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);
    }
```

```
class (abc)
{
    a=(-b+(Math.sqrt(b*b)-4*a*c))/(2*a);
    b=(-b-(Math.sqrt(b*b)-4*a*c));
    System.out.println("Root are not real");
    System.out.println("Root1="+a+"Root2="+b);
}
else if(a==c)
{
    System.out.println("Root are same");
    a=(-b)/(2*a);
    b=Math.sqrt(-b*b+4*a*c);
    System.out.println("Root1="+a+"+i"+b+a);
    System.out.println("Root2="+a+"-i"+b+a);
}
}
}
class Quadratic Main
{
    public static void main(String args[])
    {
        Quadratic q= new Quadratic();
        q.get();
        q.compute();
        System.out.println("Sriram Ram - 1BM22CS277");
    }
}
```

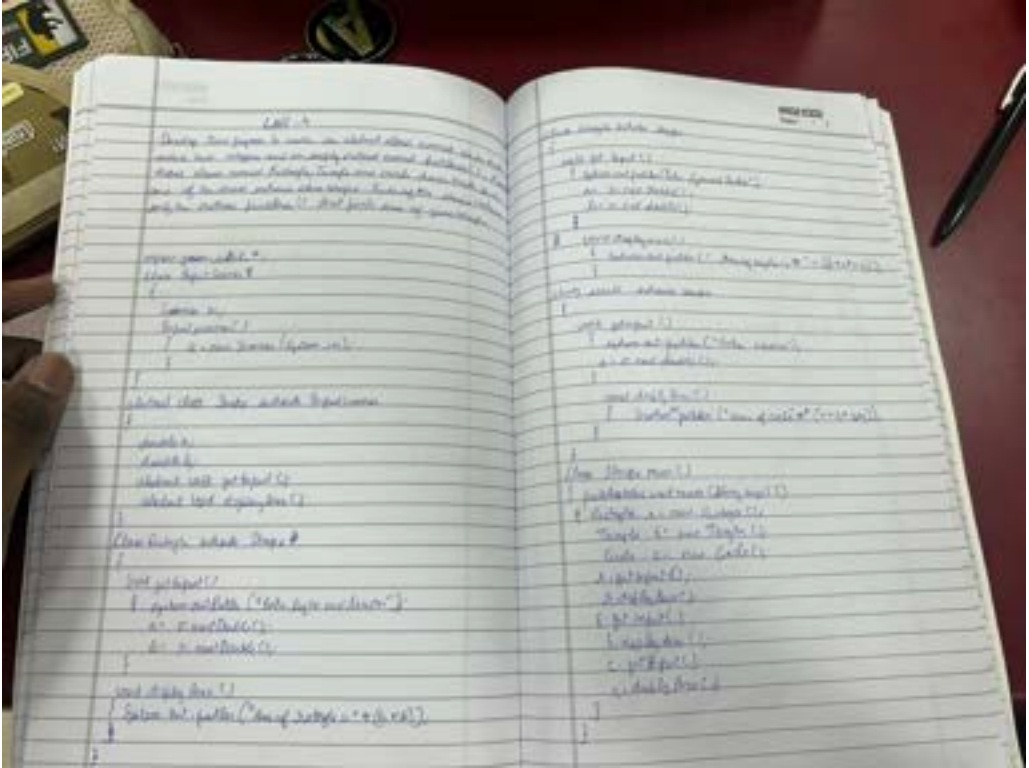
Output

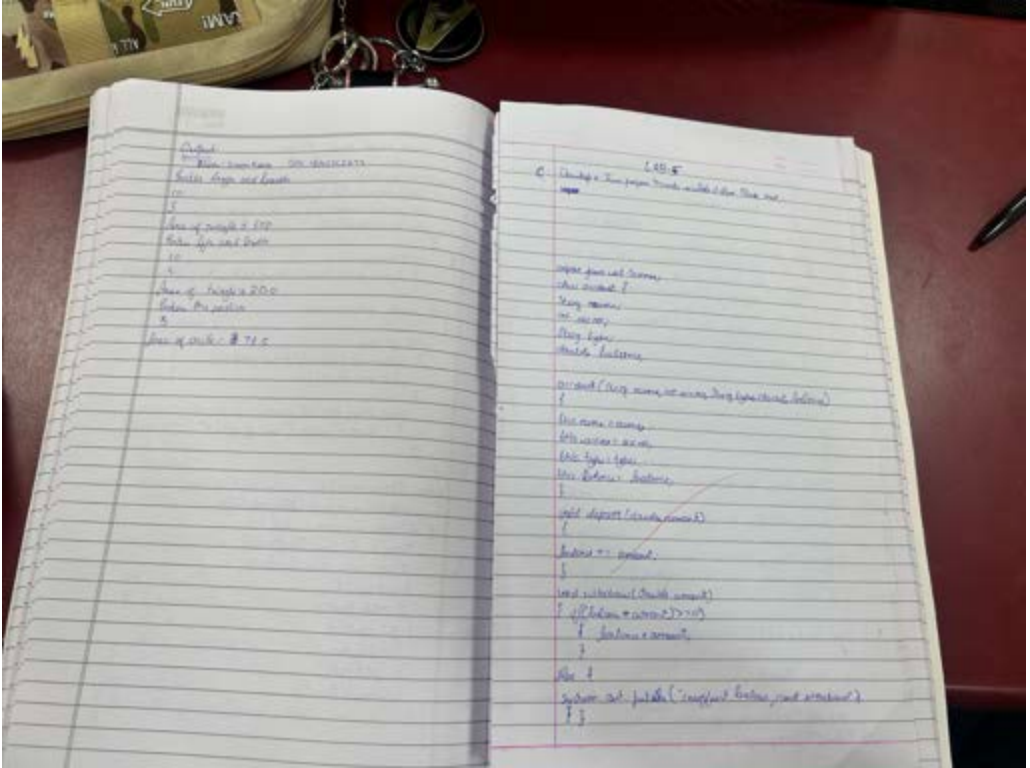
```
Sriram Ram USN 1BM22CS277
Enter the coefficient of abc
1
6
9
Root are real and equal
Root1: Root2 : -3.0
Sriram Ram 1BM22CS277

