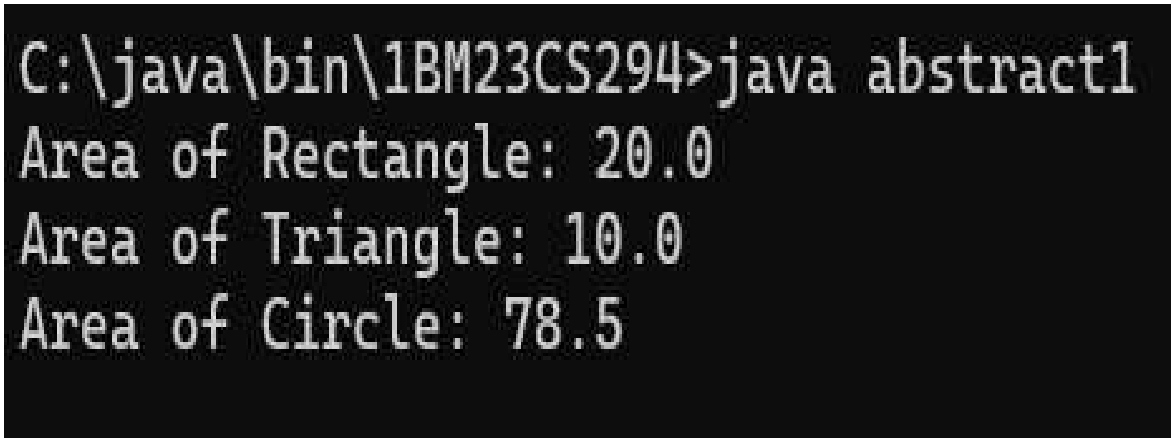
* Output:-

```
Area of Rectangle: [redacted] 20.0
Area of Triangle: [redacted] 10.0
Area of circle: [redacted] 78.5
```

Code:

```
import java.util.Scanner;
abstract class Shape{
    double a; double b;
    Shape(double a,double b){
        this.a=a;
        this.b=b;
    }
    abstract void printArea();
}
class Rectangle extends Shape{
    Rectangle(double a,double b){
        super(a,b);
    }
    void printArea(){
        System.out.println("Area of Rectangle: "+a*b);
    }
}
class Triangle extends Shape{
    Triangle(double a,double b){
        super(a,b);
    }
    void printArea(){
        System.out.println("Area of Triangle: "+0.5*a*b);
    }
}
class Circle extends Shape{
    Circle(double a){
        super(a,0.0);
    }
    void printArea(){
        System.out.println("Area of Circle: "+3.14*a*a);}
}
class abstract1{
    public static void main(String[] args){
        Rectangle r1=new Rectangle(5.0,4.0);
        Triangle t1=new Triangle(5.0,4.0);
        Circle c1=new Circle(5.0);
        r1.printArea();
        t1.printArea();
        c1.printArea();
    }
}
```

Output:


C:\java\bin\1BM23CS294>java abstract1
Area of Rectangle: 20.0
Area of Triangle: 10.0
Area of Circle: 78.5

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:

