B. M. S. COLLEGE OF ENGINEERING

(AUTONOMOUS COLLEGE UNDER VTU, BELGAUM)
BANGALORE – 560019

2022-23

LAB RECORD

OBJECT ORIENTED JAVA PROGRAMMING (23CS3PCOOJ)

Submitted by :

NAME : Simon Kevin M
USN : 1BM22CS277
SEMESTER: III
SECTION : E

Submitted to

Dr. Seema Patil
Assistant Professor
Dept. of CSE, BMSCE

12/10/24 LAB-1 1/A-12-12-24 PAGE EDGE Date: / /

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

```
import java.util.Scanner;
class Quadratic
{
  int a,b,c;
  double x1,x2,d;
  void get()
  {
    Scanner s = new Scanner(System.in);
    System.out.println("Enter the coefficients of a,b,c");
    a = s.nextInt();
    b = s.nextInt();
    c = s.nextInt();
  }
  void compute()
  {
    while (a==0)
    {
      System.out.println("Not a quadratic equation");
      System.out.println("Enter a non zero value for a");
      Scanner s = new Scanner(System.in);
      a = s.nextInt();
    }
    d = b*b-4*a*c;
    if (d==0)
    {
      x1 = (-b)/(2*a);
      System.out.println("Root are real and equal");
      System.out.println("Root 1 = Root 2 = "+x1);
    }
```

```
else if (d>0)
{
    r1 = ((-b) + (Math.sqrt(d))) / (2*a);
    r2 = ((-b) - (Math.sqrt(d))) / (2*a);
    System.out.println("Roots are real and distinct");
    System.out.println("Root1 = " + r1 + " Root2 = " + r2);
}
else if (d<0)
{
    System.out.println("Roots are imaginary");
    r1 = (-b)/(2*a);
    r2 = Math.sqrt(-d)/(2*a);
    System.out.println("Root1 = " + r1 + " + i" + r2);
    System.out.println("Root2 = " + r1 + " - i" + r2);
}
}
}
class QuadraticMain
{
    public static void main(String args[])
    {
        Quadratic q = new Quadratic();
        q.getd();
        q.compute();
        System.out.println("Simran Rao - 1BM22CS277");
    }
}
```

Output
Name: Simran Rao USN 1BM22CS277
Enter the coefficients of a,b,c
1
2
1
Roots are real and equal
Root1 = Root2 = -1.0
Simran Rao 1BM22CS277

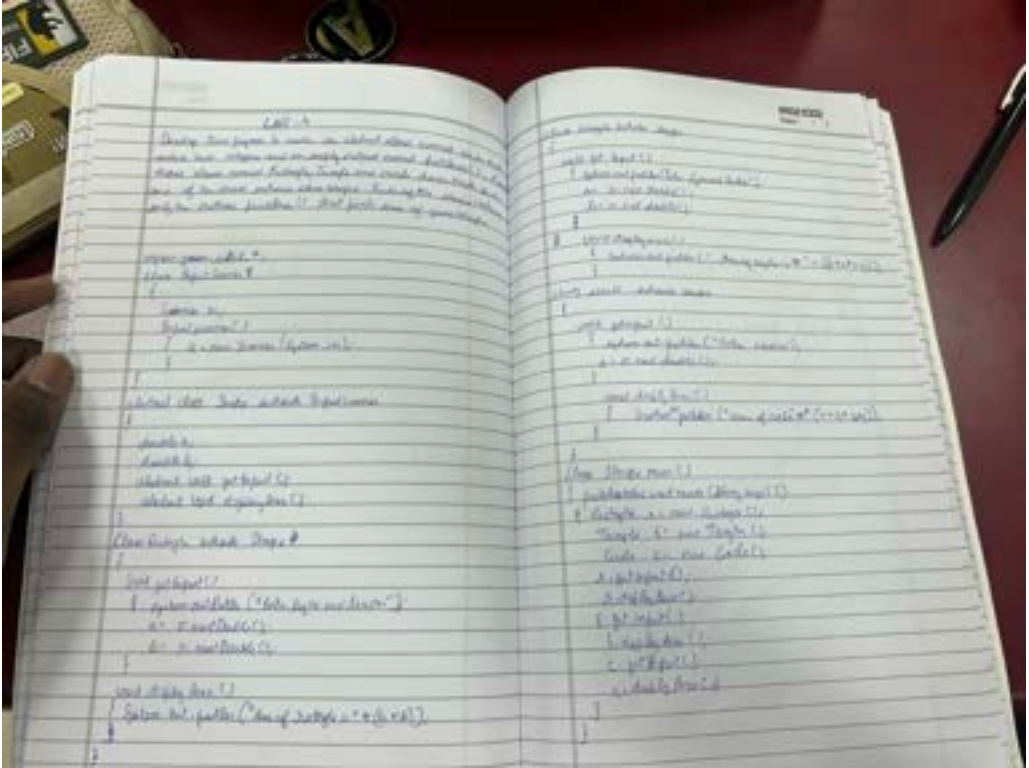
Enter the coefficients of a,b,c
1
4
3
Roots are real and distinct
Root1 = -1.0 Root2 = -3.0
Simran Rao 1BM22CS277

