Tab 1

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

Aksha S (1BM23CS019)

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 to 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 (23CS3PCOOJ)" carried out by **Aksha S (1BM23CS019),** 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. The Lab report has been approved as it satisfies the academic requirements in respect of an Object Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

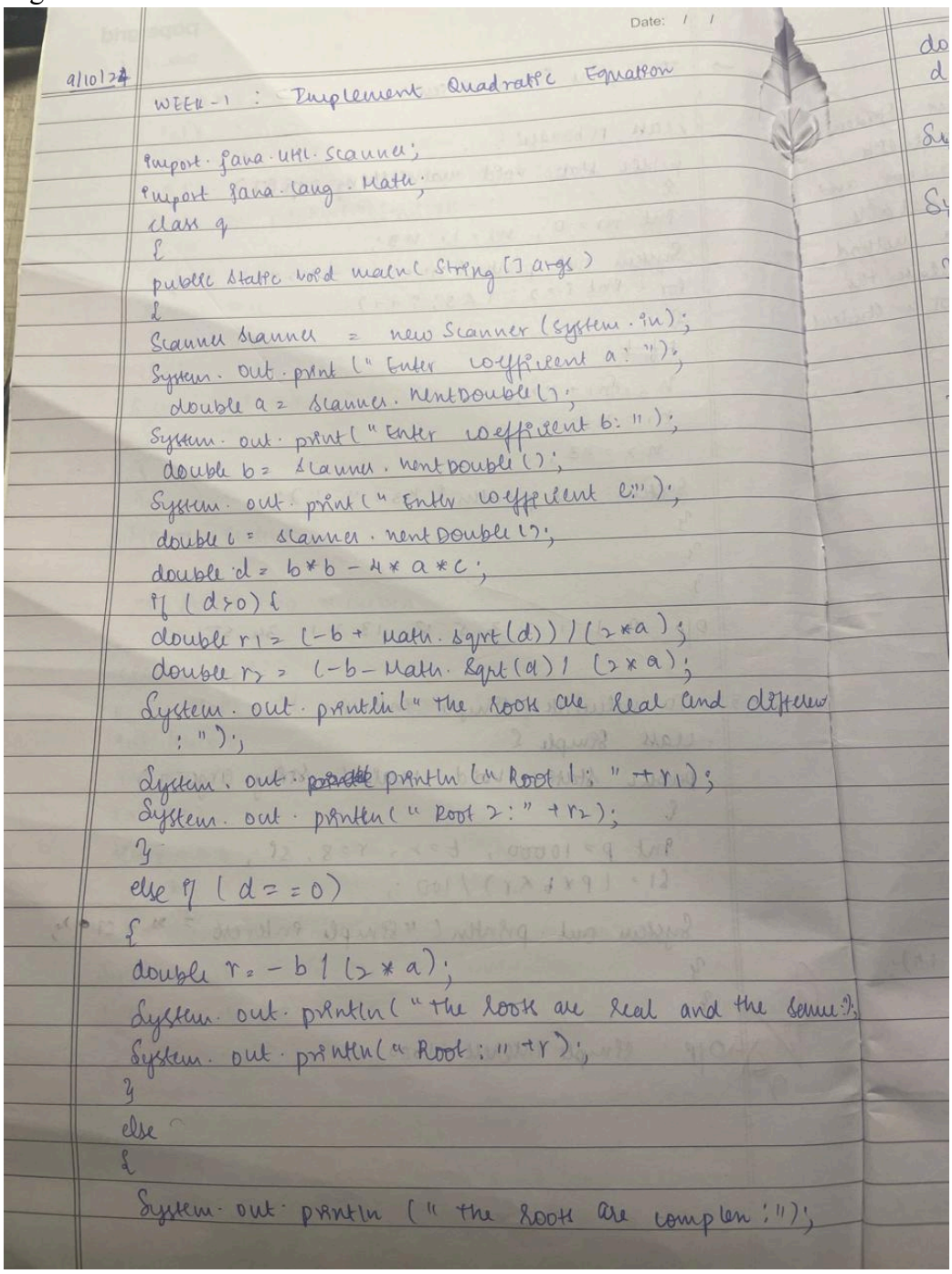
Lab faculty Incharge Name Assistant Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

Index

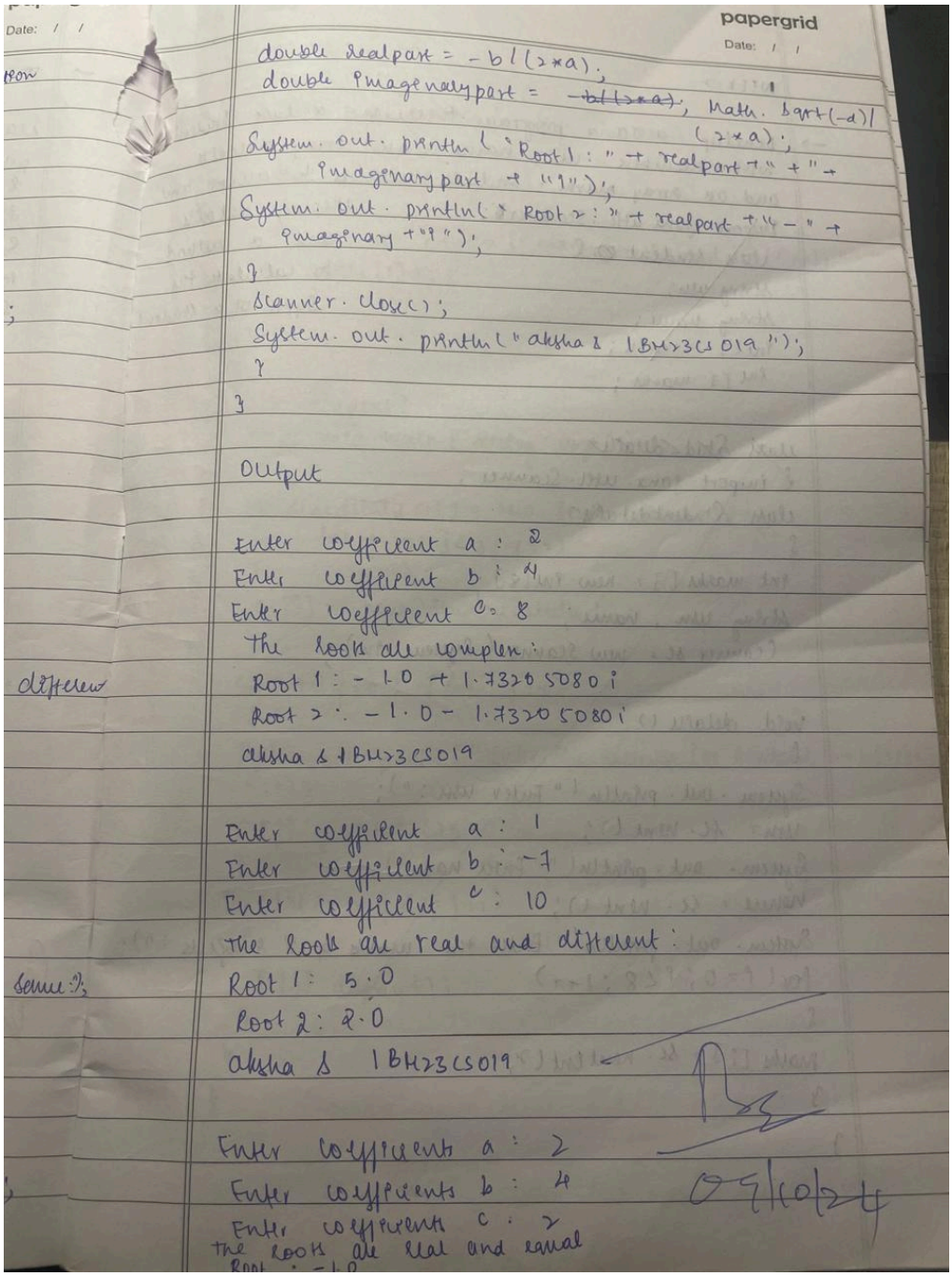
Github Link: https://github.com/akshas-25/OOJ-LAB

Program 1
Implement Quadratic Equation

Algorithm:



```
Date: / /
9/10/24
WEEK-1 : Implement Quadratic Equation

import java.util.Scanner;
import java.lang.Math;
class q
{
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 d = b*b - 4*a*c;
if (d>0){
double r1 = (-b + Math.sqrt(d)) / (2*a);
double r2 = (-b - Math.sqrt(d)) / (2*a);
System.out.println("the roots are real and differen
: ");
System.out.println("Root 1: " +r1);
System.out.println("Root 2:" +r2);
}
else if (d==0)
{
double r = -b / (2*a);
System.out.println("the roots are real and the same:");
System.out.println("Root : " +r);
}
else
{
System.out.println("the Roots are complex:");
```



```
double realpart = -b/(2*a);
double imagenarypart = -b/(2*a), Math.sqrt(-d)/
                                  (2*a);
System.out.println("Root1 : " + realpart + " + " +
    imagenarypart + "i");
System.out.println("Root2 : " + realpart + " - " +
    imagenary + "i");
}
Scanner.close();
System.out.println("akshaS 1BM23CS019");
}
}
```

Output

```
Enter coefficient a : 2
Enter coefficient b : 4
Enter coefficient c: 8
The roots are complex:
Root 1 : -1.0 + 1.7320508081
Root 2 : -1.0 - 1.7320508081
akshaS 1BM23CS019

Enter coefficient a : 1
Enter coefficient b : -7
Enter coefficient c : 10
The roots are real and different:
Root 1 : 5.0
Root 2 : 2.0
akshaS 1BM23CS019

Enter coefficients a : 2
Enter coefficients b : 4
Enter coefficients c : 2
The roots are real and equal
```

```
Enter coefficient a: 2
Enter coefficient b: 4
Enter coefficient c: 8
The roots are complex:
Root 1: -1.0 + 1.7320508075688772i
Root 2: -1.0 - 1.7320508075688772i
aksha s 1BM23CS019

C:\Users\Admin\Desktop>javac q.java

C:\Users\Admin\Desktop>java q
Enter coefficient a: 1
Enter coefficient b: -7
Enter coefficient c: 10
The roots are real and different:
Root 1: 5.0
Root 2: 2.0
aksha s 1BM23CS019

C:\Users\Admin\Desktop>javac q.java

C:\Users\Admin\Desktop>java q
Enter coefficient a: 2
Enter coefficient b: 4
Enter coefficient c: 2
The roots are real and the same:
Root: -1.0
aksha s 1BM23CS019
```

Code:

```
import java.util.Scanner;
import java.lang.Math;
 class q {
    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 d = b * b - 4 * a * c;

        if (d > 0) {
            double r1 = (-b + Math.sqrt(d)) / (2 * a);
```

```
            double r2 = (-b - Math.sqrt(d)) / (2 * a);
            System.out.println("The roots are real and different:");
            System.out.println("Root 1: " + r1);
            System.out.println("Root 2: " + r2);
        } else if (d == 0) {
            double r = -b / (2 * a);
            System.out.println("The roots are real and the same:");
            System.out.println("Root: " + r);
        } else {
            System.out.println("The roots are complex:");
            double realPart = -b / (2 * a);
            double imaginaryPart = Math.sqrt(-d) / (2 * a);
            System.out.println("Root 1: " + realPart + " + " + imaginaryPart + "i");
            System.out.println("Root 2: " + realPart + " - " + imaginaryPart + "i");
        }

        scanner.close();
        System.out.println("aksha s 1BM23CS019");
    }
}
```