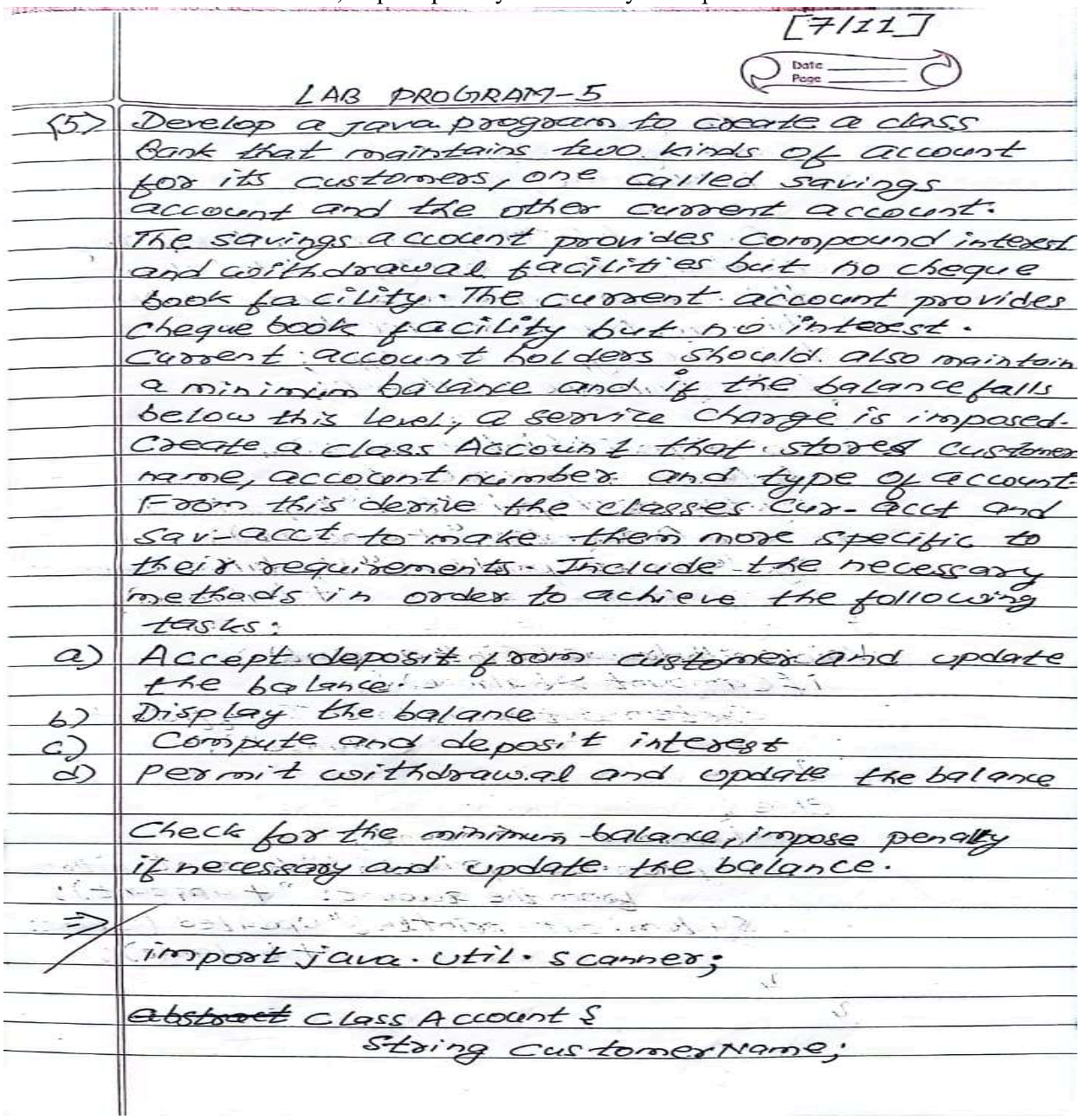
a) Accept deposit from customer and update the balance.

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.



[7/11]

Date
Page

LAB PROGRAM-5

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

a) Accept deposit from customer and update the balance.