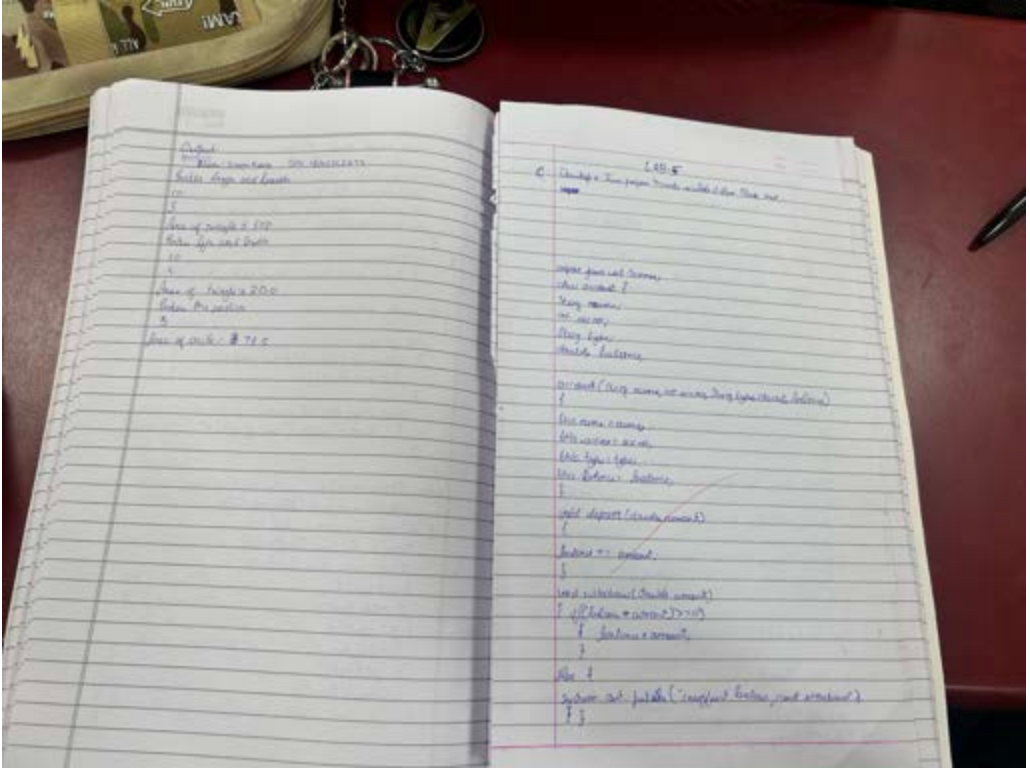
Enter the coefficients of a,b,c
1
1
2
Roots are imaginary
Root1 = 0.5 + i1.[illegible]
Root2 = 0.5 - i1.[illegible]
Simran Rao 1BM22CS277

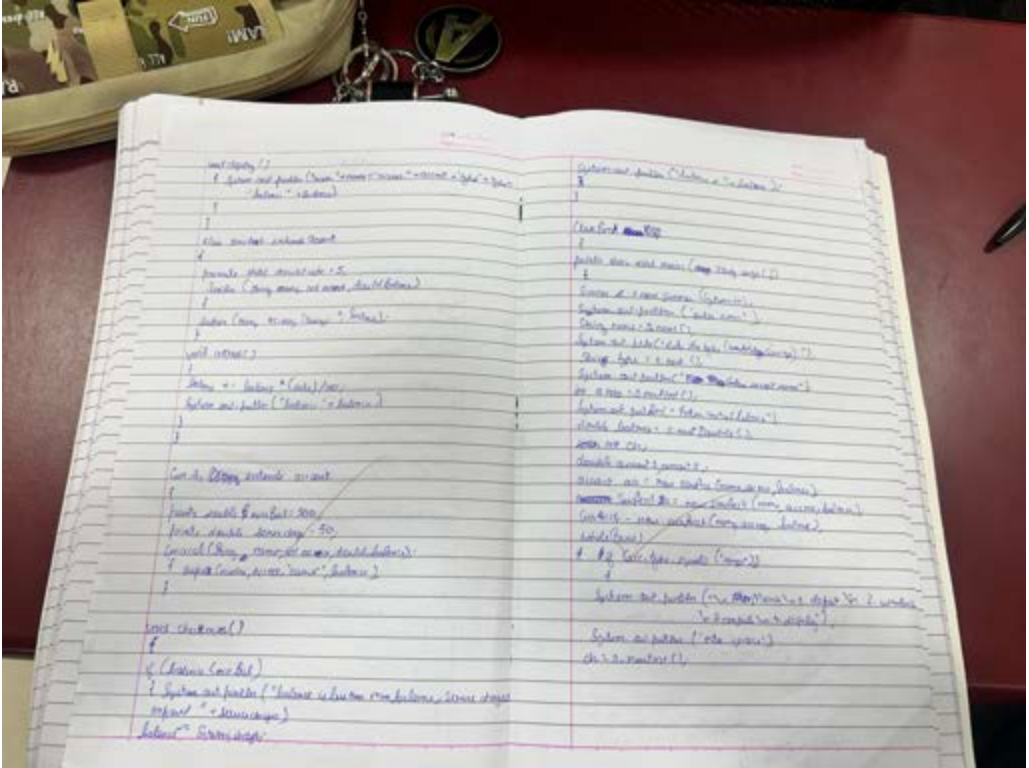
Dra-12-23

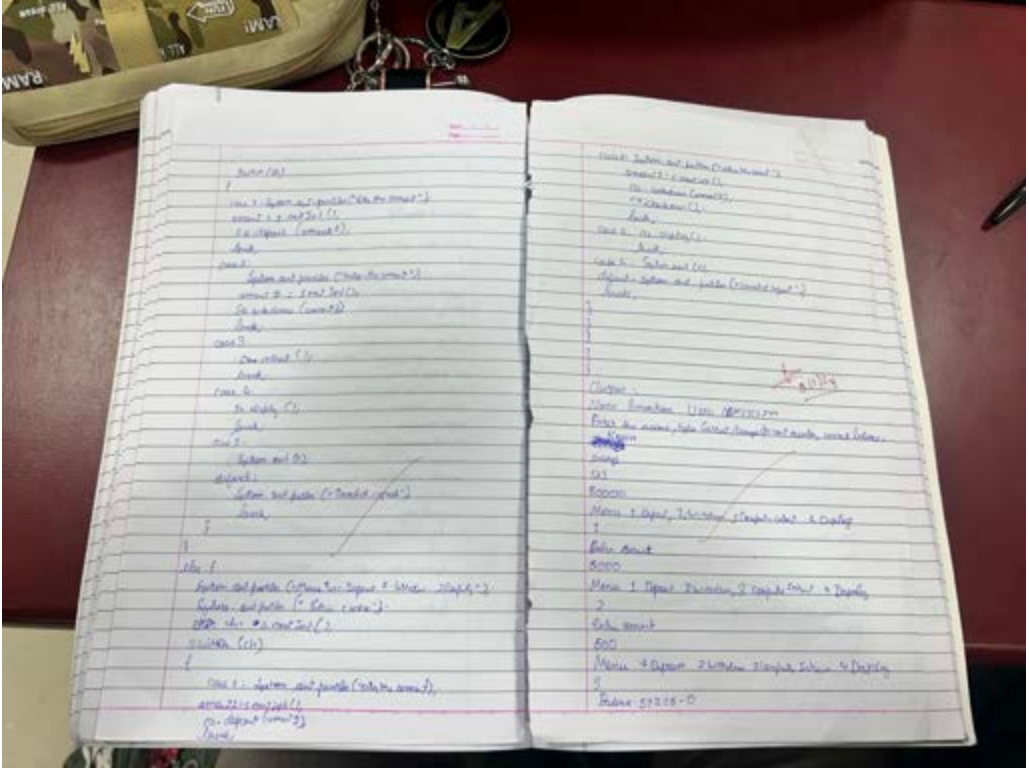
Q2) Develop Java Program to create class Student with members usn, name, four subjects and sum of marks. Include methods to accept and display details and a method to calculate SGPA of Student.