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

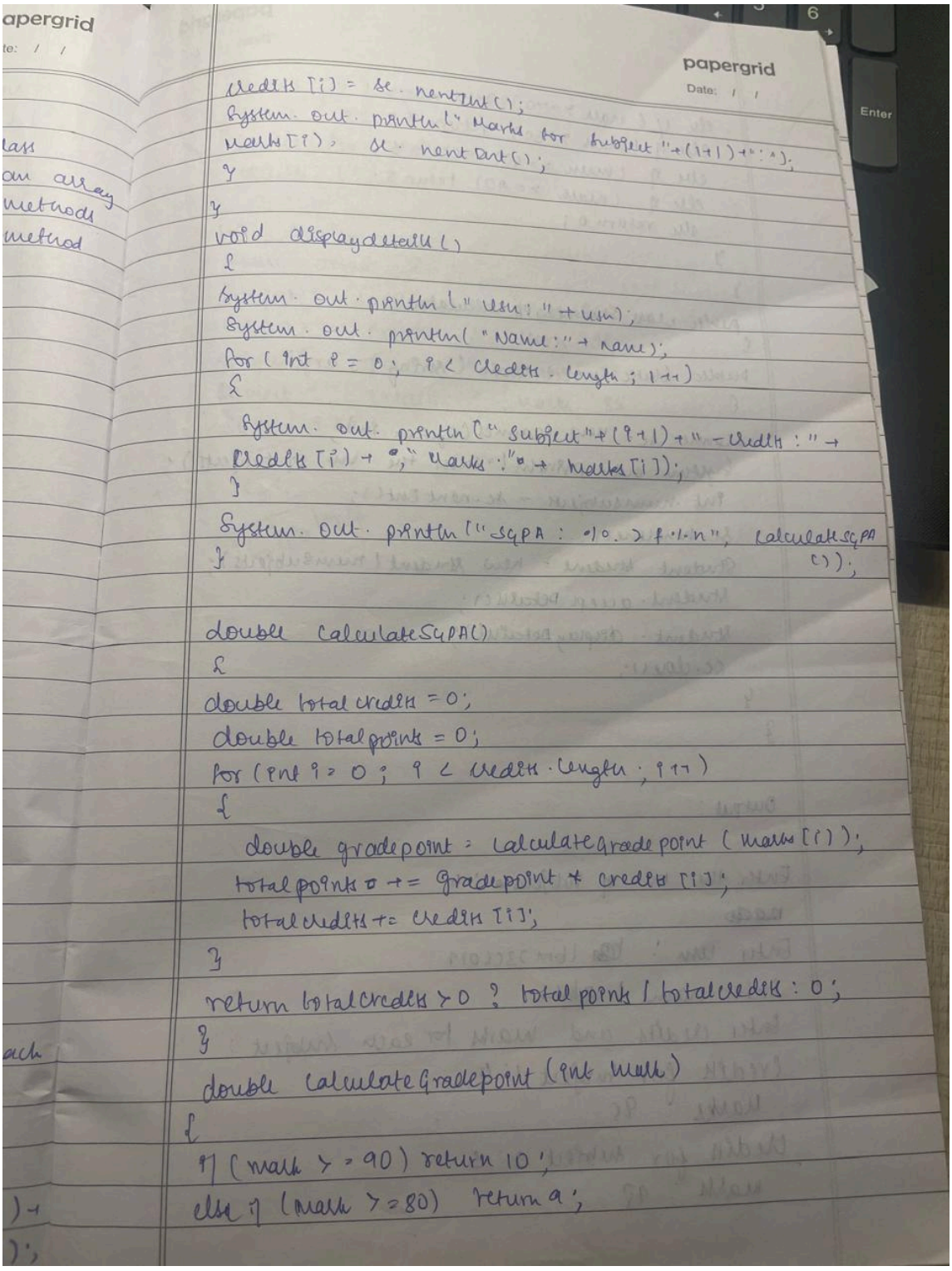
Algorithm:

7/10/24

WEEK 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 the SGPA of a student.

```
import java.util.Scanner;

class Student {
    String usn;
    String name;
    int[] credits;
    int[] marks;

    public Student(int numSubjects)
    {
    credits = new int[numSubjects];
    marks = new int[numSubjects];
    }
    void acceptDetails()
    {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter USN");
    usn = sc.nextLine();
    System.out.println("Enter Name:");
    name = sc.nextLine();
    System.out.println("Enter credits and marks for each subject");
    for (int i = 0; i < credits.length; i++)
    {
    System.out.println("Credits for subject " + (i+1) + ":");
```

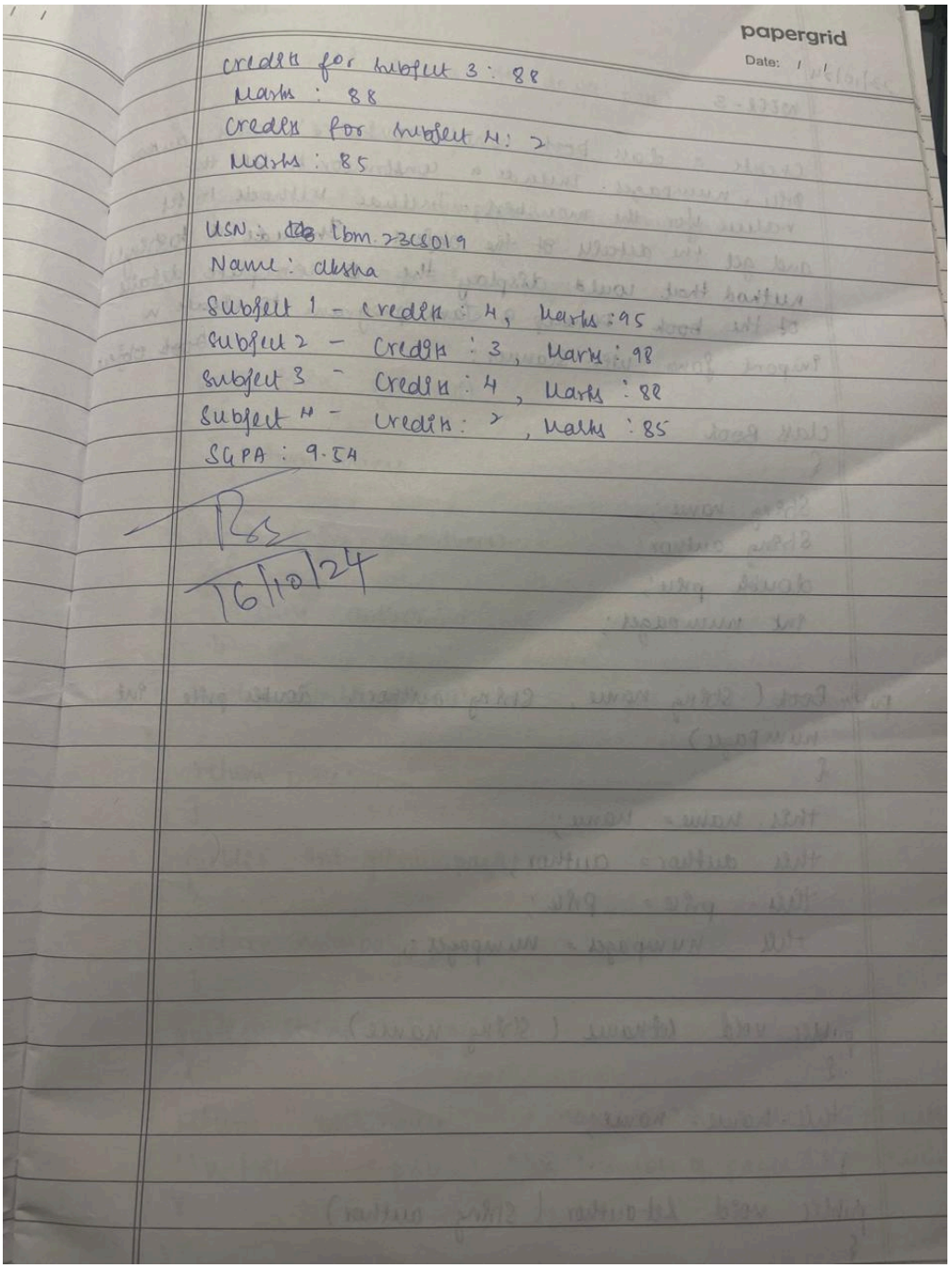


```
credits[i] = sc.nextInt();
System.out.println("Marks for subject "+(i+1)+": ");
marks[i] = sc.nextInt();
}
}
void displayDetails()
{
System.out.println("usn: "+usn);
System.out.println("Name: "+name);
for(int i=0; i<credits.length; i++)
{
System.out.println("subject "+(i+1)+" - Credits: "+
credits[i]+", Marks: "+marks[i]);
}
System.out.println("SGPA: %.2f\n", calculateSGPA
());
}

double calculateSGPA()
{
double totalCredits=0;
double totalpoints=0;
for(int i=0; i<credits.length; i++)
{
double gradepoint = calculategradepoint(marks[i]);
totalpoints += gradepoint * credits[i];
totalCredits += credits[i];
}
return totalCredits>0 ? totalpoints/totalCredits : 0;
}
double calculateGradepoint(int mark)
{
if(mark >= 90) return 10;
else if(mark >= 80) return 9;
```

```
        else if (mark >=70) return 8;
        else if (mark >=60) return 7;
        else if (mark >=50) return 6;
        else if (mark >=40) return 5;
        else return 0;
    }
}
public class StudentSGPAcalculator
{
public static void main (String [] args)
 {
 Scanner sc = new Scanner(System.in);
 System.out.println("Enter the no of subjects");
 int numsubjects = sc.nextInt();
 sc.nextLine();
 Student student = new Student(numSubjects);
 student.acceptDetails();
 student.displayDetails();
 sc.close();
}
}
```

Output:

```
Enter the no of subjects : 4
Enter usn : 1bm23cs019
Enter name : alisha
Enter credits and marks for each subject :
Credits for subject 1: 4
  Marks : 95
Credits for subject 2: 3
  Marks : 98
```


papergrid
Date: / /
credits for subject 3 : 88
Marks : 88
Credits for subject 4 : 2
Marks : 85
USN : 1bm23cs019
Name : akshra
Subject 1 - credits : 4, Marks : 95
Subject 2 - Credits : 3, Marks : 98
Subject 3 - Credits : 4, Marks : 88
Subject 4 - Credits : 2, Marks : 85
SGPA : 9.54
16/10/24

```
Enter the number of subjects: 4
Enter USN: 1bm23cs019
Enter Name: aksha
Enter credits and marks for each subj
Credits for subject 1: 4
Marks for subject 1: 95
Credits for subject 2: 3
Marks for subject 2: 98
Credits for subject 3: 4
Marks for subject 3: 88
Credits for subject 4: 2
Marks for subject 4: 85
USN: 1bm23cs019
Name: aksha
Subject 1 - Credits: 4, Marks: 95
Subject 2 - Credits: 3, Marks: 98
Subject 3 - Credits: 4, Marks: 88
Subject 4 - Credits: 2, Marks: 85
SGPA: 9.54
```

Code:

```
import java.util.Scanner;

class Student {
    String usn;
    String name;
    int[] credits;
    int[] marks;

        public Student(int numSubjects) {
        credits = new int[numSubjects];
        marks = new int[numSubjects];
    }

      void acceptDetails() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter USN: ");
        usn = sc.nextLine();  // Using nextLine to capture full input
        System.out.print("Enter Name: ");
        name = sc.nextLine(); // Using nextLine to capture full input

        System.out.println("Enter credits and marks for each subject:");
        for (int i = 0; i < credits.length; i++) {
            System.out.print("Credits for subject " + (i + 1) + ": ");
            credits[i] = sc.nextInt();
```

```
            System.out.print("Marks for subject " + (i + 1) + ": ");
            marks[i] = sc.nextInt();
        }
    }

        void displayDetails() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        for (int i = 0; i < credits.length; i++) {
            System.out.println("Subject " + (i + 1) + " - Credits: " + credits[i]
                                        + ", Marks: " + marks[i]);
        }
        System.out.printf("SGPA: %.2f%n", calculateSGPA());
    }


    double calculateSGPA() {
        double totalCredits = 0;
        double totalPoints = 0;

        for (int i = 0; i < credits.length; i++) {
            double gradePoint = calculateGradePoint(marks[i]);
            totalPoints += gradePoint * credits[i];
            totalCredits += credits[i];
        }

        return totalCredits > 0 ? totalPoints / totalCredits : 0;
    }

       double calculateGradePoint(int mark) {
        if (mark >= 90) return 10;
        else if (mark >= 80) return 9;
        else if (mark >= 70) return 8;
        else if (mark >= 60) return 7;
        else if (mark >= 50) return 6;
        else if (mark >= 40) return 5;
        else return 0; // Fail
    }
}

public class StudentSGPACalculator {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of subjects: ");
```

```
        int numSubjects = sc.nextInt();
        sc.nextLine();
        Student student = new Student(numSubjects);
        student.acceptDetails();
        student.displayDetails();

        sc.close();
    }
}
```

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.