b) Display the balance

c) Compute and deposit interest

d) permit withdrawal and update the balance

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

=>

```
import java.util.Scanner;

abstract class Account {
        String customerName;
```

```
    String accNumber;
    String accType;
    int balance;

    Account(String customerName, String
accNumber, String accType, int balance){
    this.customerName = customerName;
    this.accNumber = accNumber;
    this.accType = accType;
    this.balance = balance;
  }
  void deposit(int amount){
    balance += amount;
    System.out.println("Deposited
        Amount: "+amount);
    System.out.println("Updated Balance:"
                    +balance);
  }
  void withdrawal(int amount){
    if(amount > balance){
      System.out.println("Insufficient
          Balance");
    }
    else{
      balance -= amount;
      System.out.println("Amount withdrawn
          from the account: "+amount);
      System.out.println("Updated Balance:"
                    +balance);
    }
}
```

```
void checkBookFacility() {
  if(accType.equals("Savings")){
    System.out.println("Sorry, saving account
      does not have cheque book facility");
  }
  else{
    System.out.println("CheckBook Available");
  }
}
}
class SavingAmount extends Account{
  int interestRate;
  SavingAccount(String customerName, String
    accType, int balance, int interestRate){
    super(customerName, accNumber, "Saving",
      balance);
    this.interestRate = interestRate;
  }
  void computeAndDepositInterest(){
    int interest = balance* interestrate/100;
    balance+ = interest;
    System.out.println("Interest Earned: "
      +interest);
    System.out.println("Updated Balance: "
      +balance);
  }
  void withdrawal(int amount){
    if(amount >balance){
      System.out.println("Insufficient
        Balance");
    }
    else{
```

```
        balance -= amount;
        System.out.println("Amount withdrawn
                from the account: "+amount);
        System.out.println("Updated Balance: "
                                +balance);
    }
  }
}
Class currentAccount extends Account{
    int minimumBalance;
    int serviceCharge;
    currentAccount(String customerName,
    String accNumber, String accType, int
    balance, int minimumBalance, int serviceCharge){
        super(customerName, accNumber, "current",
                balance);
        this.minimumBalance=minimumBalance;
        this.serviceCharge = serviceCharge;
    }
  void deposit(int amount){
        balance+=amount;
        System.out.println("Deposited Amount: "
                                +amount);
        System.out.println("Updated Balance: "
                                +balance);
        checkForMinimum();
    }
    void withdrawal(int amount){
        if(amount>balance){
            System.out.println("Insufficient
                                Balance");
    }
```

```
        else{
            balance -= amount;
            System.out.println("Amount withdrawn
                from the account: "+amount);
            System.out.println("Updated Balance:"
                                +balance);
            checkForMinimum();
        }
    }
    void checkForMinimum(){
        if(balance<minimumBalance){
            balance -= serviceCharge;
            System.out.println("Account Balance
                after imposing service charge: "+balance);
        }
    }
}
class Bank{
    public static void main(String[] args){

        SavingAccount s1 = new SavingAccount
            ("Milan","123","Saving",5000,13);
        System.out.println("For saving Account");
        s1.displayBalance();
        s1.computeAndDepositInterest();
        s1.deposit(1000);
        s1.withdraw(2000);
        s1.checkBookFacility();
        s1.displayBalance();

        CurrentAccount c1 = new CurrentAccount("Rahul",
            "456","Current",100,2500,100);
```

```
        System.out.println("For Current Account");
        C1.displayBalance();
        C1.deposit(1000);
        C1.withdrawal(500);
        C1.displayBalance();

    }
}
```

* Output:-

```
For saving Account
Account Holder: Milan
Account Balance : 5000
Interest Earned: 650
Updated Balance : 5650
Deposited Amount : 1000
Updated Balance : 6650
Amount withdrawn from the account : 2000
Updated Balance : 4650
CheckBook Available.
Account Holder: Milan
Account Balance: 4650

For Current Account
Account Holder: Rahul
Account Balance: 100
Deposited Amount: 1000
Updated Balance: 1100
Account Balance after imposing service charge: 1000
Amount withdrawn from the account: 500
Updated Balance: 500
Account Holder: Rahul
Account Balance : 500
```

20/11/25

Code:

```
import java.util.Scanner;
class Account {
    String customerName;
    String accNumber;
    String accType;
    int balance;

    Account(String customerName, String accNumber, String accType, int
balance) {
        this.customerName = customerName;
        this.accNumber = accNumber;
        this.accType = accType;
        this.balance = balance;
    }
    void deposit(int amount) {
        balance += amount;
        System.out.println("Deposited Amount: " + amount);
        System.out.println("Updated Balance: " + balance);
    }
    void withdrawal(int amount) {
        if (amount > balance) {
            System.out.println("Insufficient Balance");
        } else {
            balance -= amount;
            System.out.println("Amount withdrawn from the account: " +
amount);
            System.out.println("Updated Balance: " + balance);
        }
    }
    void displayBalance() {
        System.out.println("Account Holder: " + customerName);
        System.out.println("Account Balance: " + balance);
    }

    void checkBookFacility() {
        if (accType.equals("Saving")) {
            System.out.println("Sorry, Savings account does not have cheque
book facility.");
        } else {
            System.out.println("CheckBook Available.");
        }
    }
```

```
}
class SavingAccount extends Account {
    int interestRate;
    SavingAccount(String customerName, String accNumber, String accType, int
balance, int interestRate) {
        super(customerName, accNumber, "Saving", balance);
        this.interestRate = interestRate;
    }
    void computeAndDepositInterest() {
        if (balance > 0) {
            int interest = balance * interestRate / 100;
            balance += interest;
            System.out.println("Interest Earned: " + interest);
            System.out.println("Updated Balance: " + balance);
        } else {
            System.out.println("No interest added due to zero or negative
balance.");
        }
    }
    void withdrawal(int amount) {
        if (amount > balance) {
            System.out.println("Insufficient Balance");
        } else {
            balance -= amount;
            System.out.println("Amount withdrawn from the account: " +
amount);
            System.out.println("Updated Balance: " + balance);
        }
    }
}
class CurrentAccount extends Account {
    int minimumBalance;
    int serviceCharge;

    CurrentAccount(String customerName, String accNumber, String accType, int
balance, int minimumBalance, int serviceCharge) {
        super(customerName, accNumber, "Current", balance);
        this.minimumBalance = minimumBalance;
        this.serviceCharge = serviceCharge;
    }
    void deposit(int amount) {
        balance += amount;
        System.out.println("Deposited Amount: " + amount);
        System.out.println("Updated Balance: " + balance);
```