Enter the coefficient of abc
1
2
3
Root are imaginary
Root1= -1.0+1.4142135623730951i
Root2= -1.0-1.4142135623730951i
Sriram Ram 1BM22CS277
```

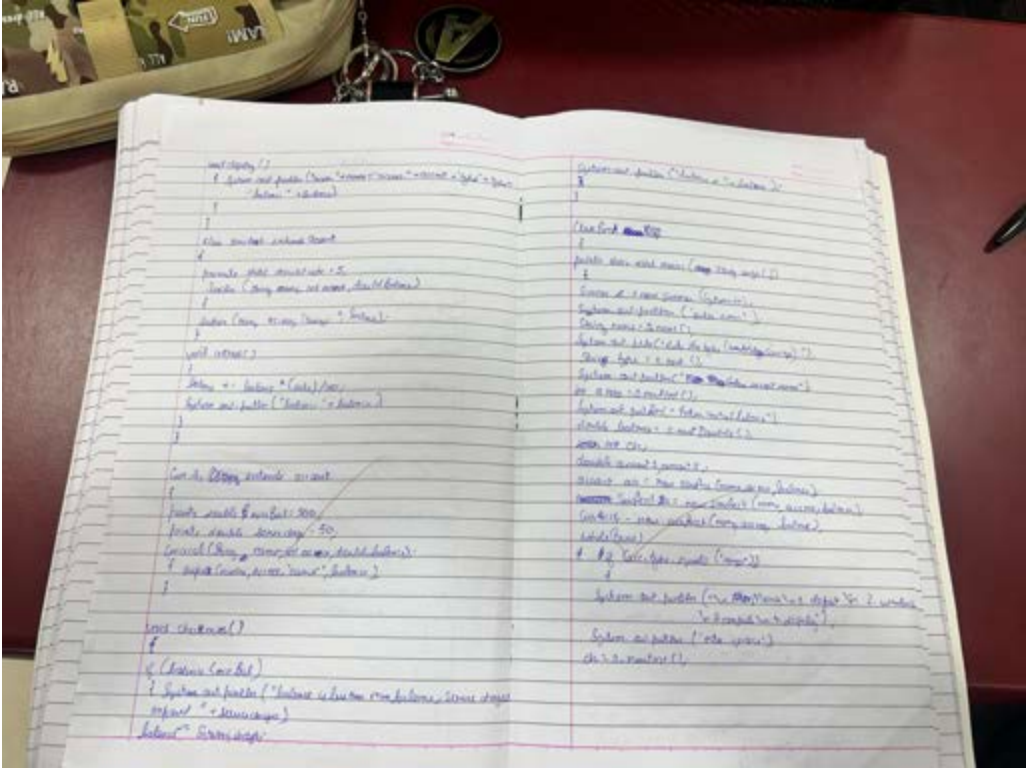
Dra-12-23

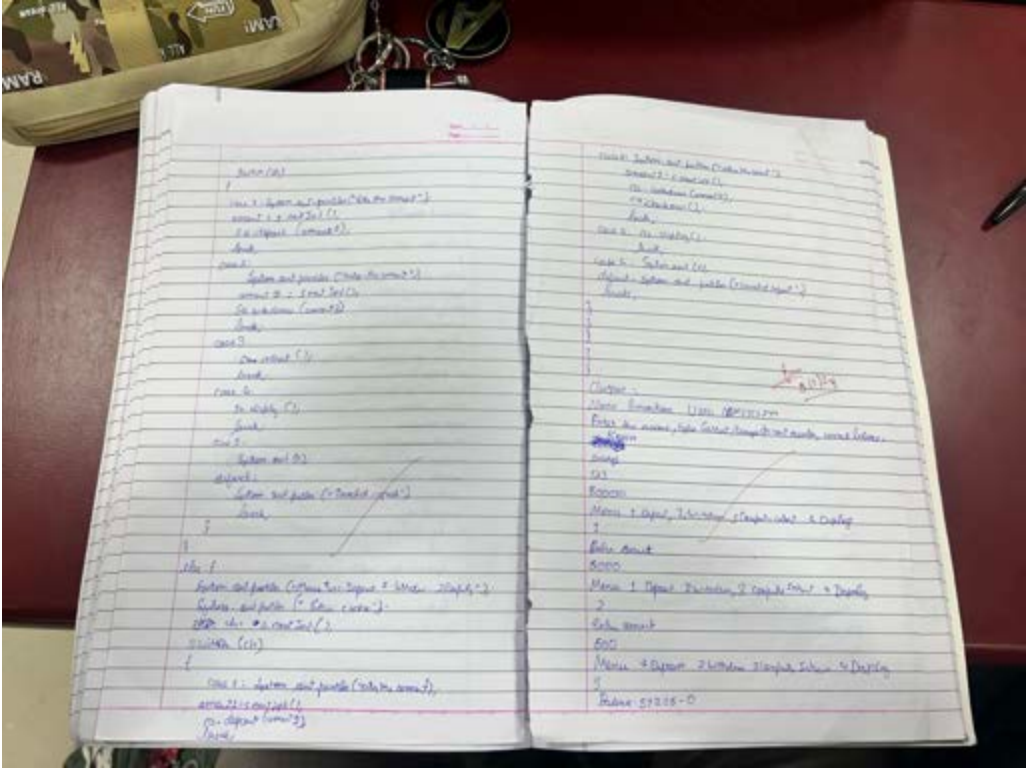
Q2) Develop Java Program to create class Student with members usn, name, four subjects and sum of marks. Include methods to accept and display details and a method to calculate SGPA of Student.