Algorithm:

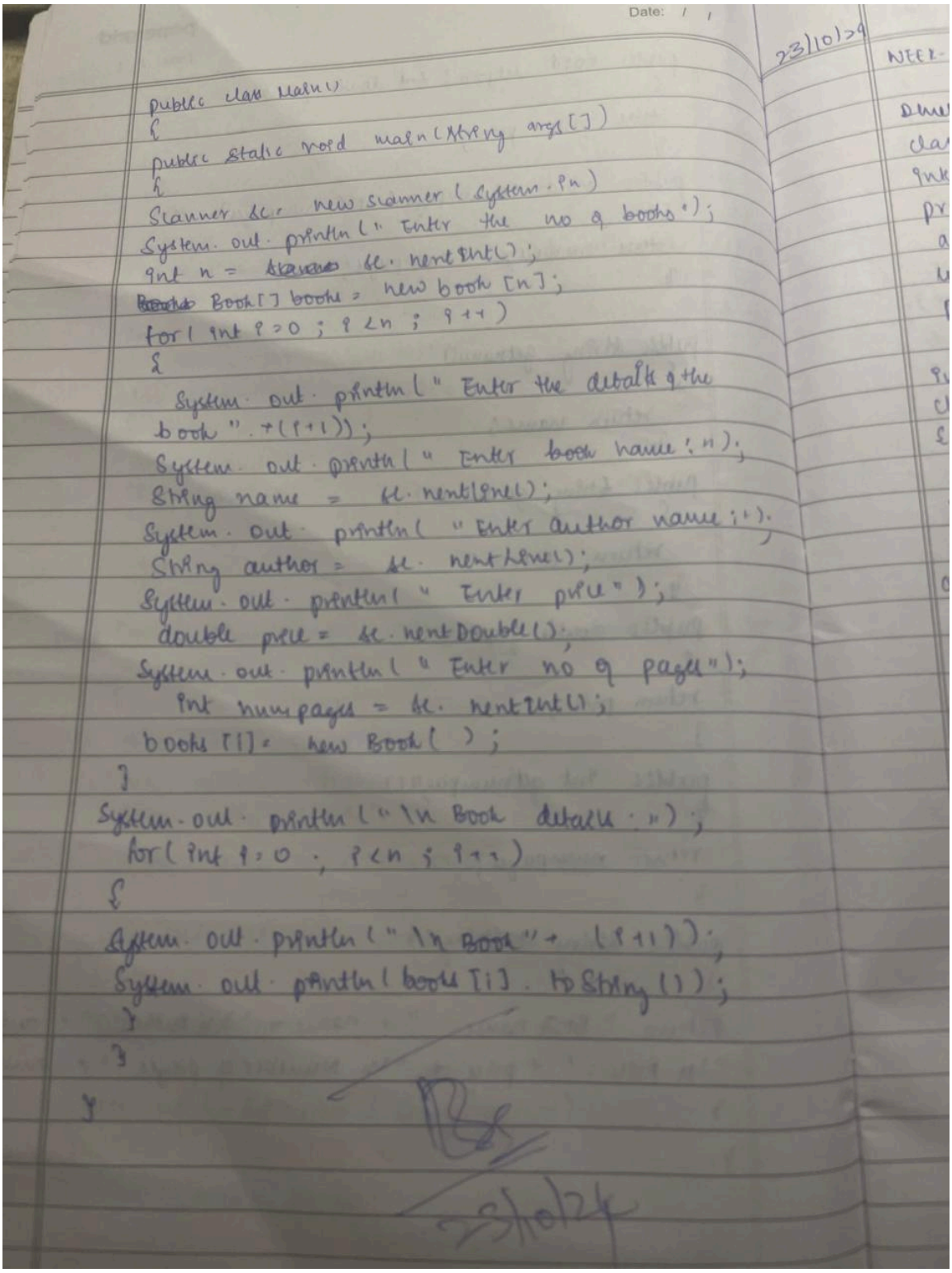
23/10/24

WEEK-3

Create a class Book which contains: name, author, price, numpages. 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 Book
{
 String name;
 String author;
 double price;
 int numpages;

public Book ( String name, String author, double price, int
    numpages)
 {
  this.name = name;
  this.author = author;
  this.price = price;
  this.numpages = numpages;

public void setname ( String name)
 {
  this.name = name;
 }
public void setauthor ( String author)
 {
  this.author = author;
 }
```

```
public void setPrice( int double price)
{
this price = price;
}
public void setnumpages ( int numpages)
{
this.numpages = numpages;
}

public String getname()
{
return name;
}
public String getauthor()
{
return author;
}
public double getprice()
{
return price;
}
public int getnumpages()
{
return numpages;
}
public String toString()
{
return "Book name : " + name + "\n Author :" + author +
"\n Price : " + price + "\n Number of pages :" + numpages;
}
}
```



```
public class Main()
{
public static void main(String args[])
{
Scanner sc = new scanner(System.in)
System.out.println("Enter the no of books");
int n = sc.nextInt();
Book[] books = new book[n];
for(int i=0; i<n; i++)
{
  System.out.println("Enter the details of the
book".+(i+1));
System.out.println("Enter book name:");
String name = sc.nextLine();
System.out.println("Enter author name:");
String author = sc.nextLine();
System.out.println("Enter price");
double price = sc.nextDouble();
System.out.println("Enter no of pages");
  int numpages = sc.nextInt();
books[i] = new Book();
}
System.out.println("\n Book details:");
 for(int i=0; i<n; i++)
{
System.out.println("\n Book"+(i+1));
System.out.println(books[i].toString());
}
}
}
```

```
Enter details for Book 1
Enter book name: aaa
Enter author name: abc
Enter price: 799
Enter number of pages: 900

Enter details for Book 2
Enter book name: bbb
Enter author name: efg
Enter price: 897
Enter number of pages: 500

Enter details for Book 3
Enter book name: ccc
Enter author name: fgh
Enter price: 899
Enter number of pages: 600

Book Details:

Book 1
Book Name: aaa
Author: abc
Price: 799.0
Number of Pages: 900

Book 2
Book Name: bbb
Author: efg
Price: 897.0
Number of Pages: 500

Book 3
Book Name: ccc
Author: fgh
Price: 899.0
Number of Pages: 600
```

Code:

```
import java.util.Scanner;

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


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

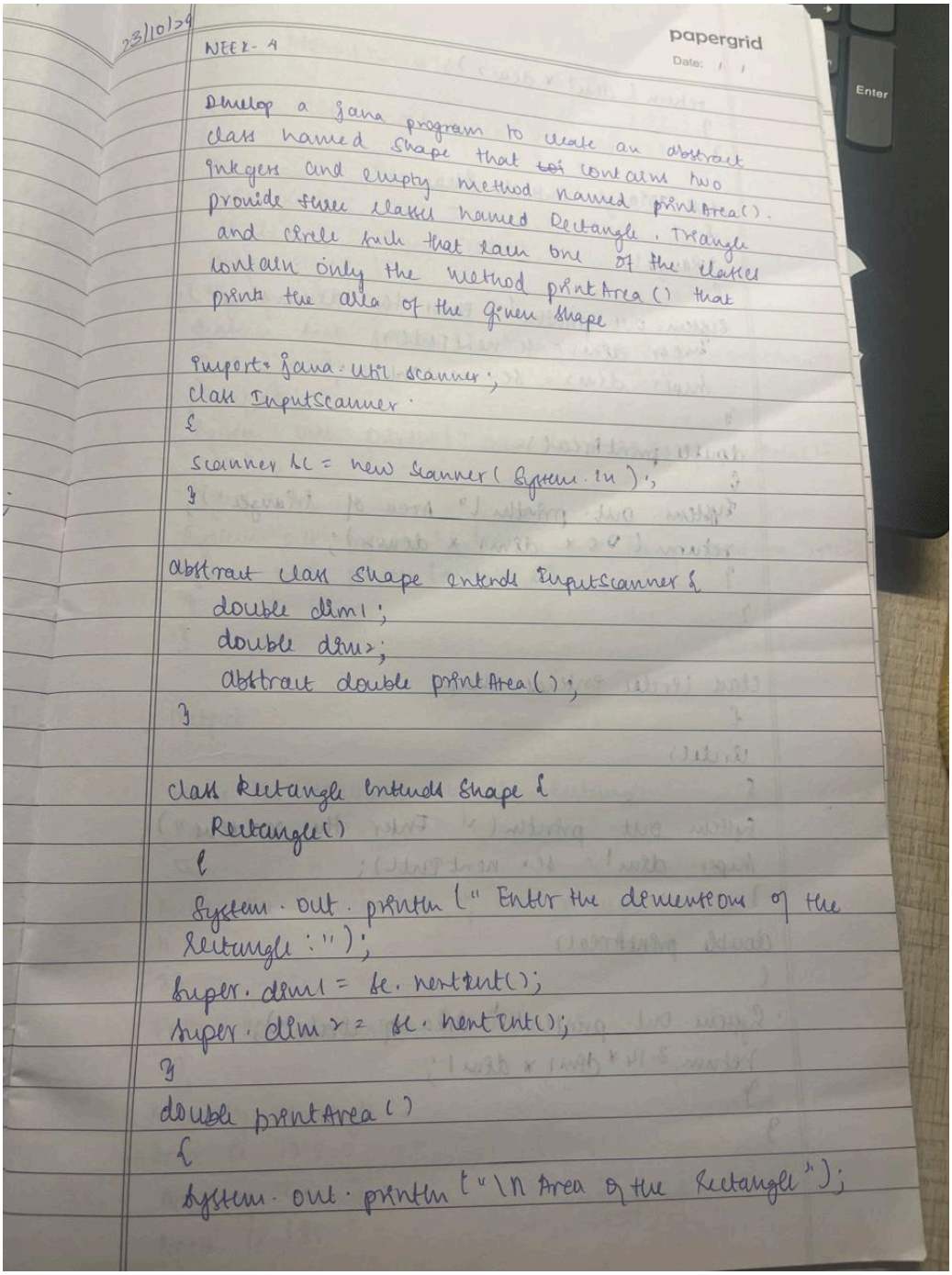
```
    this.author = author;
    this.price = price;
    this.numPages = numPages;
}
public void setName(String name) {
    this.name = name;
}

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

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

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


public String getName() {
    return name;
}

public String getAuthor() {
    return author;
}

public double getPrice() {
    return price;
}

public int getNumPages() {
    return numPages;
}



public String toString() {
    return "Book Name: " + name + "\nAuthor: " + author + "\nPrice: " +
                                  price + "\nNumber of Pages: " +
                                  numPages;
}
```

```
}

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

      Book[] books = new Book[n];

      for (int i = 0; i < n; i++) {
         System.out.println("\nEnter details for Book " + (i + 1));
         System.out.print("Enter book name: ");
         String name = sc.nextLine();
         System.out.print("Enter author name: ");
         String author = sc.nextLine();
         System.out.print("Enter price: ");
         double price = sc.nextDouble();
         System.out.print("Enter number of pages: ");
         int numPages = sc.nextInt();
         sc.nextLine();

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

      System.out.println("\nBook Details:");
      for (int i = 0; i < n; i++) {
         System.out.println("\nBook " + (i + 1));
         System.out.println(books[i].toString());
      }
   }}
```

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.