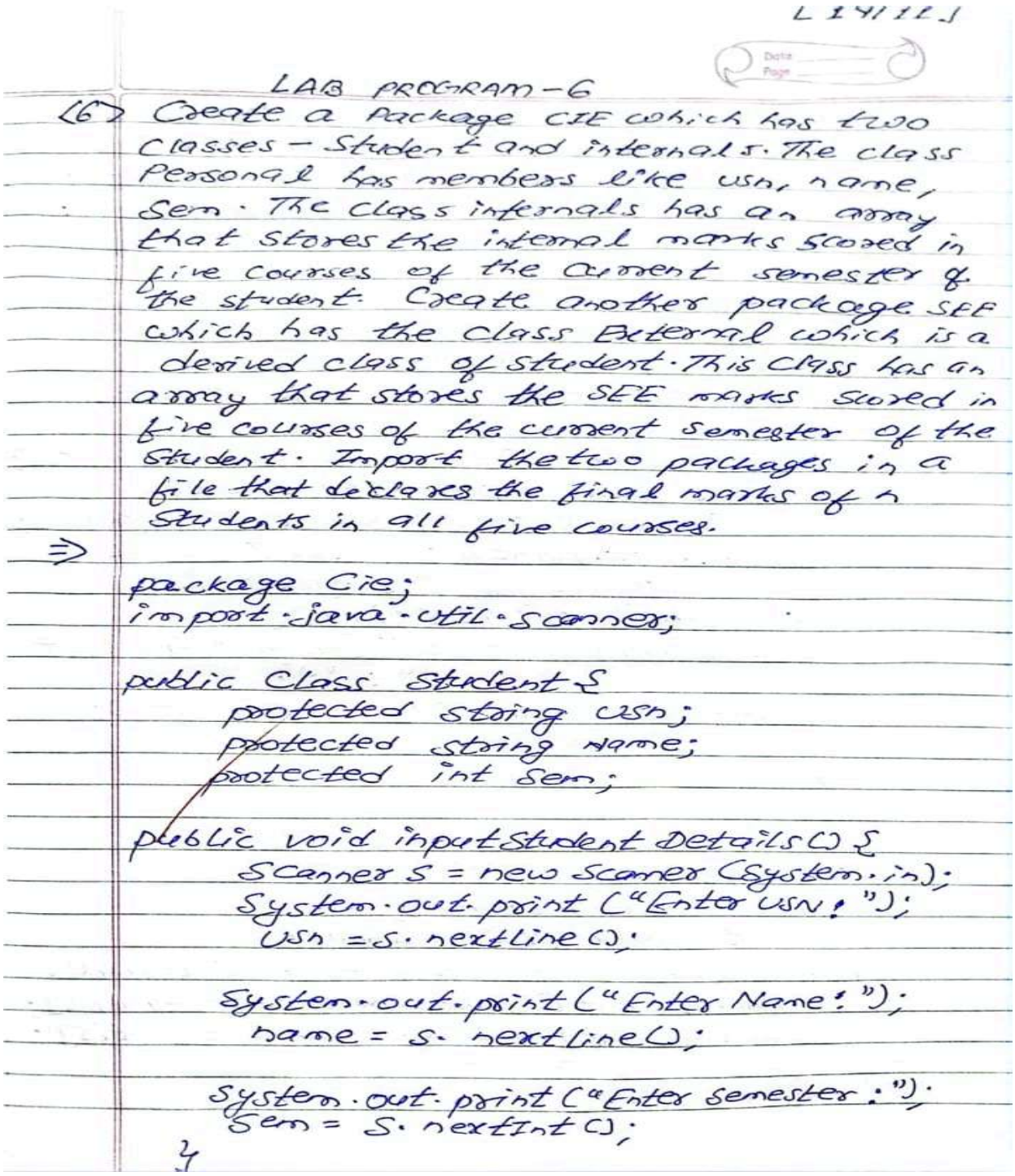
```
        checkForMinimum();
    }
    void withdrawal(int amount) {
        if (amount > balance) {
            System.out.println("Insufficient Balance");
        } else {
            balance -= amount;
            System.out.println("Amount withdrawn from the account: " +
amount);
            System.out.println("Updated Balance: " + balance);
            checkForMinimum();
        }
    }
    void checkForMinimum() {
        if (balance < minimumBalance) {
            balance -= serviceCharge;
            System.out.println("Account Balance after imposing service charge:
" + balance);
        }
    }
}
class Bank {
    public static void main(String[] args) {
        SavingAccount s1 = new SavingAccount("Milan", "123", "Saving", 5000,
13);
        System.out.println("For Saving Account:");
        s1.displayBalance();
        s1.computeAndDepositInterest();
        s1.deposit(1000);
        s1.withdrawal(2000);
        s1.checkBookFacility();
        s1.displayBalance();

        CurrentAccount c1 = new CurrentAccount("Rahul", "456", "Current", 100,
2500, 100);
        System.out.println("\nFor Current Account:");
        c1.displayBalance();
        c1.deposit(1000);
        c1.withdrawal(500);
        c1.displayBalance();
    }
}
```