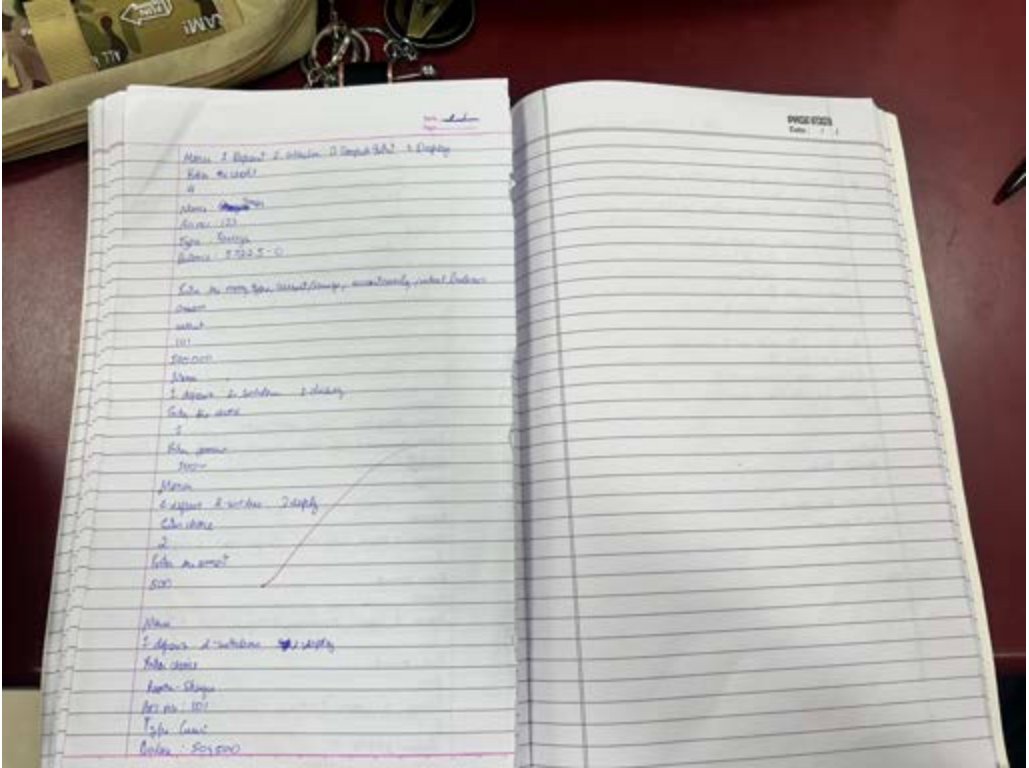
```
import java.util.*;
class Subject {
    int subjectMarks;
    int credits;
    int grade;
}
class Student {
    Subject subject[];
    String name;
    String usn;
    double sgpa;
    Scanner sc;

    Student() {
    sc =
    subject = new Subject[4];
    sc = new Scanner(System.in);
    for (int i=0; i<4; i++)
        subject[i] = new Subject();
    }
    void getStudentDetails() {
    System.out.println("Enter name");
    this.name = sc.next();
    System.out.println("Enter USN");
    this.usn = sc.next();
    }
void getMarks() {
for(int i=0; i<4; i++) {
System.out.println("Enter marks for subject " + (i+1) + ":");
subject[i].subjectMarks = sc.nextInt();
System.out.println("Enter credits for subject " + (i+1) + ":");
subject[i].credits = sc.nextInt();
if (subject[i].subjectMarks<=...)
    subject[i].grade = ...;
else if (subject[i].subjectMarks<=...)
    subject[i].grade = ...;
}
}
void computeSGPA() {
int effectiveScore = 0;
int totalCredits = 0;
for(int i=0; i<4; i++) {
    effectiveScore += (subject[i].grade * subject[i].credits);
    totalCredits += subject[i].credits;
}
sgpa = (double) effectiveScore/totalCredits;
}
}
class Main {
    public static void main(String[] args) {
    Student s1 = new Student();
    s1.getStudentDetails();
    s1.getMarks();
    s1.computeSGPA();
    System.out.println("Name" + s1.name);
    System.out.println("USN" + s1.usn);
    System.out.println("SGPA" + s1.sgpa);
    }
}
```