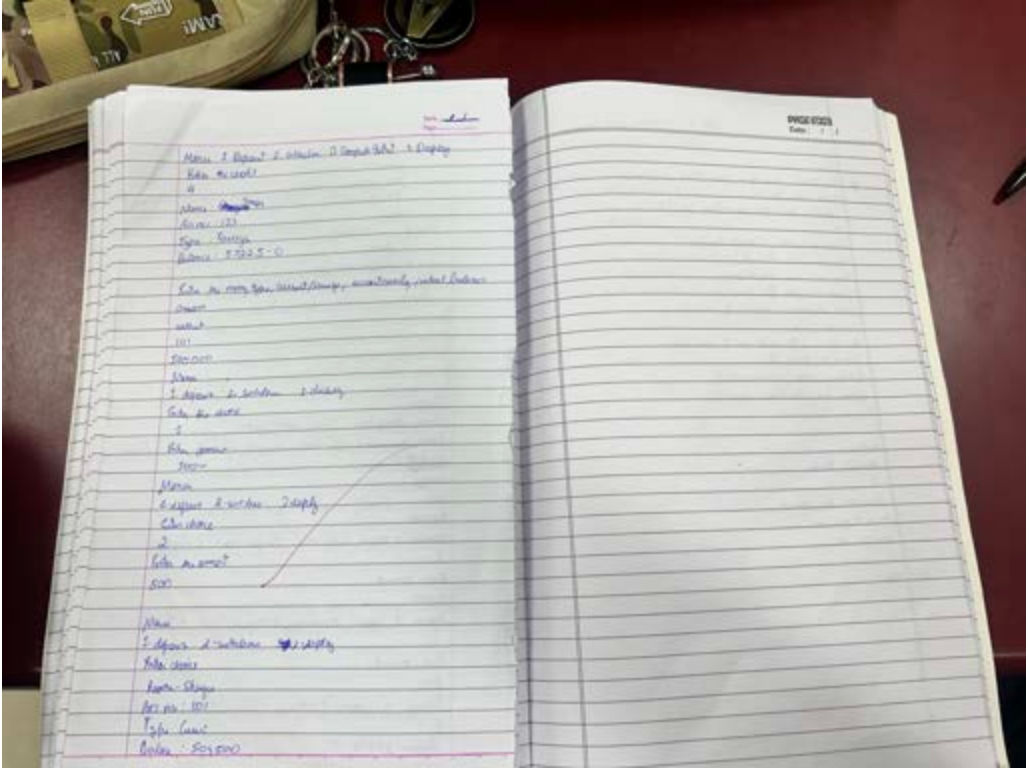
```
import java.util.*;
class Subject {
    int subjectMarks;
    int credits;
    int grade;
}
class Student {
    Subject subject[];
    String name;
    String usn;
    double sgpa;
    Scanner sc;

    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);
    }
}
```

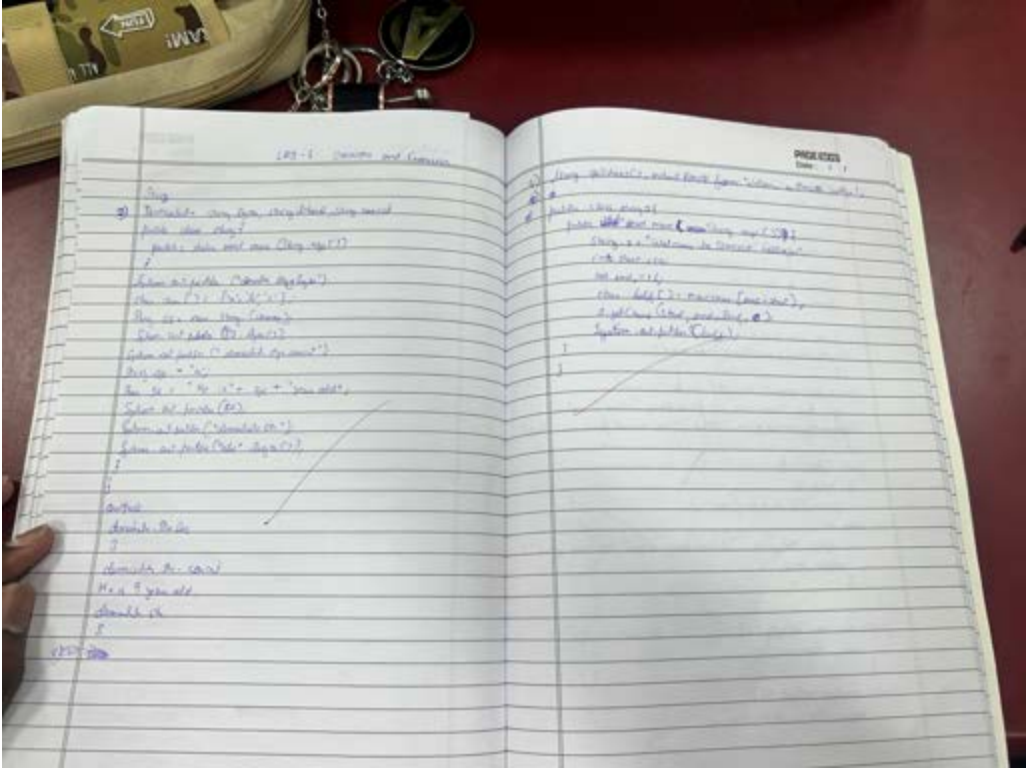


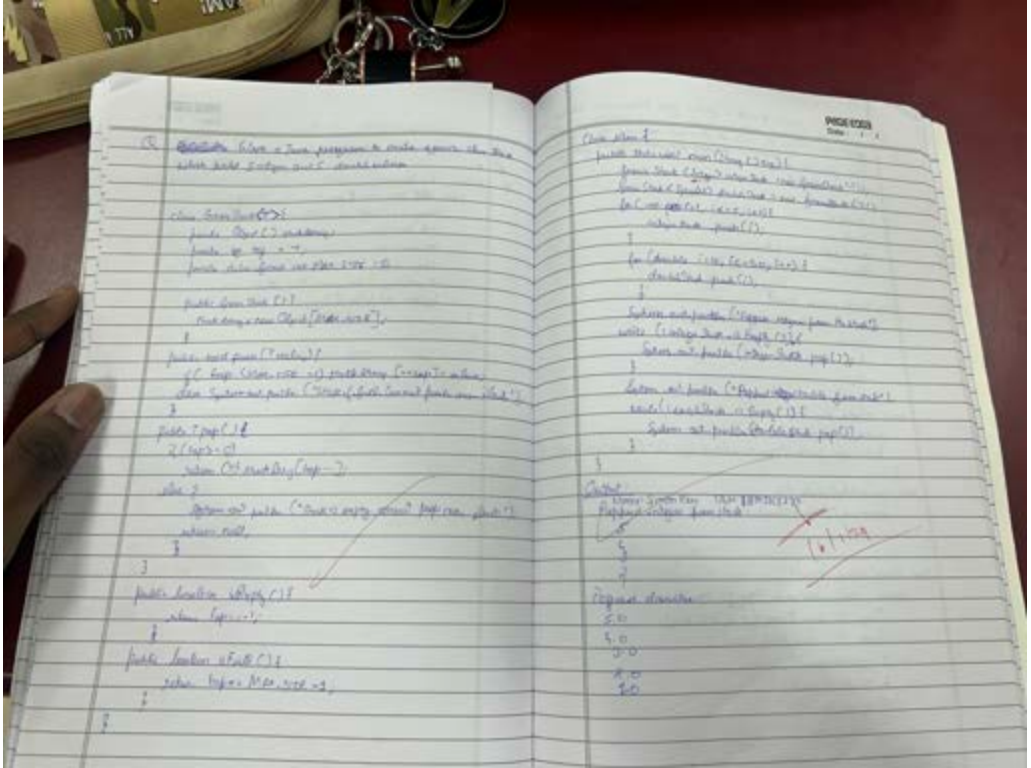


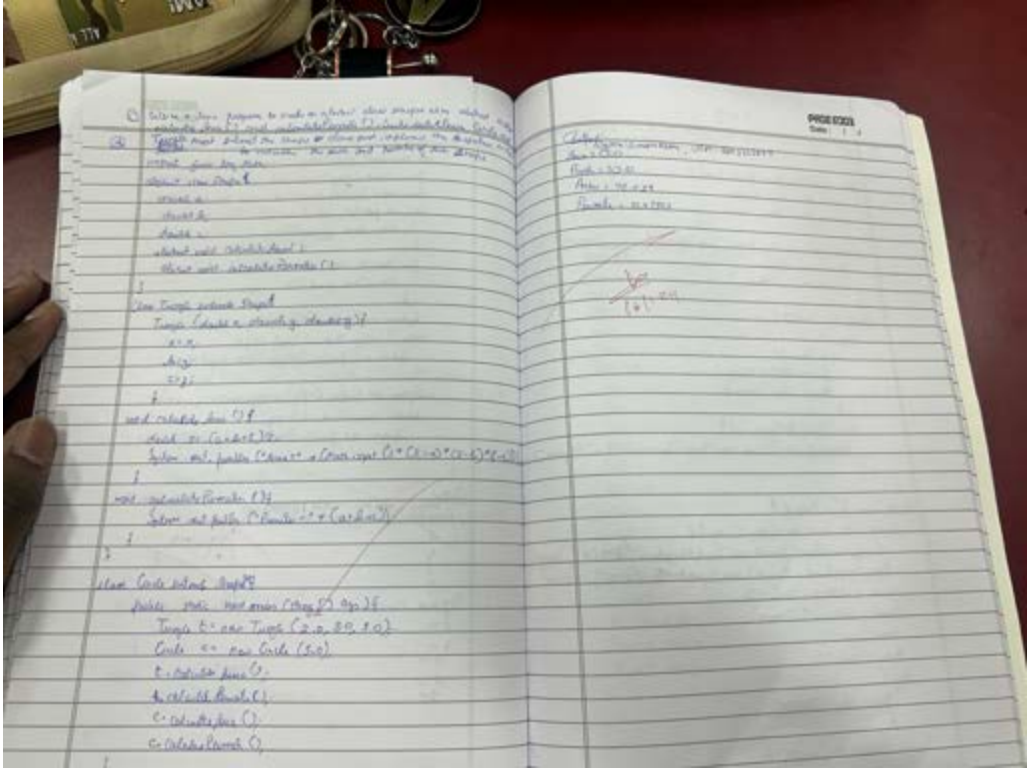








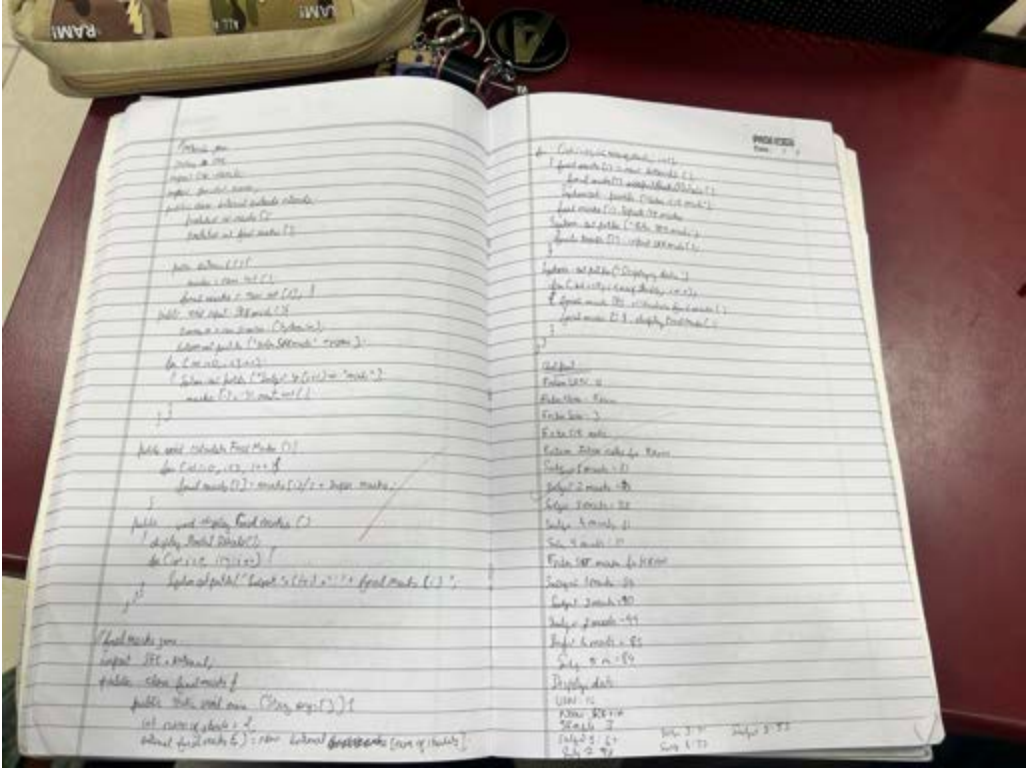




LAB-6 Packages

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 Ctrl+C to stop
Put: 0
Initiate Consumer
Producer waiting
Got: 0
Initiate Producer
Put: 1
Initiate Consumer
Consumed: 0
Producer waiting
Got: 1
Initiate Producer
Consumed: 1
Put: 2
Initiate Producer
Consumed: 1
Put: 2
Initiate Consumer
Producer waiting
Got: 2
Initiate Consumer
Consumed: 3
Put: 3
Initiate Consumer
Producer waiting
Got: 3
Initiate Producer
Consumer [illegible]
[illegible]
```

LAB-7

Page no.
Date: / /

Q. [illegible]

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

class SwingDemo {