Output:

```
C:\java\bin\1BM23CS294>java Bank
For Saving Account:
Account Holder: Milan
Account Balance: 5000
Interest Earned: 650
Updated Balance: 5650
Deposited Amount: 1000
Updated Balance: 6650
Amount withdrawn from the account: 2000
Updated Balance: 4650
Sorry, Savings account does not have cheque book facility.
Account Holder: Milan
Account Balance: 4650

For Current Account:
Account Holder: Rahul
Account Balance: 100
Deposited Amount: 1000
Updated Balance: 1100
Account Balance after imposing service charge: 1000
Amount withdrawn from the account: 500
Updated Balance: 500
Account Balance after imposing service charge: 400
Account Holder: Rahul
Account Balance: 400
```

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.



LAB PROGRAM-6

(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.

⇒

```
package Cie;
import java.util.Scanner;

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

public void inputStudentDetails() {
    Scanner s = new Scanner (System.in);
    System.out.print ("Enter USN: ");
     usn = s.nextLine();

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

    System.out.print ("Enter semester: ");
     sem = s.nextInt();
}
```

```
public void displayStudentDetails(){
    System.out.println("USN : "+USN);
    System.out.println("Name: "+Name);
    System.out.println("Semester: "+sem);
  }
}
```

```
Package cie;
import java.util.Scanner;
public class Internals extend Student{

  protected int[] internalMarks = new int[5];
  public void input CIEmarks(){
      Scanner s = new Scanner(System.in);
  System.out.println("Enter Internal
          marks for 5 subjects : ");
  For(int i=0; i<5; i++){
      System.out.print("Subject "+(i+1)+
                  " : ");
      internalMarks[i] = s.nextInt();
  }
 }
}
```

```
Package see;
import cie.Internals;
import java.util.Scanner;
public class External extends Internals{
    private int[] SEEmarks = new int[5];
    private int[] finalMarks = new int[5];
```

```
public void inputseemarks(){
  Scanner s = new Scanner(System.in);
  System.out.println("Enter SEE marks
                      for 5 subjects:");
    seemarks[i] = s.nextInt();
  }
}

public void displayfinalmarks(){
    for(int i=0; i<5; i++){
      finalmark[i] = internal marks[i] +
                     Final marks
  }
}

public void display final marks(){
    displaystudentDetails();
    System.out.println("Final marks for
                        5 students:");
    For(int i=0; i<5; i++).
    System.out.println("subject"+(i+1)+
                       ":"+ Final marks[i]
  }
}

import see.Externals;
import java.util.Scanner;
Class Main{
    public static void main(String[] args){
    Scanner s = new Sconner(System.in)
    System.out.print("Enter number of Student:")
     int n = s.nextInt();
```

```
Externals [] students = new Externals [n];
for (int i=0; i<n; i++) {
    System.out.println ("\n Enter
    details for student: " + (i+1) + ": ");

    students [i] = new Externals ();
    Students [i] . inputs student Details ();
    students [i] . inputs CIE marks ();
    students [i] . input SEE marks ();
    students [i] . calculate Final marks ();
}

System.out.println ("\n Final marks
                    of students: ");
for (int i=0; i<n; i++) {
    System.out.println ("\n student " +
    (i+1) + ": ");
    students [i] . display Final marks ();
}
}
```