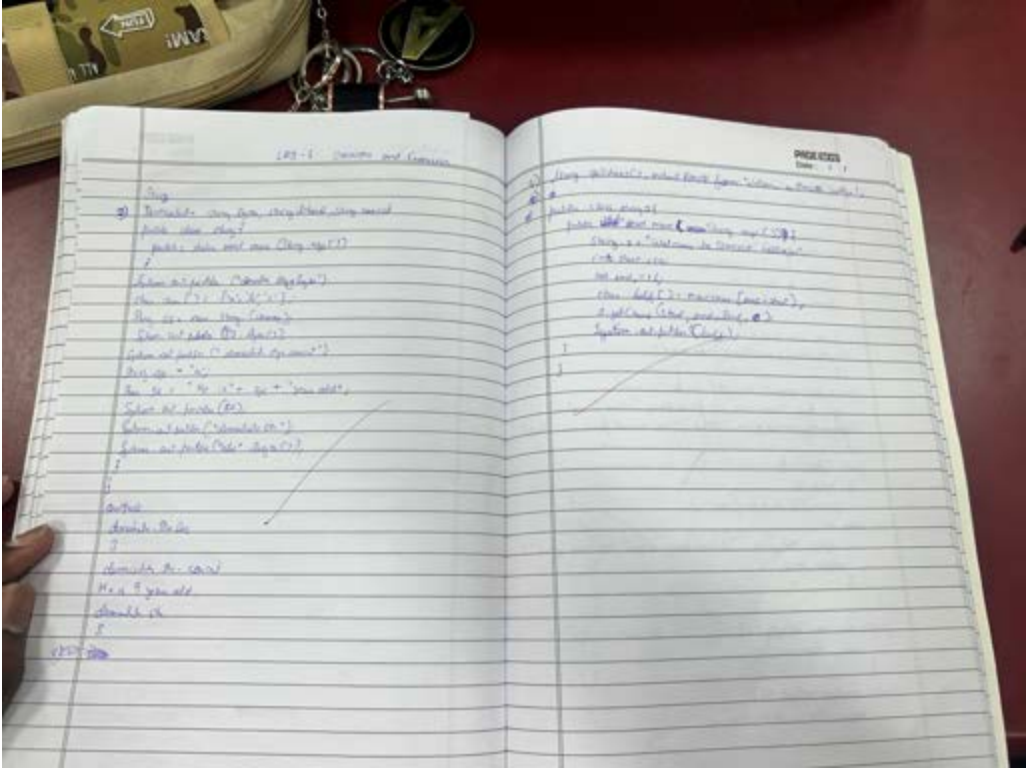


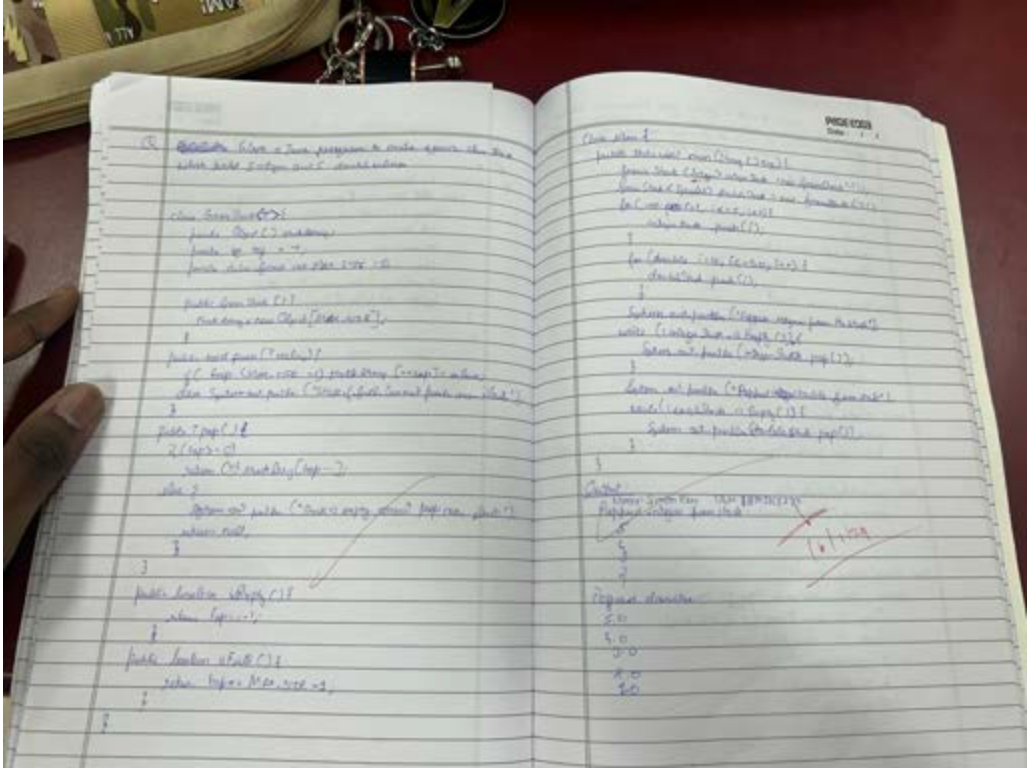


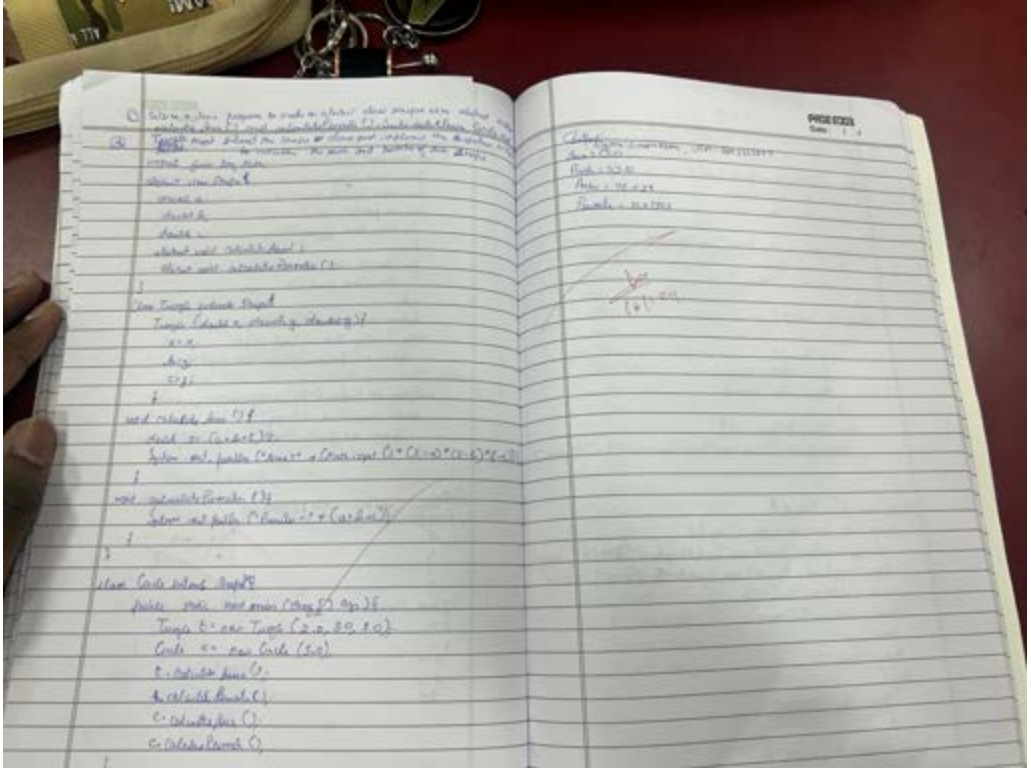








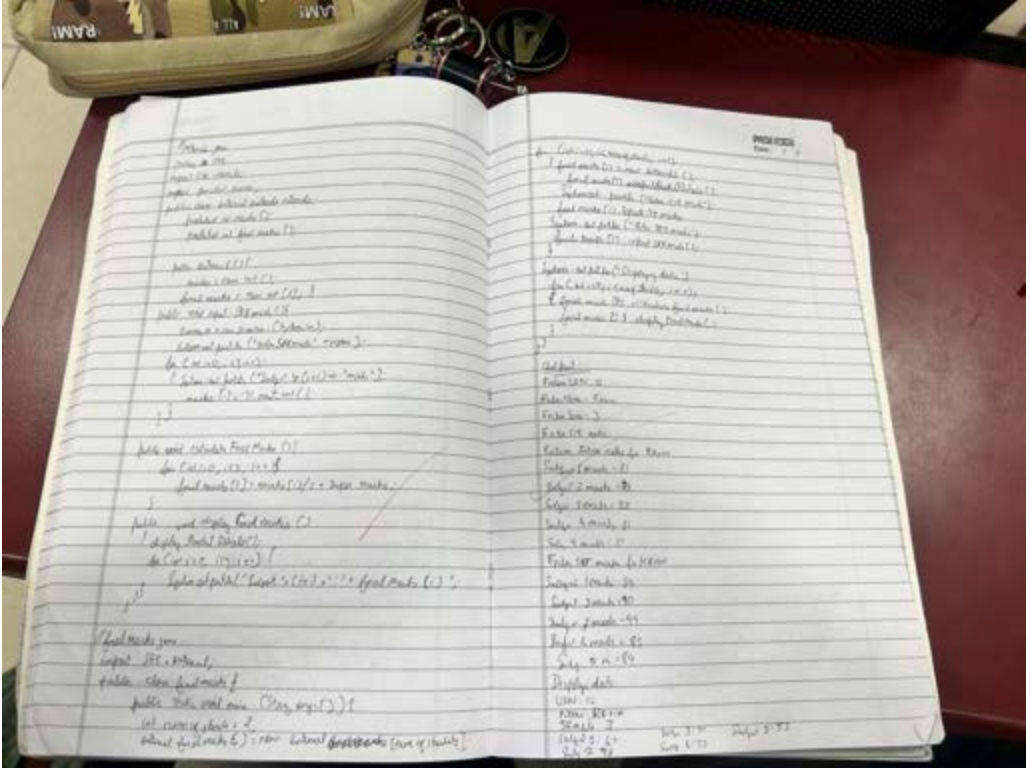




LAB-6 Packages

PAGE NO:
Date: / /

```
//Student.java
package CIE;
import java.util.Scanner;
public class Student {
    protected String usn;
    protected String name;
    protected int sem;
    public void inputStudentDetails() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter USN:");
        usn = sc.next();
        System.out.println("Enter name:");
        name = sc.next();
        System.out.println("Enter semester:");
        sem = sc.nextInt();
    }
    public void displayStudentDetails() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Sem: " + sem);
    }
}

//Internals.java
package CIE;
import java.util.Scanner;
public class Internals extends Student {
    protected int marks[] = new int[5];
    public void inputInternalMarks() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter internal marks for " + name);
        for (int i = 0; i < 5; i++) {
            System.out.println("Subject " + (i+1) + " marks:");
            marks[i] = sc.nextInt();
        }
    }
```



Lab 7 PAGE 6003
Date: / /

Assignment

import java.util.Scanner;
class InvalidAgeException extends Exception {
    public InvalidAgeException(String s) {
        super(s);
    }
}
class InputScanner {
    Scanner sc = new Scanner(System.in);
}
class Father extends InputScanner {
    int fatherAge;
    public Father() throws InvalidAgeException {
        System.out.println("Enter father's age:");
        fatherAge = sc.nextInt();
        if (fatherAge < 0) {
            throw new InvalidAgeException("Age cannot be negative");
        }
    }
    public void display() {
        System.out.println("Father's age: " + fatherAge);
    }
}
class Son extends Father {
    int sonAge;
    public Son() throws InvalidAgeException {
        super();
        System.out.println("Enter son's age:");
        sonAge = sc.nextInt();
        if (sonAge < 0) {
            throw new InvalidAgeException("Son's age cannot be less than father's age");
        } else if (sonAge < 0) {
            throw new InvalidAgeException("Age cannot be negative");
        }
    }
    public void display() {
        super.display();