Algorithm:



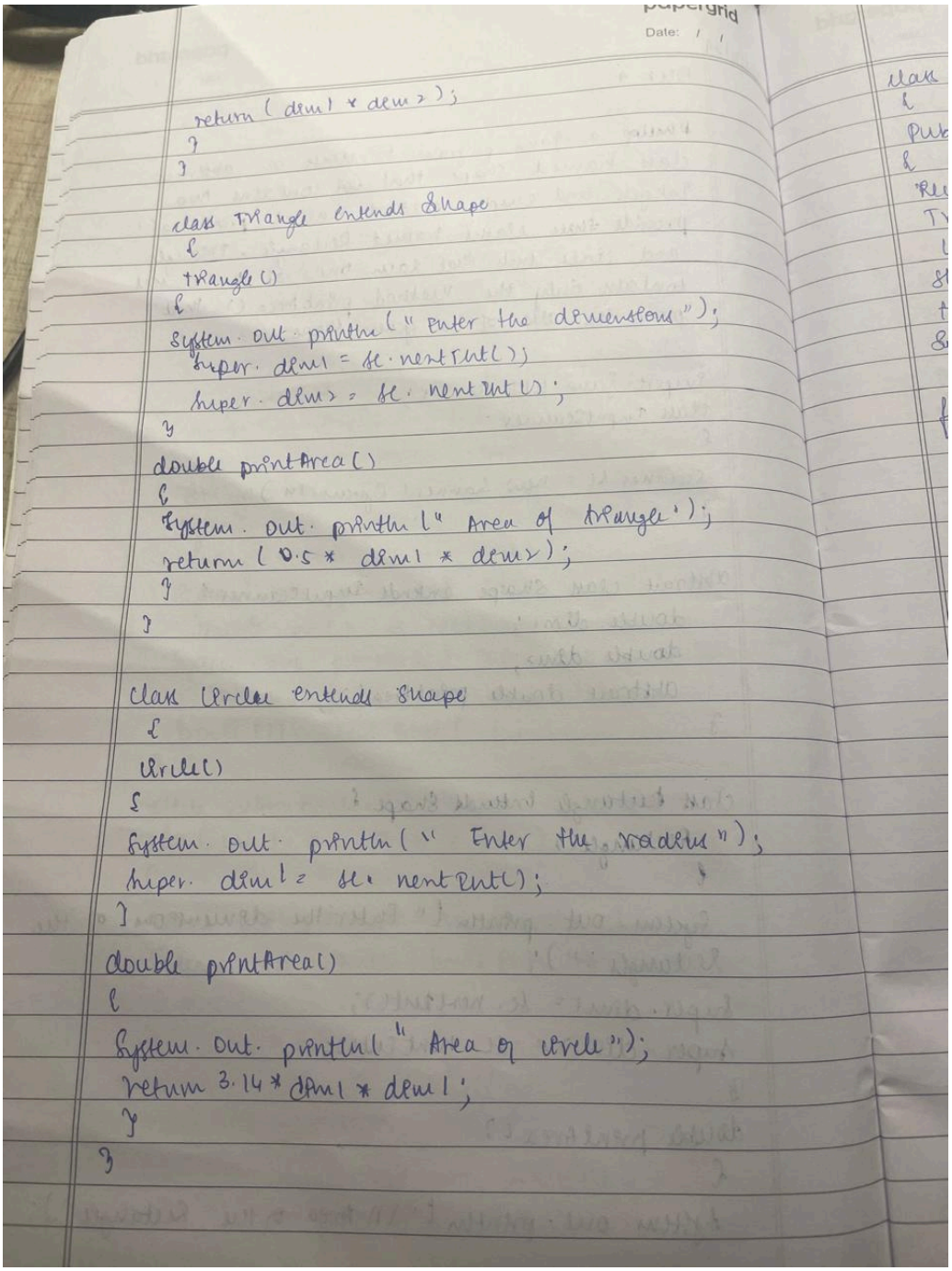
23/10/24

WEEK- 4

papergrid
Date: / /

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

```
import java.util.Scanner;
class InputScanner
{
Scanner sc = new Scanner(System.in);
}

abstract class Shape extends InputScanner {
    double dim1;
    double dim2;
    abstract double printArea();
}

class Rectangle extends Shape {
    Rectangle()
    {
    System.out.println("Enter the dementions of the
    Rectangle:");
super.dim1 = sc.nextInt();
super.dim2 = sc.nextInt();
}
double printArea()
{
System.out.println("\n Area of the Rectangle");
```



```
return (dim1 * dim2);
}
}

class Triangle extends Shape
{
Triangle()
{
System.out.println("Enter the dimensions");
super.dim1 = sc.nextInt();
super.dim2 = sc.nextInt();
}
double printArea()
{
System.out.println("Area of triangle");
return (0.5 * dim1 * dim2);
}
}

class Circle extends Shape
{
Circle()
{
System.out.println("Enter the radius");
super.dim1 = sc.nextInt();
}
double printArea()
{
System.out.println("Area of circle");
return 3.14 * dim1 * dim1;
}
}
```

```
class AbstractDemo
{
public static void main(String args[])
{
Rectangle r = new Rectangle();
Triangle t = new Triangle();
Circle c = new Circle();
Shape fegref;
fegref = r;
System.out.println("Area is:" + fegref.printArea()
+ "\n");
figref = t;
System.out.println("Area is:" + fegref.printArea()+
"\n");
fegref = c;
System.out.println("Area is:" + fegref.printArea()+"\n");
System.out.println("ahsha");
}
}
```

Output:

Enter the dementions of the Rectangle:
>>
55
Enter the dementions of the Triangle:
11
33
Enter the ~~diameter~~ radius
20
Area of rectangle:
Area is 1210.0
Area of Triangle:
Area is 181.5

23/10/24

```
Enter the dimensions of the Rectangle:
22
55
Enter the dimensions of the Triangle:
11
33
Enter the dimension (radius) of the Circle:
20

Area of rectangle:
Area is: 1210.0

Area of Triangle:
Area is: 181.5

Area of Circle:
Area is: 1256.0

aksha s
```

Code:

```
import java.util.Scanner;
class InputScanner{
   Scanner sc = new Scanner(System.in);
}

abstract class Shape extends InputScanner{
   double dim1;
   double dim2;
   abstract double printArea();
}
class Rectangle extends Shape{

   Rectangle(){
      System.out.println("Enter the dimensions of the Rectangle: ");
   super.dim1 = sc.nextInt();
   super.dim2 = sc.nextInt();
   }

   double printArea(){
      System.out.println("\nArea of rectangle: ");
      return(dim1 * dim2);
   }
}

class Triangle extends Shape{
```

```
    Triangle (){
    System.out.println("Enter the dimensions of the Triangle: ");
    super.dim1 = sc.nextInt();
    super.dim2 = sc.nextInt();
    }

    double printArea(){
        System.out.println("Area of Triangle: ");
        return 0.5 * dim1 * dim2;
    }
}

class Circle extends Shape{
    Circle (){
    System.out.println("Enter the dimension (radius) of the Circle: ");
    super.dim1 = sc.nextInt();
    }

    double printArea(){
        System.out.println("Area of Circle: ");
        return 3.14*dim1*dim1;
    }
}
class AbstractDemo{
    public static void main(String args[]){
        Rectangle r = new Rectangle();
        Triangle t = new Triangle();
        Circle c = new Circle ();

        Shape figref;

        figref = r;
        System.out.println("Area is: "+figref.printArea()+"\n");

        figref = t;
        System.out.println("Area is: "+ figref.printArea()+"\n");

        figref = c;
        System.out.println("Area is: "+figref.printArea()+"\n");
        System.out.println("aksha s");
    }

}
```

Program 5:

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.

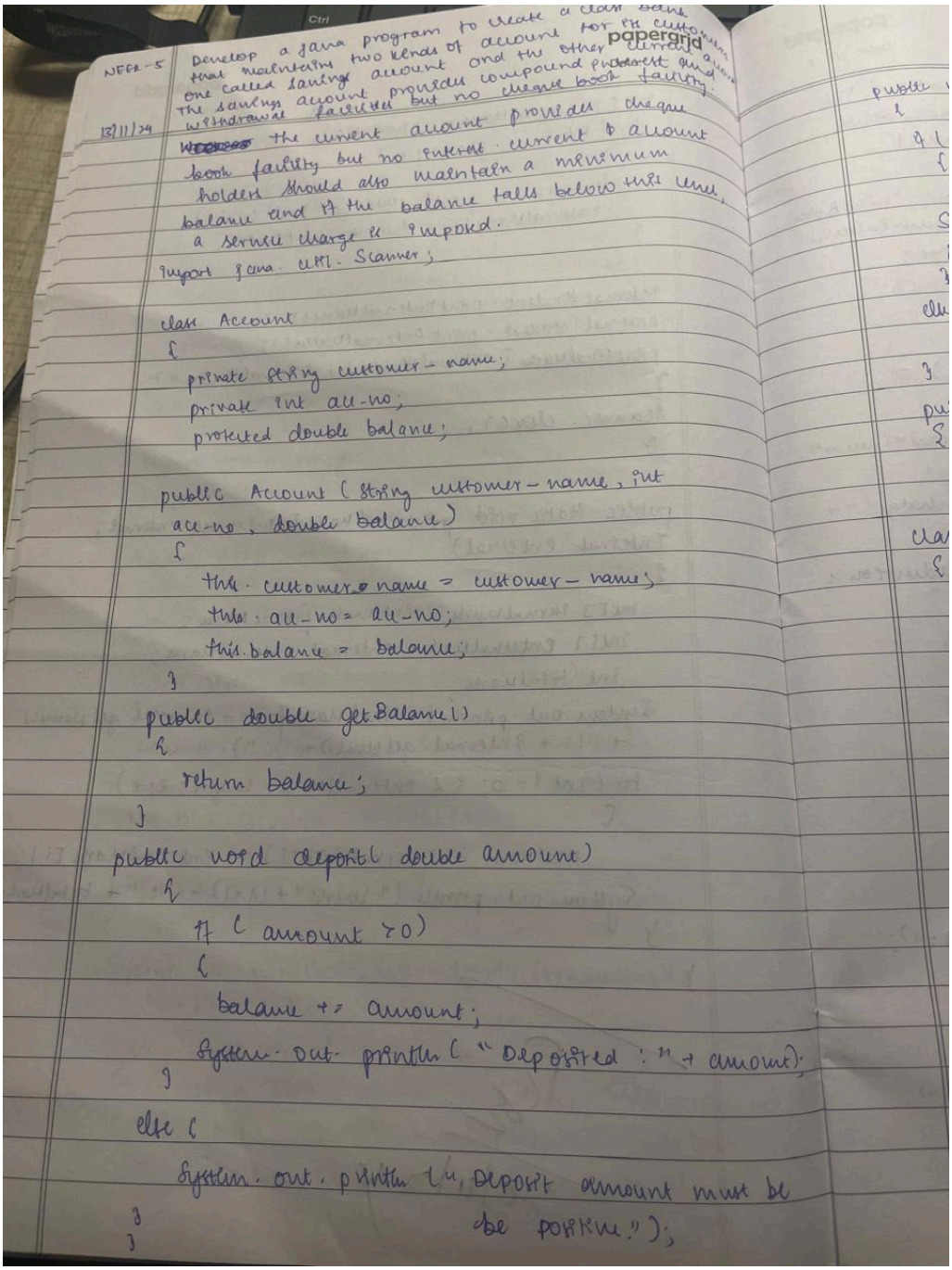
b)Display the balance.

c)Compute and deposit interest

d)Permit withdrawal and update the balance

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

Algorithm:

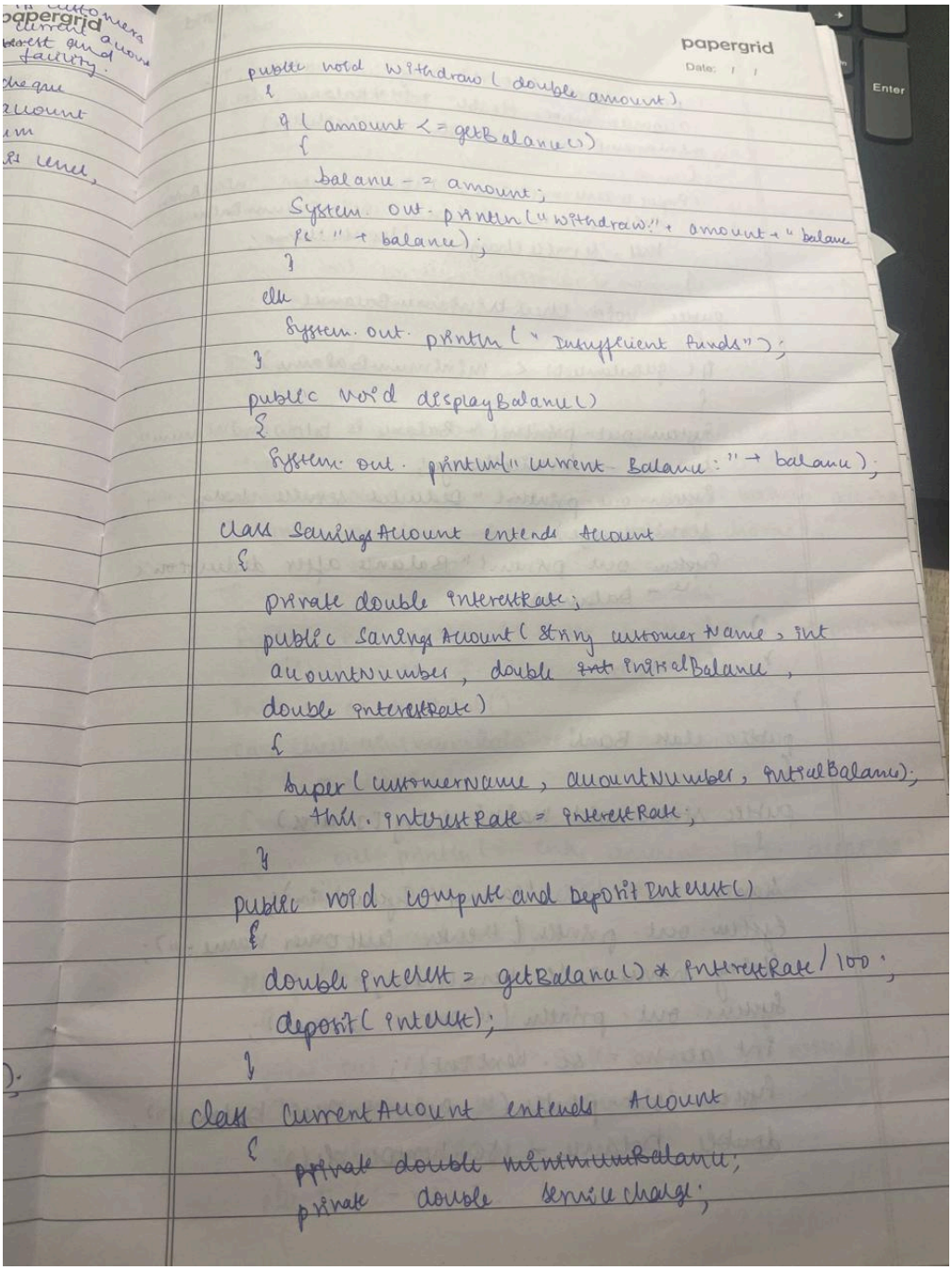


NEW-5

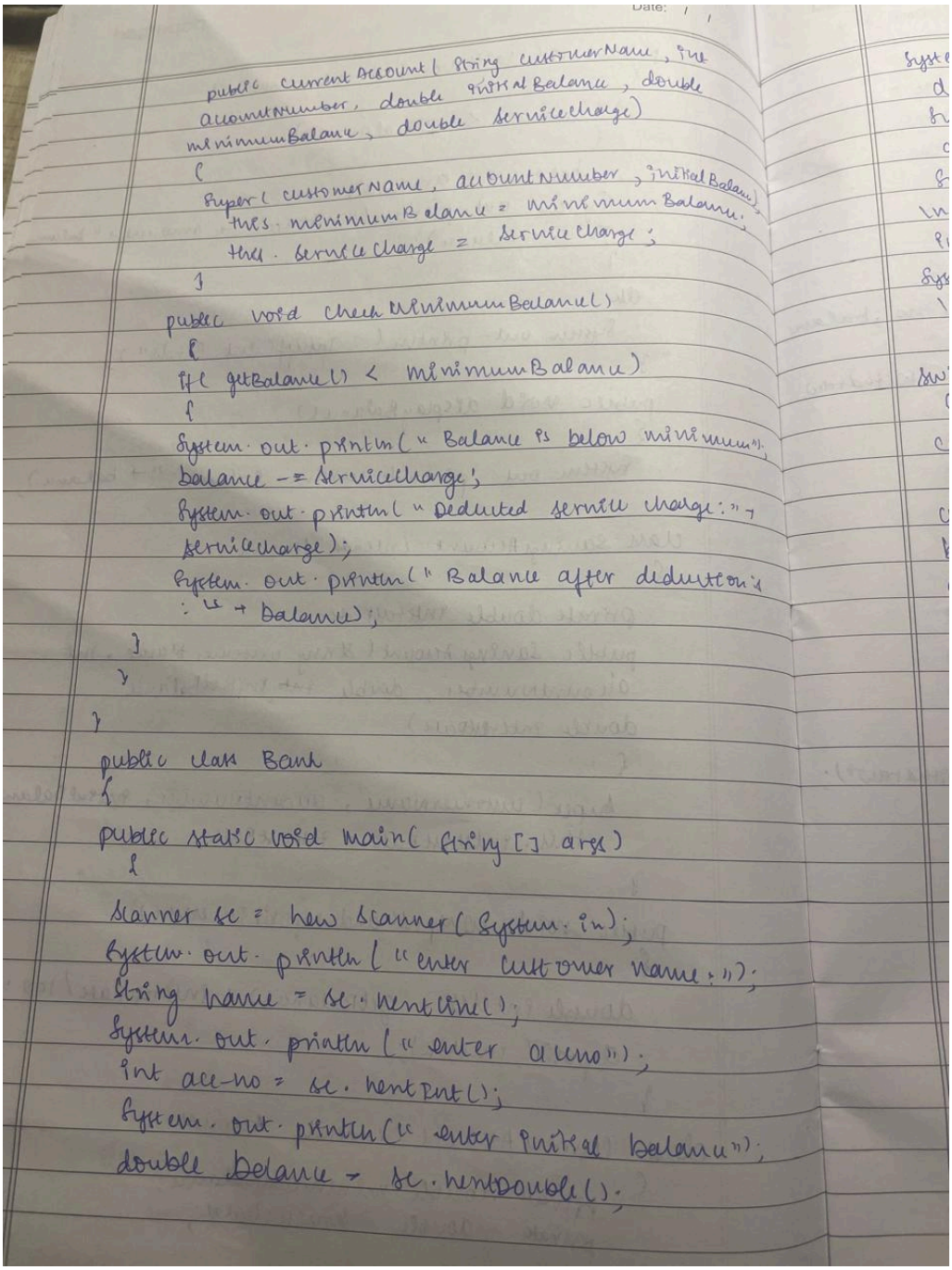
13/11/24

Develop a java program to create a class Bank that maintains two kinds of account for its customers, one called saving account and the other current account. The saving 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.

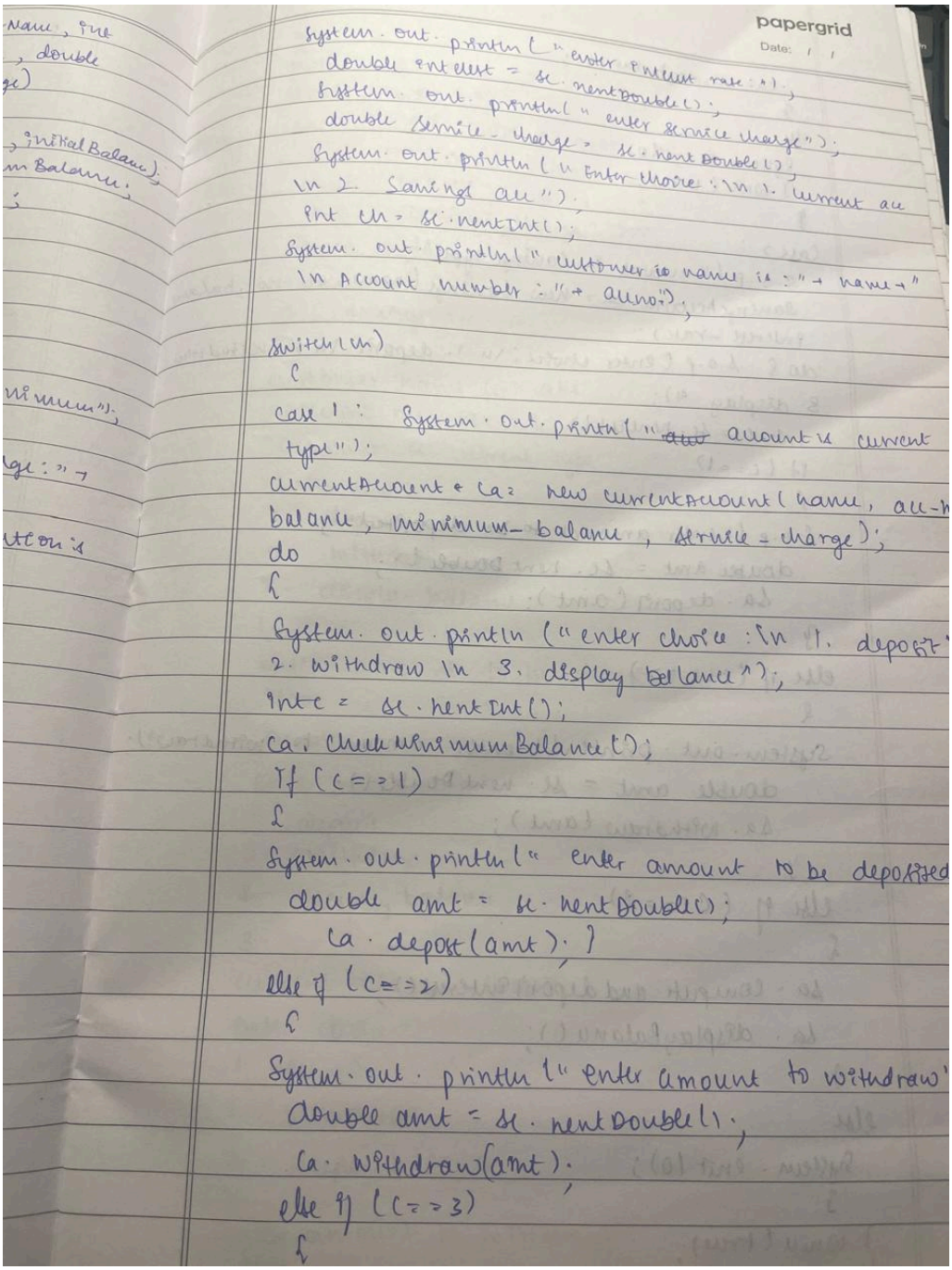
```
import java.util.Scanner;

class Account
{
  private String customer_name;
  private int ac_no;
  protected double balance;

  public Account (String customer_name, int
   ac_no, double balance)
   {
     this.customer_name = customer_name;
     this.ac_no = ac_no;
     this.balance = balance;
   }
  public double getBalance()
   {
     return balance;
  }
public void deposit(double amount)
   {
      if (amount > 0)
      {
        balance += amount;
      System.out.println("Deposited : " + amount);
   }
  else {
     System.out.println("Deposit amount must be
                          be positive");
  }
}
```