Output:-

```
Enter number of students: 1
Enter details for student 1:
Enter USN: 1BM23CS294
Enter name: SAMIR CHAUDHARY
Enter sem: 3

Enter Internal marks for 5 subjects:
  Subject 1: 40
  Subject 2: 43
  Subject 3: 42
  Subject 4: 43
  Subject 5: 38

Enter SEE marks for 5 students:
  Subject 1: 43
  Subject 2: 42
  Subject 3: 38
  Subject 4: 48
  Subject 5: 38

Student 1:
USN: 1BM23CS294
Name: SAMIR CHAUDHARY
Sem: 3

Final marks for 5 students:
Subject 1: 83
Subject 2: 85
Subject 3: 80
Subject 4: 91
Subject 5: 76
```

28/11/24

Code:

```
package cie;
import java.util.Scanner;

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

    public void inputStudentDetails() {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter USN: ");
        usn = s.nextLine();
        System.out.print("Enter Name: ");
        name = s.nextLine();
        System.out.print("Enter Semester: ");
        sem = s.nextInt();
    }

    public void displayStudentDetails() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Semester: " + sem);
    }
}
package cie;
import java.util.Scanner;

public class Internals extends Student {
    protected int[] internalMarks = new int[5];

    public void inputCIEmarks() {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter Internal Marks for 5 subjects:");
        for (int i = 0; i < 5; i++) {
            System.out.print("Subject " + (i + 1) + ": ");
            internalMarks[i] = s.nextInt();
        }
    }
}
package see;
import cie.Internals;
import java.util.Scanner;
```

```java
public class Externals extends Internals {
    private int[] seeMarks = new int[5];
    private int[] finalMarks = new int[5];

    public void inputSEEmarks() {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter SEE Marks for 5 subjects:");
        for (int i = 0; i < 5; i++) {
            System.out.print("Subject " + (i + 1) + ": ");
            seeMarks[i] = s.nextInt();
        }
    }

    public void calculateFinalMarks() {
        for (int i = 0; i < 5; i++) {
            finalMarks[i] = internalMarks[i] + seeMarks[i];
        }
    }

    public void displayFinalMarks() {
        displayStudentDetails();
        System.out.println("Final Marks for 5 subjects:");
        for (int i = 0; i < 5; i++) {
            System.out.println("Subject " + (i + 1) + ": " + finalMarks[i]);
        }
    }
}
import see.Externals;
import java.util.Scanner;

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

        Externals[] students = new Externals[n];

        for (int i = 0; i < n; i++) {
            System.out.println("\nEnter details for student " + (i + 1) +
":");
            students[i] = new Externals();
            students[i].inputStudentDetails();
```