        System.out.println("Son's age: " + sonAge);
    }
}

```
public class Demo {
    public static void main(String[] args) {
        try {
            Son son = new Son();
            son.display();
        } catch (WrongAge e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

Enter Father Age:
10
Enter Son age:
5
Error: Son age cannot be greater than Father age

28/11/24

LAB-8

```
class DisplayThread extends Thread {
    private final String message;
    private final long interval;

    DisplayThread(String message, long interval) {
        this.message = message;
        this.interval = interval;
    }

    public void run() {
        try {
            while (true) {
                System.out.println(message);
                Thread.sleep(interval);
            }
        } catch (InterruptedException e) {
            System.out.println(Thread.currentThread().getName() + " Stopped.");
        }
    }
}

public class ThreadDemo {
    public static void main(String[] args) {
        DisplayThread t1 = new DisplayThread("BMSCE", 2000);
        DisplayThread t2 = new DisplayThread("CSE", 3000);
        t1.setName("Thread-1");
        t2.setName("Thread-2");
        t1.start();
        t2.start();
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted");
        }
```

```
        Thread t = Thread.currentThread();
        t.setName("Main Thread");
        System.out.println("Current thread: " + t);
    }
}
```

```
BMSCE
CSE
CSE
CSE
CSE
BMSCE
CSE
CSE
CSE
CSE
BMSCE
CSE
CSE
CSE
CSE
CSE
Main Thread exiting.
Thread-1 Stopped.
Thread-2 Stopped.
```

Lab: #10

PAGE EDGE
Date: / /

Demonstrate Interruption Handling

```
class Q8 {
int n;
boolean valueSet = false;
synchronized int get() {
while (!valueSet)
try {
System.out.println("\n Consumer waiting");
wait();
} catch (InterruptedException e) {
System.out.println("Interrupted Exception");
}
System.out.println("Got: " + n);
valueSet = false;
System.out.println("Consumer Returning");
notify();
return n;
}
synchronized void put(int n) {
while (valueSet)
try {
System.out.println("Producer waiting");
wait();
} catch (InterruptedException e) {
System.out.println("Interrupted Exception");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("Interrupted (Done)");
notify();
}}
class Producer implements Runnable {
Q8 q;
Producer(Q8 q) {
```