  SwingDemo() {
    JFrame jfrm = new JFrame("Division App");
    jfrm.setSize(275, 100);
    jfrm.setLayout(new FlowLayout());
    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JLabel jlab = new JLabel("Enter the divider and divid");
    JTextField ajtf = new JTextField(8);
    JTextField bjtf = new JTextField(8);
    JButton button = new JButton("Calculate");
    JLabel err = new JLabel();
    JLabel alab = new JLabel();
    JLabel blab = new JLabel();
    JLabel anslab = new JLabel();
    jfrm.add(err);
    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 ae) {
      System.out.println("Action event from text field"); };
```

```
public void main() {
public void main(String[]
public class Hello World Class Main(String[]
    public void main(String[] args) {
        try {
            System.out.println("Hello World");
            System.out.println("Hello World");
            int i = 0;
            while (i < 5) {
                System.out.println("Hello World");
                i++;
            }
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: " + e);
            System.out.println("Hello World");
            System.out.println("Hello World");
            System.out.println("Hello World");
            System.out.println("Hello World");
        } catch (Exception e) {
            System.out.println("Error: " + e);
            System.out.println("Hello World");
            System.out.println("Hello World");
            System.out.println("Hello World");
            System.out.println("Hello World");
        }
    }
    System.out.println("Hello World");
}
public class Hello World {
    public static void main(String[] args) {
        System.out.println("Hello World");
        try {
            int i = 0;
            while (i < 5) {
                System.out.println("Hello World");
                i++;
            }
        } catch (Exception e) {
            System.out.println("Error: " + e);
        }
    }
}
```

Database Class

- Create - To create a new database, you can use the CREATE DATABASE statement. For example, to create a database named "mydb", you would execute the following command:

```
CREATE DATABASE mydb;
```

- Open (String path) - This method opens the database at the specified path.

- Close () - This method closes the database connection. It is important to close the database connection when you are finished with it to free up resources.

- Delete (String path) - This method deletes the database at the specified path.

- Update - To update data in a table, you can use the UPDATE statement. For example, to update the value of the "name" column in the "users" table where the "id" column is equal to 1, you would execute the following command:

```
UPDATE users SET name = 'John' WHERE id = 1;
```

- Insert (String) - This method inserts a new record into the database.

- Query - To query data from a table, you can use the SELECT statement. For example, to select all rows from the "users" table, you would execute the following command:

```
SELECT * FROM users;
```

- getTables () - This method returns a list of all the tables in the database.

- getTable (String name) - This method returns a Table object for the table with the specified name.