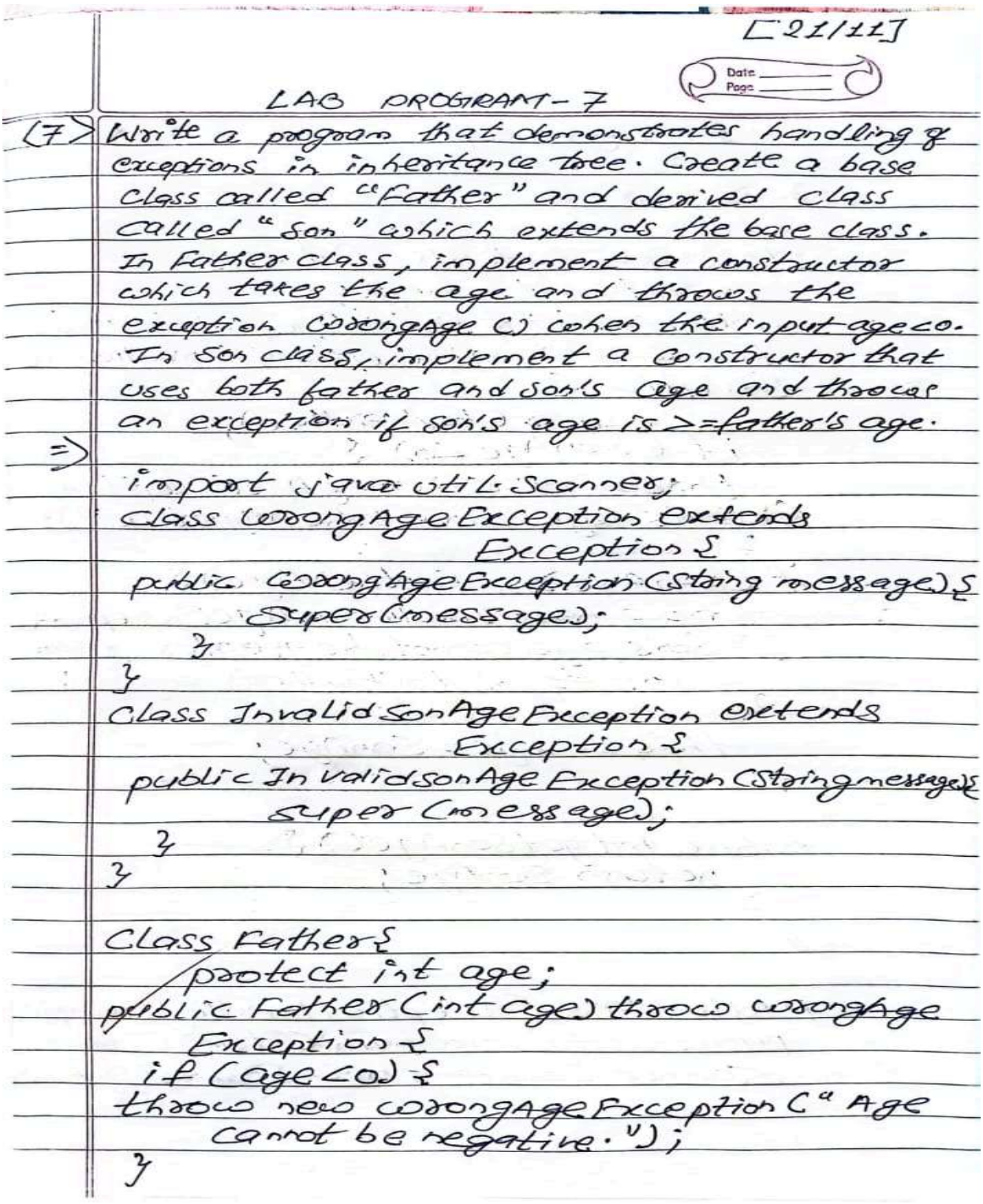
```
            students[i].inputCIEmarks();
            students[i].inputSEEmarks();
            students[i].calculateFinalMarks();
        }

        System.out.println("\nFinal Marks of Students:");
        for (int i = 0; i < n; i++) {
            System.out.println("\nStudent " + (i + 1) + ":");
            students[i].displayFinalMarks();
        }
    }
}
```

Output:

```
C:\java\bin\Packages>java Main
Enter number of students: 1

Enter details for student 1:
Enter USN: 1BM23CS294
Enter Name: SAMIR CHAUDHARY
Enter Semester: 3
Enter Internal Marks for 5 subjects:
Subject 1: 40
Subject 2: 43
Subject 3: 42
Subject 4: 43
Subject 5: 38
Enter SEE Marks for 5 subjects:
Subject 1: 43
Subject 2: 42
Subject 3: 38
Subject 4: 48
Subject 5: 38

Final Marks of Students:

Student 1:
USN: 1BM23CS294
Name: SAMIR CHAUDHARY
Semester: 3
Final Marks for 5 subjects:
Subject 1: 83
Subject 2: 85
Subject 3: 80
Subject 4: 91
Subject 5: 76
```

Program 7

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



[21/11]

Date
Page

LAB PROGRAM-7

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

=>

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

Class Father{
   protect int age;
public Father(int age) throws WrongAge
    Exception {
  if(age<0){
throw new WrongAgeException("Age
     cannot be negative.");
}
```

```
        this.age = age
    }
}

class Son extends Father{
    private int SonAge;
public Son(int FatherAge, int SonAge)
  throws WrongException, InvalidAge
  Exception{
        super(FatherAge);
    if(SonAge<0){
    throw new WrongAgeException
("Son's age cannot be negative");
    }
if(SonAge>=FatherAge){
  throw new InvalidSonAgeException
("Son's age cannot be greater than
        or equal to Father's age");
    }
      this.SonAge = SonAge;
}

public int getSonAge(){
    return SonAge;
    }
}

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

```
try {
    System.out.println("Enter Father's age:");
    int FatherAge = scanner.nextInt();

    System.out.println("Enter Son's age:");
    int SonAge = scanner.nextInt();

    Son son = new Son(FatherAge, SonAge);

    System.out.println("Father's age:" +
                        FatherAge);
    System.out.println("Son's age:" +
                        son.getSonAge());
}

catch (WrongAgeException | InvalidSonAge
                        : Exception e) {
    System.out.println("Error:" + e.get
                        message());
}
finally {
        scanner.close();
    }
}
```