```
this.q = q;
new Thread(this, "Producer").start();
}
public void run() {
int i = 0;
while (i < 5) {
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 < 5) {
int x = q.get();
System.out.println("consumed " + x);
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:

```
Press Control-C to stop
Put : 0
Intimate Consumer
Producer waiting
Got : 0
Intimate Producer
Put : 1
Intimate Consumer
Consumed : 0
Producer waiting
Got : 1
Intimate Producer
consumed 1
Put : 2
Intimate Producer
consumed 1
Put : 2
Intimate Consumer
Producer waiting
[illegible]
Intimate Producer
Consumed 2
Put : 3
Intimate Consumer
Producer waiting
Got : 3
Intimate Producer
Consumer waiting
[illegible]
```

LAB-7

PAGE EDGE
Date: / /

Q. [illegible] that create a [illegible] [illegible] [illegible] [illegible] [illegible] two fields, [illegible]

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

class Swing Demo {
  Swing Demo() {
    JFrame jfrm = new JFrame("[illegible] App");
    jfrm.setSize(275, 100);
    jfrm.setLayout(new FlowLayout());
    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JLabel jlab = new JLabel("[illegible]");
    JTextField [illegible] = new JTextField(8);
    JTextField [illegible] = new JTextField(8);
    JButton button = new JButton("[illegible]");
    JLabel [illegible] = new JLabel();
    JLabel [illegible] = new JLabel();
    JLabel [illegible] = new JLabel();
    JLabel [illegible] = new JLabel();
    jfrm.add([illegible]);
    jfrm.add(jlab);
    jfrm.add([illegible]);
    jfrm.add([illegible]);
    jfrm.add(button);
    jfrm.add([illegible]);
    jfrm.add([illegible]);
    jfrm.add([illegible]);
ActionListener l = new ActionListener() {
    public void actionPerformed(ActionEvent ae) {
        System.out.println("[illegible]"); };
```

[illegible]

Functions (Def)

- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]

[illegible]

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

```
import java.util.Scanner;


class Quadratic {
   int a, b, c;
   double r1, r2, d;


   void getd() {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter the coefficients of a,b,c");
      a = s.nextInt();
      b = s.nextInt();
      c = s.nextInt();
   }


   void compute() {
      while (a == 0) {
         System.out.println("Not a quadratic equation");
         System.out.println("Enter a non zero value for a:");
         Scanner s = new Scanner(System.in);
         a = s.nextInt();
      }
      d = b * b - 4 * a * c;
      if (d == 0) {


         r1 = (-b) / (2 * a);
         System.out.println("Roots are real and equal");
         System.out.println("Root1 = Root2 = " + r1);
      } else if (d > 0) {
         r1 = ((-b) + (Math.sqrt(d))) / (double) (2 * a);
```

```
            r2 = ((-b) - (Math.sqrt(d))) / (double) (2 * a);
            System.out.println("Roots are real and distinct");
            System.out.println("Roo1 = " + r1 + " Root2 = " + r2);
        } else if (d < 0) {
            System.out.println("Roots are imaginary");
            r1 = (-b) / (2 * a);
            r2 = Math.sqrt(-d) / (2 * a);
            System.out.println("Root1 = " + r1 + " + i" + r2);
            System.out.println("Root1 = " + r1 + " - i" + r2);
        }

    }
}

class QuadraticMain {
    public static void main(String args[]) {
        Quadratic q = new Quadratic();
        q.getd();

        q.compute();
        System.out.println("Shreyas Rao M-1BM22CS272");
    }
}
```

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.*;


class Subject {
   int subjectMarks;
   int credits;
   int grade;
}


class Student {
   Subject subject[];
   String name;
   String usn;
   double sgpa;
   Scanner sc;


   Student() {
      int i;
      subject = new Subject[8];
      sc = new Scanner(System.in);
      for (i = 0; i < 8; i++)
         subject[i] = new Subject();
   }


   void getstudentdetails() {
      System.out.println("Enter your name:");
      this.name = sc.next();
      System.out.println("Enter your USN:");
      this.usn = sc.next();
   }
```

```
    void getMarks() {
        for (int i = 0; i < 8; i++) {
            System.out.println("Enter marks for subject " + (i + 1) + ":");
            subject[i].subjectMarks = sc.nextInt();
            System.out.println("Enter credits for subject" + (i + 1) + ":");
            subject[i].credits = sc.nextInt();
            subject[i].grade = subject[i].subjectMarks / 10 + 1;
            if (subject[i].grade == 11)
                subject[i].grade = 10;
            if (subject[i].grade <= 4)
                subject[i].grade = 0;
        }
    }

    void computeSGPA() {
        int effectiveScore = 0;
        int totalCredits = 0;
        for (int i = 0; i < 8; i++) {
            effectiveScore += (subject[i].grade * subject[i].credits);
            totalCredits += subject[i].credits;
        }
        sgpa = (double) effectiveScore / (double) totalCredits;
    }

}

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

```
        s1.getMarks();
        s1.computeSGPA();
        System.out.println("Name=" + s1.name);
        System.out.println("USN:" + s1.usn);
        System.out.println("SGPA=" + s1.sgpa);
    }
}
```

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

```
import java.util.Scanner;


class Books
{
	String name;
	String author;
	int price;
	int numPages;

	Books(String name,String author,int price,int numPages)
	{
		this.name=name;
		this.author=author;
		this.price=price;
		this.numPages=numPages;
	}

	public String toString()
	{
		String name,author,price,numPages;
		name="Book name:" +this.name+ "\n";
		author="Author name:" +this.author+ "\n";
		price="Price:" +this.price+ "\n";
		numPages="Number of pages:" +this.numPages+ "\n";
		return name+author+price+numPages;
	}
}
```

```
public class Mainbook
{
	public static void main(String args[])
	{
		Scanner s=new Scanner(System.in);
		int n;
		int i;
		String name;
		String author;
		int price;
		int numPages;

		System.out.println("Enter the number of books:");
		n=s.nextInt();

		Books b[];
		b=new Books[n];

		for(i=0;i<n;i++)
		{
			System.out.println("Enter the details of book" + (i+1) + ":");
			System.out.println("Enter the name of the book:");
			name=s.next();
			System.out.println("Enter the author name:");
			author=s.next();
			System.out.println("Enter the price:");
			price=s.nextInt();
			System.out.println("Enter the number of pages:");
			numPages=s.nextInt();
```