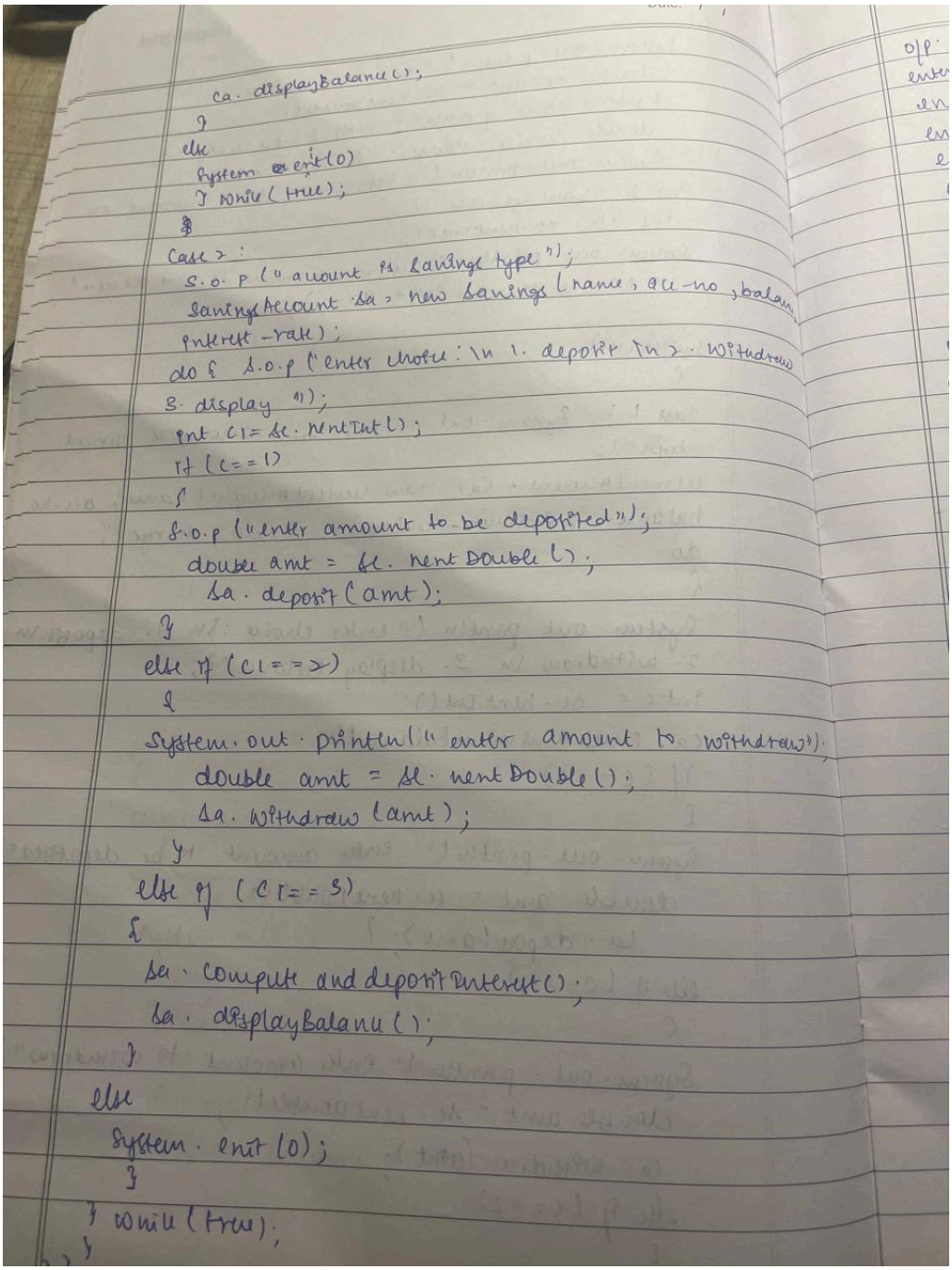
```
public void Withdraw ( double amount).
{
  if ( amount <= getBalance())
  {
    balance -= amount;
    System out. println ("Withdraw:" + amount + " balance
    is: " + balance);
  }
  else
    System. out. println ( " Insufficient funds");
}
public void displayBalance()
{
  System. out. println("Current Balance: " + balance);

class SavingsAccount extends Account
{
  private double interestRate;
  public SavingsAccount ( String customerName, int
  accountNumber, double initialBalance,
  double interestRate)
  {
    super ( customerName, accountNumber, initialBalance);
    this. interestRate = interestRate;
  }
public void compute and depositInterest()
{
  double interest = getBalance() * interestRate / 100;
  deposit ( interest);
}
class CurrentAccount extends Account
{
  private double minimumBalance;
  private double service charge;
```



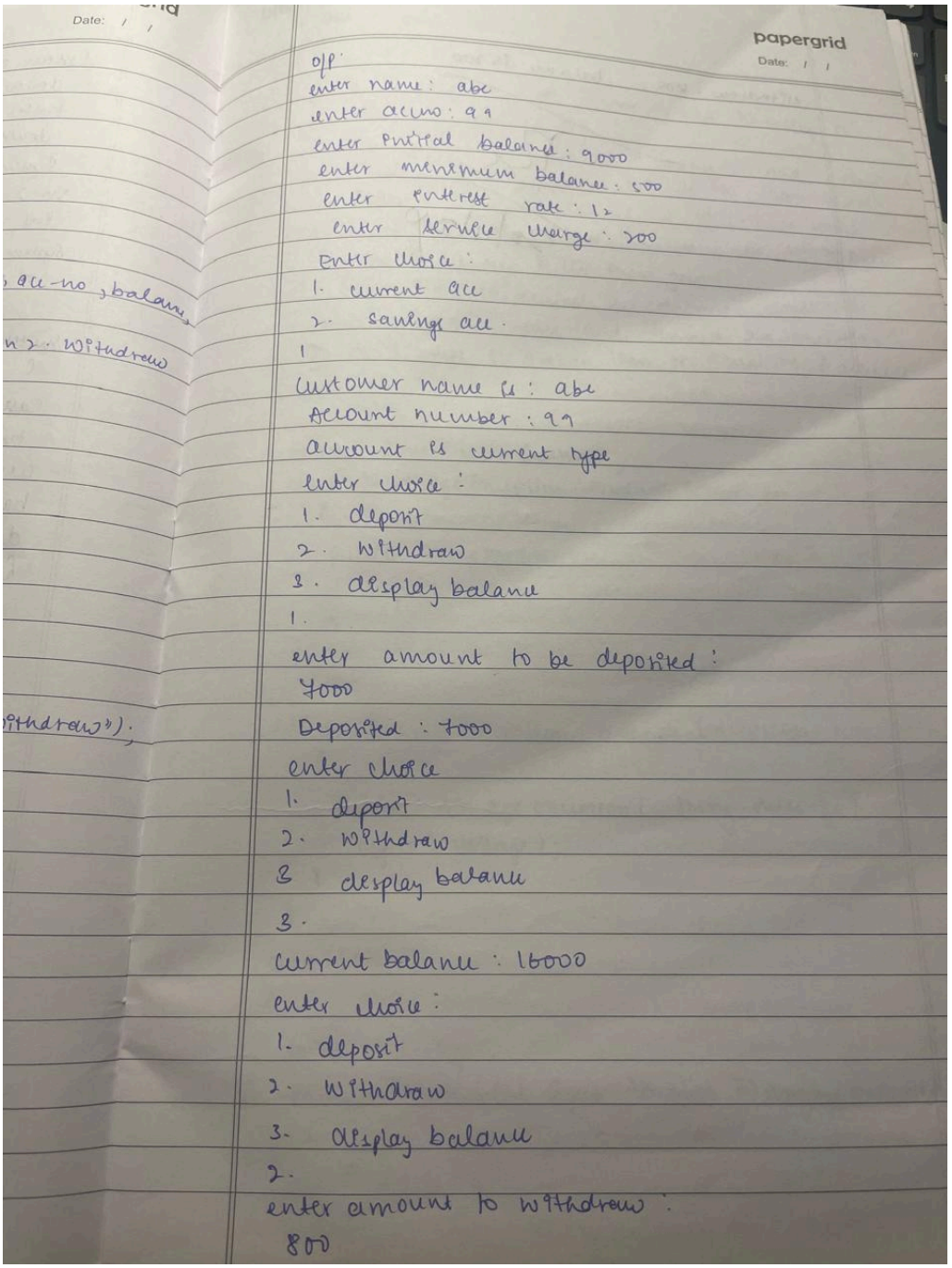
```
public CurrentAccount(String customerName, int
accountNumber, double initialBalance, double
minimumBalance, double serviceCharge)
{
super(customerName, accountNumber, initialBalance);
this.minimumBalance = minimumBalance;
this.serviceCharge = serviceCharge;
}
public void checkMinimumBalance()
{
if(getBalance() < minimumBalance)
{
System.out.println("Balance is below minimum");
balance -= serviceCharge;
System.out.println("Deducted service charge:"+
serviceCharge);
System.out.println("Balance after deduction
is:"+balance);
}
}
}
public class Bank
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
System.out.println("enter customer name:");
String name = sc.nextLine();
System.out.println("enter acno");
int acc-no = sc.nextInt();
System.out.println("enter initial balance");
double balance = sc.nextDouble();
```



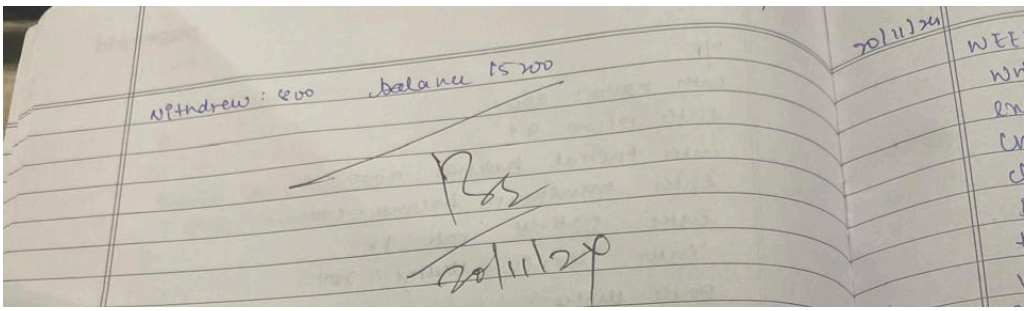
```
System.out.println("enter interest rate:");
double intRate = sc.nextDouble();
System.out.println("enter service charge");
double service_charge = sc.nextDouble();
System.out.println("Enter choice: \n 1. current ac
\n 2. Savings ac");
int ch = sc.nextInt();
System.out.println("customer is name is:"+name+"
\n Account number:"+acno);

switch(ch)
{
case 1: System.out.println("acount is current
type");
currentAccount ca= new currentAccount(name, ac-n
balance, minimum_balance, service_charge);
do
{
System.out.println("enter choice: \n 1. deposit
2. withdraw \n 3. display balance");
int c = sc.nextInt();
ca.checkMinimumBalance();
If (c==1)
{
System.out.println("enter amount to be deposited
double amt = sc.nextDouble();
ca.deposit(amt);}
else if (c==2)
{
System.out.println("enter amount to withdraw
double amt = sc.nextDouble();
ca.withdraw(amt);
else if (c==3)
{
```



```
ca.displayBalance();
}
else
System.exit(0)
} while(true);
}
case 2:
S.o.p("account is Savings type");
SavingsAccount sa = new Savings(name, acc-no, balance,
interest-rate);
do{ S.o.p("enter choice: \n 1. deposit \n 2. withdraw
3. display");
int c1 = sc.nextInt();
if(c==1)
{
S.o.p("enter amount to be deposited");
double amt = sc.nextDouble();
sa.deposit(amt);
}
else if(c1==2)
{
System.out.println("enter amount to withdraw");
double amt = sc.nextDouble();
sa.withdraw(amt);
}
else if(c1==3)
{
sa.computeanddepositInterest();
sa.displayBalance();
}
else
System.exit(0);
}
} while(true);
}
```



o/p
enter name : abc
enter accno : 99
enter initial balance : 9000
enter minimum balance : 500
enter interest rate : 12
enter service charge : 200
Enter choice :
1. current acc
2. saving acc.
1
Customer name is : abc
Account number : 99
account is current type
enter choice :
1. deposit
2. withdraw
3. display balance
1.
enter amount to be deposited :
7000
Deposited : 7000
enter choice
1. deposit
2. withdraw
3 display balance
3.
current balance : 16000
enter choice :
1. deposit
2. withdraw
3. display balance
2.
enter amount to withdraw :
800



withdrew : 800 balance 15200

20/11/24

```
enter customer name:
abc
enter accno:
99
enter initial balance:
9000
enter minimum balance:
500
enter interest rate:
12
enter service charge:
200
Enter choice:
 1.Current acc
 2.Savings acc
1
Customer name is:abc
Account number:99
Bhoomika BG-1BM23CS067
account is current type
enter choice:
 1.deposit
 2.withdraw
 3.display balance
1
enter amount to be deposited:
7000
Deposited: 7000.0
enter choice:
 1.deposit
 2.withdraw
 3.display balance
3
Current Balance: 16000.0
enter choice:
 1.deposit
 2.withdraw
 3.display balance
2
enter amount to withdraw:
800
withdrew:800.0 balance is:15200.0
enter choice:
 1.deposit
 2.withdraw
 3.display balance
3
Current Balance: 15200.0
enter choice:
 1.deposit
```

```
enter amount to withdraw:
800
withdrew:800.0 balance is:15200.0
enter choice:
 1.deposit
 2.withdraw
 3.display balance
3
Current Balance: 15200.0
enter choice:
 1.deposit
 2.withdraw
 3.display balance
```