Output:

case1:- Father's Age: 40
Son's Age: 10

Father's Age is : 40
Son's Age is : 10

case2:- Father's Age: 15
Son's Age : 20
Error: Son's Age cannot be greater than or equal to Father Age.

case3:- Father's Age: -1
Son's Age : 2
Error: Age cannot be negative

28/11/24

Code:

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

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


class Father {
    int age;

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

    public int getAge() {
        return age;
    }
}

class Son extends Father {
    private int sonAge;

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

```
    public int getSonAge() {
        return sonAge;
    }
}

public class ExceptionHandling {
    public static void main(String[] args) {
        Scanner sc = new 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); // Create Son object

            System.out.println("Father's age is: " + fatherAge);
            System.out.println("Son's age is: " + son.getSonAge());
        } catch (WrongAgeException | SonAgeException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            sc.close(); // Ensure the scanner is closed
        }
    }
}
```

Output:

```
C:\java\bin\1BM23CS294>java ExceptionHandling
Enter Father's Age: 40
Enter Son's Age: 10
Father's age is: 40
Son's age is: 10

C:\java\bin\1BM23CS294>java ExceptionHandling
Enter Father's Age: 15
Enter Son's Age: 20
Error: Son's age cannot be greater than or equal to father's age.

C:\java\bin\1BM23CS294>java ExceptionHandling
Enter Father's Age: -1
Enter Son's Age: 2
Error: Age cannot be negative.
```

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.

[28/11]

Date
Page

LAB PROGRAM-8

(8) Write a program which create two thread, one thread displaying "BMS college of Engineering" once every ten second and another display "CSE" once every two second.

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

    Thread thread1 = new thread(()->{
while(true){
        System.out.println("BMS College
                of Engineering");
try{
        Thread.sleep(10000);
    }
 catch(InterruptedException e){
        e.printstack.Trace();
    }
}
});

Thread thread2 = new thread(()->{
while(true){
        System.out.println("CSE");
try{
        Thread.sleep(2000);
    }
Catch(InterruptedException e){
        e.printStack.Trace();
    }
}
});
```

```
        thread1.start();
        thread2.start();
    }
}
```

Output:-

```
CSE
BMS college of Engineering
CSE
CSE
CSE
CSE
BMS college of Engineering
CSE
CSE
CSE
CSE
BMS college of Engineering
continue---
```

28/11/24

Code:

```
public class MultiThreadDemo {

    public static void main(String[] args) {

        // Thread for displaying "BMS College of Engineering" every 10 seconds
        Thread thread1 = new Thread(() -> {
            while (true) {
                System.out.println("BMS College of Engineering");
                try {
                    Thread.sleep(10000); // Sleep for 10 seconds
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

        // Thread for displaying "CSE" every 2 seconds
        Thread thread2 = new Thread(() -> {
            while (true) {
                System.out.println("CSE");
                try {
                    Thread.sleep(2000); // Sleep for 2 seconds
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

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

Output:

```
C:\java\bin\1BM23CS294>java MultiThreadDemo
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
```

PROGRAM 9

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

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

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

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

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

```
        outResult = new Label("Result:",Label.RIGHT);

        add(number
        1);
        add(num1);
        add(number
        2);
        add(num2);
        add(dResult
        );
        add(outRes
        ult);

        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(resu
                    ltNum); 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(),o
        utResult.getY()+outResult. getHeight()-8);
        else
        g.drawString(out,
        100,200); flag=0;
    }
```

Program 10
Demonstrate Interprocess communication and deadlock.

i. Demonstration of Interprocess communication

```
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.");
}
}
```

Output:

ii. Demonstration of deadlock

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

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

}
```

```
class Deadlock implements Runnable
 {
  A a = new A(); B b = new B();
  Deadlock( ) {
    Thread.currentThread().setName("MainThread");
    Thread t = new Thread(this, "RacingThread");
     t.start(); a.foo(b); // get lock on a in this thread.
     System.out.println("Back in main thread");
   }
 public void run() { b.bar(a); // get lock on b in other thread.
  System.out.println("Back in other thread");
  }
public static void main(String args[]) { new Deadlock(); }
```

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