```
                b[i]=new Books(name,author,price,numPages);
            }

            System.out.println("Book Details:");
            for(i=0;i<n;i++)
            {
                System.out.println(b[i]);
            }
        }
}
```

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.*;
import java.math.*;

class InputScanner {

   Scanner sc;

   InputScanner() {
      sc = new Scanner(System.in);
   }

}

abstract class Shape extends InputScanner {

   double a;

   double b;

   abstract void getInput();

   abstract void displayArea();

}

class Rectangle extends Shape {
   void getInput() {
```

```
        System.out.println("Enter the length and breadth:");
        a = sc.nextDouble();
        b = sc.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of rectangle is :" + (a * b));
    }
}

class Triangle extends Shape {
    void getInput() {
        System.out.println("Enter the length and height:");
        a = sc.nextDouble();
        b = sc.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of triangle is :" + (a * b * 0.5));
    }
}

class Circle extends Shape {
    void getInput() {
        System.out.println("Enter the radius:");
        a = sc.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of circle is :" + (a * a * Math.PI));
    }
```

```
}


class ShapeMain {
    public static void main(String[] args) {
        Rectangle r = new Rectangle();
        Triangle t = new Triangle();
        Circle c = new Circle();
        r.getInput();
        r.displayArea();
        t.getInput();
        t.displayArea();
        c.getInput();
        c.displayArea();
    }
}
```

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;

class account {
   String name;
   int accno;
   String type;
   double balance;

   account(String name, int accno, String type, double balance) {
      this.name = name;
      this.accno = accno;
      this.type = type;
      this.balance = balance;
   }

   void deposit(double amount) {
```

```
        balance += amount;
    }

    void withdraw(double amount) {
        if ((balance - amount) >= 0) {
            balance -= amount;
        } else {
            System.out.println("Insufficient balance,cant withdraw");
        }
    }

    void display() {
        System.out.println("Name:" + name + "\nAccno:" + accno + "\nType:" + type + "\nBalance:" +
balance);
    }
}

class savAcct extends account {

    private static double rate = 5;

    savAcct(String name, int accno, double balance) {
        super(name, accno, "Savings", balance);

    }

    void interest() {
        balance += balance * (rate) / 100;
        System.out.println("Balance:" + balance);
    }
```

```
}

class curAcct extends account {

   private double minBal = 500;
   private double serviceCharges = 50;

   curAcct(String name, int accno, double balance) {
      super(name, accno, "Current", balance);

   }

   void checkmin() {

      if (balance < minBal) {
         System.out.println("Balance is less than min balance,service charges imposed:" +
serviceCharges);
         balance -= serviceCharges;
         System.out.println("Balance is:" + balance);
      }

   }

}

class Bank {
   public static void main(String a[]) {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter the name,type(current/savings),account number,initial balance:");
      String name = s.next();
      String type = s.next();
```

```
int accno = s.nextInt();
double balance = s.nextDouble();
int ch;
double amount1, amount2;
account acc = new account(name, accno, type, balance);
savAcct sa = new savAcct(name, accno, balance);
curAcct ca = new curAcct(name, accno, balance);
while (true) {
   if (acc.type.equals("savings")) {
      System.out.println("\nMenu\n1.deposit 2.withdraw 3.compute interest 4.display");
      System.out.println("Enter the choice:");
      ch = s.nextInt();
      switch (ch) {
         case 1:
            System.out.println("Enter the amount:");
            amount1 = s.nextInt();
            sa.deposit(amount1);
            break;
         case 2:
            System.out.println("Enter the amount:");
            amount2 = s.nextInt();
            sa.withdraw(amount2);
            break;
         case 3:
            sa.interest();
            break;
         case 4:
            sa.display();
            break;
         case 5:
            System.exit(0);
```

```
        default:
            System.out.println("invalid input");
            break;
    }
} else {
    System.out.println("\nMenu\n1.deposit 2.withdraw  3.display");
    System.out.println("Enter the choice:");
    ch = s.nextInt();
    switch (ch) {
        case 1:
            System.out.println("Enter the amount:");
            amount1 = s.nextInt();
            ca.deposit(amount1);
            break;
        case 2:
            System.out.println("Enter the amount:");
            amount2 = s.nextInt();
            ca.withdraw(amount2);
            ca.checkmin();
            break;

        case 3:
            ca.display();
            break;
        case 4:
            System.exit(0);
        default:
            System.out.println("Invalid input");
            break;
    }
}
```

```
    }


  }
}
```

GENERICS

```
class GenericStack<T> {
    private Object[] stackArray;
    private int top = -1;
    private static final int MAX_SIZE = 5;

    public GenericStack() {
        stackArray = new Object[MAX_SIZE];
    }

    public void push(T value) {
        if (top < MAX_SIZE - 1) stackArray[++top] = value;
        else System.out.println("Stack is full. Cannot push more elements.");
    }

    @SuppressWarnings("unchecked")
    public T pop() {
        if (top >= 0)
            return (T) stackArray[top--];
        else {
            System.out.println("Stack is empty. Cannot pop more elements.");
            return null;
        }
    }

    public boolean isEmpty() {
        return top == -1;
    }

    public boolean isFull() {
        return top == MAX_SIZE - 1;
```

```
    }
}
class Main{
public static void main(String[] args) {
        GenericStack<Integer> integerStack = new GenericStack<>();
        GenericStack<Double> doubleStack = new GenericStack<>();

        // Push integers to the integer stack
        for (int i = 1; i <= 5; i++) {
            integerStack.push(i);
        }

        // Push doubles to the double stack
        for (double i = 1.0; i <= 5.0; i++) {
            doubleStack.push(i);
        }

        // Pop and print integers from the integer stack
        System.out.println("Popped integers from the stack:");
        while (!integerStack.isEmpty()) {
            System.out.println(integerStack.pop());
        }

        // Pop and print doubles from the double stack
        System.out.println("Popped doubles from the stack:");
        while (!doubleStack.isEmpty()) {
            System.out.println(doubleStack.pop());
        }
    }
}
```

Abstract class prog

```
import java.lang.Math;

abstract class Shape {
   double a;
   double b;
   double c;

   abstract void calculateArea();

   abstract void calculatePerimeter();
}

class Triangle extends Shape {
   Triangle(double x, double y, double z) {
      a = x;
      b = y;
      c = z;
   }

   void calculateArea() {
      double s = (a + b + c) / 2;
      System.out.println("Area=" + (Math.sqrt(s * (s - a) * (s - b) * (s - c))));
   }

   void calculatePerimeter() {
      System.out.println("Perimeter=" + (a + b + c));
   }
}

class Circle extends Shape {
```

```
    Circle(double r) {
        a = r;
    }

    void calculateArea() {
        System.out.println("Area=" + (Math.PI * a * a));
    }

    void calculatePerimeter() {
        System.out.println("Perimeter=" + (2 * Math.PI * a));
    }
}

class ShapeM {
    public static void main(String[] args) {
        Triangle t = new Triangle(2.0, 3.0, 5.0);
        Circle c = new Circle(5.0);
        t.calculateArea();
        t.calculatePerimeter();
        c.calculateArea();
        c.calculatePerimeter();
    }
}
```

6. Create a package CIE which has two classes- Student and Internals. The class Student has members like usn, name, sem. The class Internals derived from Student 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 Internals extends Student {
   protected int marks[] = new int[5];

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

public class Student {
   protected String usn = new String();
```

```
    protected String name = new String();
    protected int sem;

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

import CIE.Internals;

import java.util.Scanner;

public class Externals extends Internals {
    protected int marks[];
    protected int finalMarks[];

    public Externals() {
        marks = new int[5];
```

```
        finalMarks = new int[5];
    }

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

    public void calculateFinalMarks() {
        for (int i = 0; i < 5; i++)
            finalMarks[i] = marks[i] / 2 + super.marks[i];
    }

    public void displayFinalMarks() {
        displayStudentDetails();
        for (int i = 0; i < 5; i++)
            System.out.println("Subject " + (i + 1) + ": " + finalMarks[i]);
    }
}
import SEE.Externals;

public class Pkgmain {
    public static void main(String args[]) {
        int numOfStudents = 2;
        Externals finalMarks[] = new Externals[numOfStudents];

        for (int i = 0; i < numOfStudents; i++) {
```

```
            finalMarks[i] = new Externals();
            finalMarks[i].inputStudentDetails();
            System.out.println("Enter CIE marks");
            finalMarks[i].inputCIEmarks();
            System.out.println("Enter SEE marks");
            finalMarks[i].inputSEEmarks();
        }

        System.out.println("Displaying data:\n");

        for (int i = 0; i < numOfStudents; i++) {
            finalMarks[i].calculateFinalMarks();
            finalMarks[i].displayFinalMarks();
        }
    }
}
```

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 cases both father and son's age and throws an exception if son's age is

>=father's age.

```
import java.util.Scanner;


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

class InputScanner
{
 protected Scanner s;
 public InputScanner()
 {
  s = new Scanner(System.in);
 }
}



class Father extends InputScanner
{
 protected int fatherAge;
 public Father() throws WrongAge
```

```
{
 System.out.println("Enter Father's Age:");
 fatherAge=s.nextInt();

 if(fatherAge<0)
 {
  throw new WrongAge("Age cannot be negetive:");
 }
}

public void display()
{
 System.out.println("Father's Age:" + fatherAge);
}

}


class Son extends Father
{
 private int sonAge;

 public Son() throws WrongAge
 {
  super();
  System.out.println("Enter Son's age:");
  sonAge=s.nextInt();


  if(sonAge>fatherAge)
  {
```

```
  throw new WrongAge("Son's age cannot be greater than father's age");
 }
 else if (sonAge<0)
 {
  throw new WrongAge("Age cannot be negative");
 }
}

 public void display()
 {
  super.display();
  System.out.println("Son's Age: " + sonAge);
 }

}


public class FatherSonAge
{
 public static void main(String args[])
 {
  try
  {
   Son son=new Son();
   son.display();
  }

  catch (WrongAge e)
  {
   System.out.println("Error: " + e.getMessage());
  }
```

```
    }

}
```

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

```
class DisplayMessageThread extends Thread {
   private final String message;
   private final long interval;

   DisplayMessageThread(String message, long interval) {
      this.message = message;
      this.interval = interval;
   }

   public void run() {
      try {
         while (true) {
            System.out.println(message);
            Thread.sleep(interval);
         }
      } catch (InterruptedException e) {
         System.out.println(Thread.currentThread().getName() + " interrupted.");
      }
   }
}

public class TwoThreadDemo {
   public static void main(String[] args) {
      DisplayMessageThread thread1 = new DisplayMessageThread("BMS College of Engineering",
10000);
      DisplayMessageThread thread2 = new DisplayMessageThread("CSE", 2000);

      thread1.setName("Thread 1");
      thread2.setName("Thread 2");
```

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

        try {
            Thread.sleep(30000);
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted.");
        }

        thread1.interrupt();
        thread2.interrupt();

        System.out.println("Main thread exiting.");
    }
}
```

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 javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class SwingDemo {
  SwingDemo() {
    // create jframe container
    JFrame jfrm = new JFrame("Divider App");
    jfrm.setSize(275, 150);
    jfrm.setLayout(new FlowLayout());
    // to terminate on close
    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // text label
    JLabel jlab = new JLabel("Enter the divider and divident:");

    // add text field for both numbers
    JTextField ajtf = new JTextField(8);
    JTextField bjtf = new JTextField(8);

    // calc button
    JButton button = new JButton("Calculate");

    // labels
    JLabel err = new JLabel();
    JLabel alab = new JLabel();
```

```
JLabel blab = new JLabel();
JLabel anslab = new JLabel();

// add in order :)
jfrm.add(err); // to display error bois
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);

ActionListener l = new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
      System.out.println("Action event from a text field");
   }
};
ajtf.addActionListener(l);
bjtf.addActionListener(l);

button.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
      try {
         int a = Integer.parseInt(ajtf.getText());
         int b = Integer.parseInt(bjtf.getText());
         int ans = a / b;

         alab.setText("\nA = " + a);
         blab.setText("\nB = " + b);
         anslab.setText("\nAns = " + ans);
```

```
            } catch (NumberFormatException e) {
                alab.setText("");
                blab.setText("");
                anslab.setText("");
                err.setText("Enter Only Integers!");
            } catch (ArithmeticException e) {
                alab.setText("");
                blab.setText("");
                anslab.setText("");
                err.setText("B should be NON zero!");
            }
          }
        });

        // display frame
        jfrm.setVisible(true);
    }

    public static void main(String args[]) {
        // create frame on event dispatching thread
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new SwingDemo();
            }
        });
    }
}
```

10a) Interprocess communication using consumer and producer

```
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 < 6) {
            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 < 6) {
            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.");
    }
}
```

10b) 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();
   }

   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();
   }
```

```
    void last() {
        System.out.println("Inside B.last");
    }
}

class Deadlock implements Runnable {
    A a = new A();
    B b = new B();

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