Code:

```
import java.util.Scanner;

class Account {
    private String customer_name;
    private int acc_no;
    protected double balance;

    public Account(String customer_name, int acc_no, double balance) {
        this.customer_name = customer_name;
        this.acc_no = acc_no;
        this.balance = balance;
    }

    public double getBalance() {
        return balance;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: " + amount);
        } else {
            System.out.println("Deposit amount must be positive.");
        }
    }
   public void withdraw(double amount)
     {
       if(amount<=getBalance()){
          balance-=amount;
          System.out.println("withdrew:"+amount + " balance is:"+ balance);
          }
       else
         System.out.println("Insufficient funds!!");
     }
    public void displayBalance(){
        System.out.println("Current Balance: " + balance);
    }
}

class SavingsAccount extends Account {
    private double interestRate;
```

```
    public SavingsAccount(String customerName, int accountNumber,
                                        double initialBalance, double
                                        interestRate) {
        super(customerName, accountNumber, initialBalance);
        this.interestRate = interestRate;
    }

    public void computeAndDepositInterest() {
        double interest = getBalance() * interestRate / 100;
        deposit(interest);
    }
}
class CurrentAccount extends Account {
    private double minimumBalance;
    private double serviceCharge;

    public CurrentAccount(String customerName, int accountNumber,
                                        double initialBalance, double
                                        minimumBalance, double serviceCharge)
                                        {
        super(customerName, accountNumber, initialBalance);
        this.minimumBalance = minimumBalance;
        this.serviceCharge = serviceCharge;
    }
    public void checkMinimumBalance() {
        if (getBalance() < minimumBalance) {
            System.out.println("Balance is below minimum");
            balance-=serviceCharge;
            System.out.println("Deducted service charge:" +serviceCharge);
            System.out.println("Balance after deduction is:"+balance);
        }
    }
}

public class Bank {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("enter customer name:");
        String name=sc.nextLine();
        System.out.println("enter accno:");
        int acc_no=sc.nextInt();
        System.out.println("enter initial balance:");
        double balance=sc.nextDouble();
        System.out.println("enter minimum balance:");
```

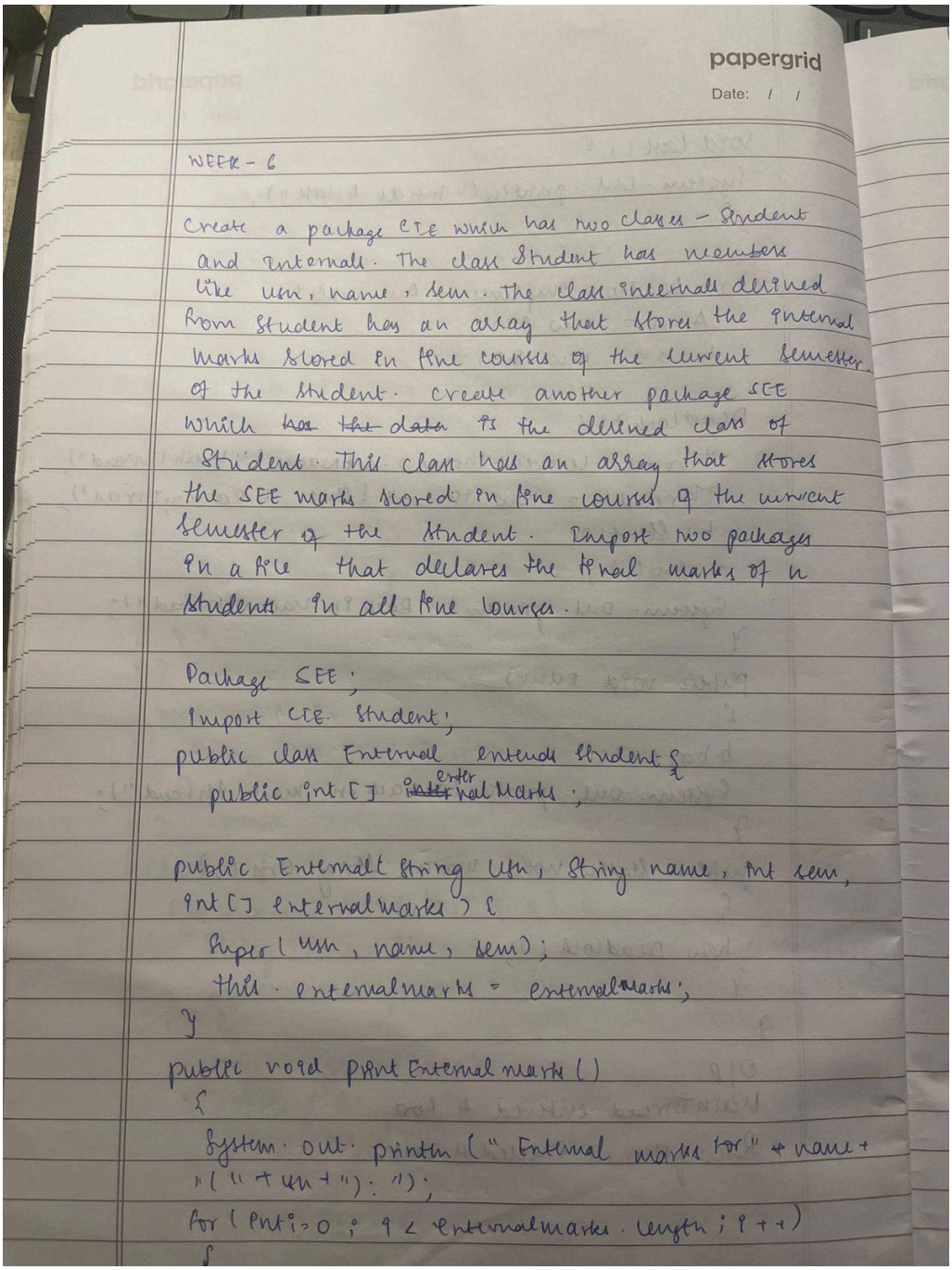
```
double minimum_balance=sc.nextDouble();
System.out.println("enter interest rate:");
double interest_rate=sc.nextDouble();
System.out.println("enter service charge:");
double service_charge=sc.nextDouble();
System.out.println("Enter choice:\n 1.Current acc\n 2.Savings acc");
int ch=sc.nextInt();
System.out.println("Customer name is:"+ name+"\nAccount
                                number:"+acc_no+");

switch(ch){
   case(1):
      System.out.println("account is current type");
      CurrentAccount ca = new
                                CurrentAccount(name,acc_no,balance,mi
                                nimum_balance,service_charge);
     do{ System.out.println("enter choice:\n 1.deposit\n 2.withdraw\n
                                3.display balance");
      int c=sc.nextInt();
      ca.checkMinimumBalance();
      if(c==1){
        System.out.println("enter amount to be deposited:");
        double amt=sc.nextDouble();
          ca.deposit(amt);}
      else if(c==2){
        System.out.println("enter amount to withdraw:");
        double amt=sc.nextDouble();
        ca.withdraw(amt);}
      else if(c==3){
        ca.displayBalance();}
      else
       System.exit(0);
       }while(true);

  case(2):
       System.out.println("account is savings type");
      SavingsAccount sa=new
                                SavingsAccount(name,acc_no,balance,in
                                terest_rate);
      do{ System.out.println("enter choice:\n 1.deposit\n
                                2.withdraw\n 3.display balance");
      int c1=sc.nextInt();
      if(c1==1){
        System.out.println("enter amount to be deposited:");
```

```
                double amt=sc.nextDouble();
                  sa.deposit(amt);}
              else if(c1==2){
                System.out.println("enter amount to withdraw:");
                double amt=sc.nextDouble();
                sa.withdraw(amt);}
              else if(c1==3){
               sa.computeAndDepositInterest();
                sa.displayBalance();}
              else{
                System.exit(0);
                    }
              }while(true);
    }
}
}
```

Program 6:

Create a package CIE which has two classes- Student and Internals. The classStudent 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.

Algorithm:



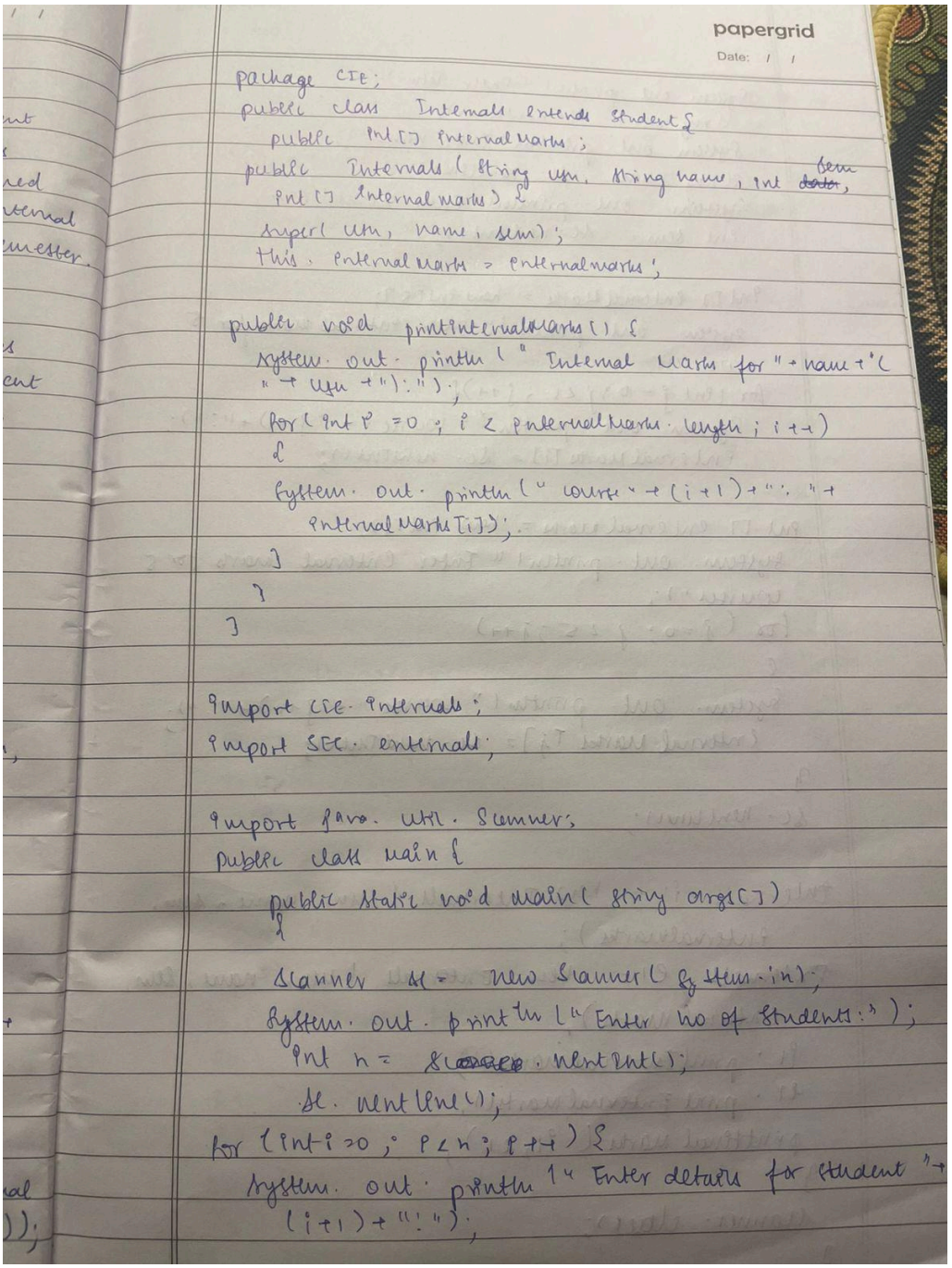
papergrid

Date: / /

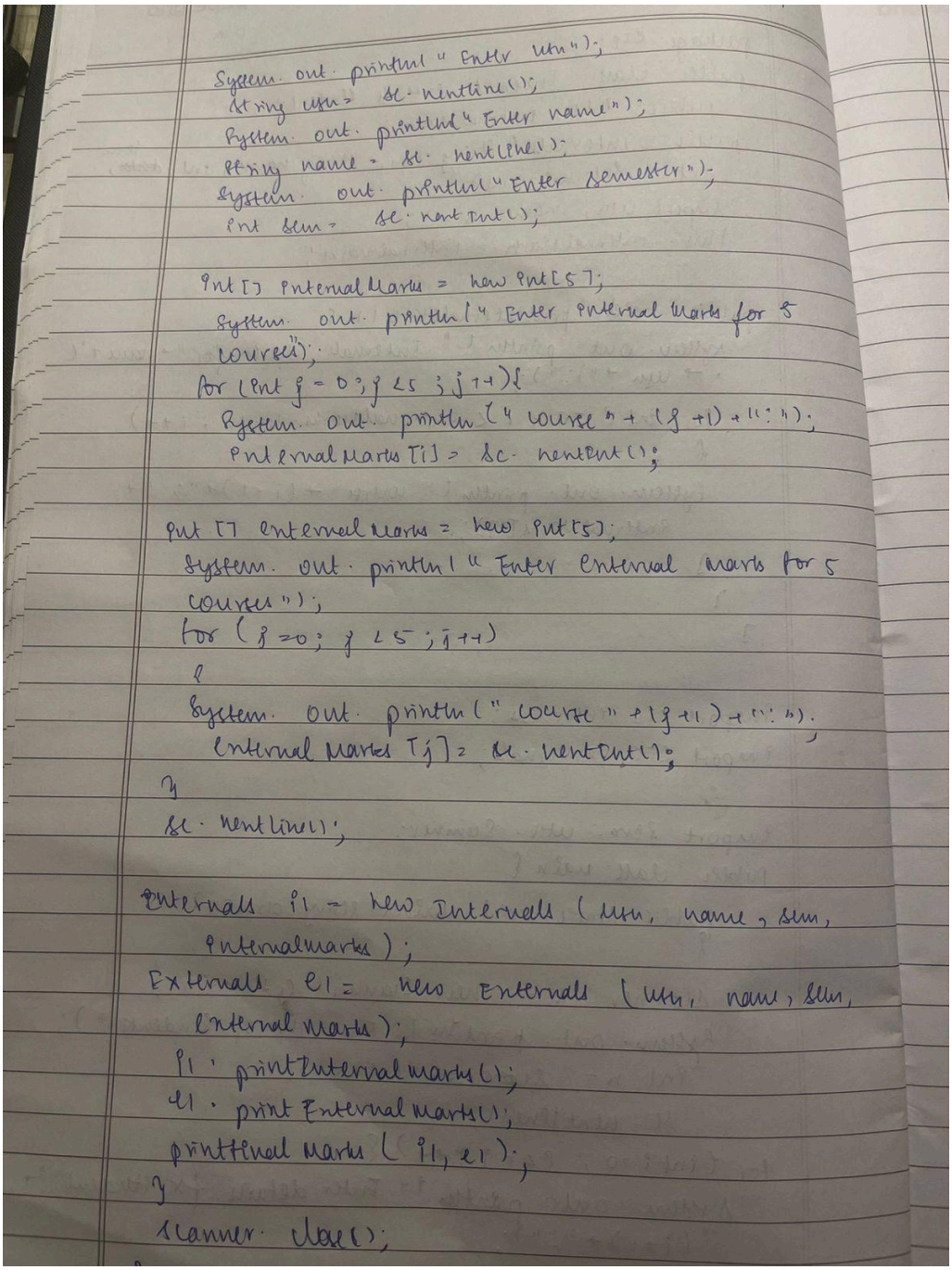
WEEK - 6

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 data~~ is the 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 two packages in a file that declares the final marks of n students in all five courses.

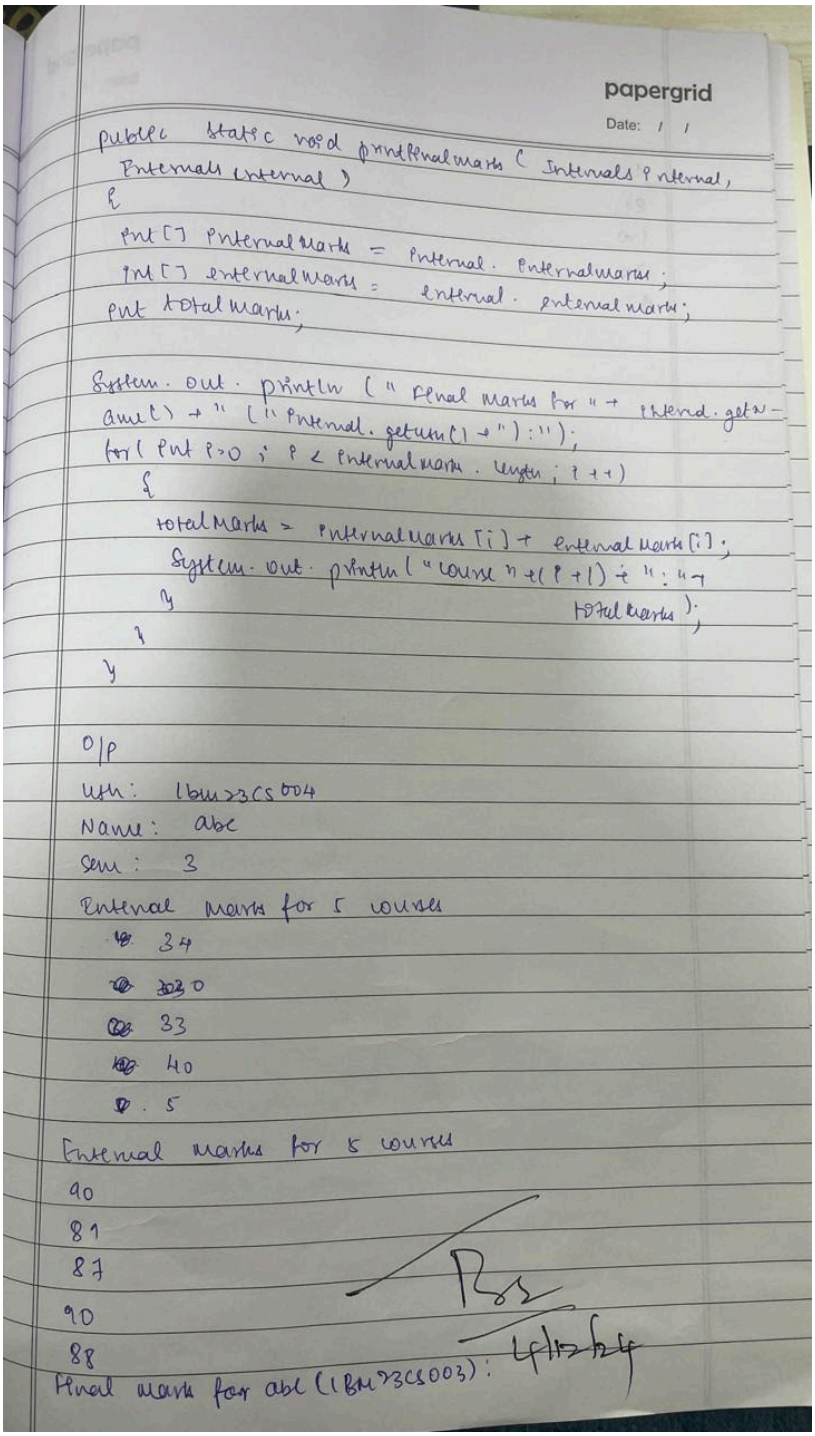
```
Package SEE;
Import CIE.Student;
public class External extends Student {
  public int[] enterernal Marks;

public External(String usn, String name, int sem,
int[] externalmarks) {
  Super(usn, name, sem);
  this.externalmarks = externalmarks;
}
public void printExternalmarks()
 {
  System.out.println("External marks for" + name +
"(" + usn + "):");
for(int i=0; i< externalmarks.length; i++)
 {
```



```
package CIE;
public class Internals extends Student {
    public int[] internalMarks;
public Internals(String usn, String name, int sem,
    int[] internalMarks) {
    super(usn, name, sem);
    this.internalMarks = internalMarks;

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

import CIE.Internals;
import SEE.externals;

import java.util.Scanner;
public class main {
    public static void main(String args[])
    {

    Scanner sc = new Scanner(System.in);
    System.out.println("Enter no of students:");
      int n = sc.nextInt();
      sc.nextLine();
for(int i=0; i<n; i++) {
  System.out.println("Enter details for student " +
    (i+1) + "!");
```



```
System.out.println("Enter usn");
String usn = sc.nextLine();
System.out.println("Enter name");
String name = sc.nextLine();
System.out.println("Enter Semester");
int sem = sc.nextInt();

int[] internalMarks = new int[5];
System.out.println("Enter internal marks for 5 courses");
for(int j=0;j<5;j++){
System.out.println("course"+(j+1)+":");
internalMarks[i] = sc.nextInt();

int[] externalMarks = new int[5];
System.out.println("Enter external marks for 5 courses");
for(j=0;j<5;j++)
{
System.out.println("course"+(j+1)+":");
externalMarks[j] = sc.nextInt();
}
sc.nextLine();

Internals i1 = new Internals(usn, name, sem, internalMarks);
External e1 = new Externals(usn, name, sem, externalMarks);
i1.printInternalMarks();
e1.printExternalMarks();
printFinalMarks(i1, e1);
}
scanner.close();
```



```
public static void printFinalMarks ( Internals internal,
External external )
{
int[] internalMarks = internal.internalMarks;
int[] externalMarks = external.externalMarks;
int totalMarks;

System.out.println (" Final Marks for " + internal.getName() + " (" internal.getusn() + "):");
for( int i=0 ; i < internalMarks.length ; i++)
  {
    totalMarks = internalMarks[i] + externalMarks[i];
    System.out.println("course no "+(i+1)+" : "+
                                    totalMarks);
  }
}
```

O/P

usn: 1bm23cs004

Name: abc

Sem: 3

External marks for 5 courses

34

30

33

40

5

Internal marks for 5 courses

90

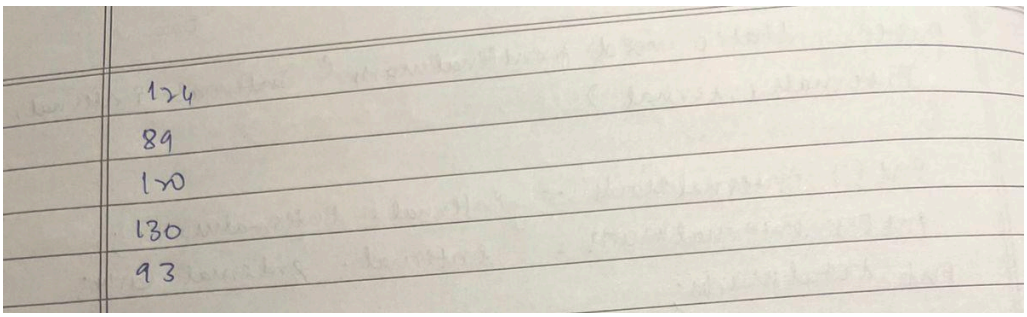
81

87

90

88

Final mark for abc (1BM23CS003):



124

89

130

130

93

```
Enter details for Student 1:
Enter USN: 1bm23cs003
Enter Name: abc
Enter Semester: 3
Enter Internal Marks for 5 Courses:
Course 1: 34
Course 2: 0
Course 3: 33
Course 4: 40
Course 5: 5
Enter External Marks for 5 Courses:
Course 1: 90
Course 2: 89
Course 3: 87
Course 4: 90
Course 5: 88
Internal Marks for abc (1bm23cs003):
Course 1: 34
Course 2: 0
Course 3: 33
Course 4: 40
Course 5: 5
External Marks for abc (1bm23cs003):
Course 1: 90
Course 2: 89
Course 3: 87
Course 4: 90
Course 5: 88
Final Marks for abc (1bm23cs003):
Course 1: 124
Course 2: 89
Course 3: 120
Course 4: 130
Course 5: 93

Enter details for Student 2:
Enter USN: 1bm23cs019
Enter Name: bgh
Enter Semester: 3
Enter Internal Marks for 5 Courses:
Course 1: 39
Course 2: 40
Course 3: 35
Course 4: 36
Course 5: 37
Enter External Marks for 5 Courses:
Course 1: 90
Course 2: 97
```

```
Enter details for Student 2:
Enter USN: 1bm23cs019
Enter Name: bgh
Enter Semester: 3
Enter Internal Marks for 5 Courses:
Course 1: 39
Course 2: 40
Course 3: 35
Course 4: 36
Course 5: 37
Enter External Marks for 5 Courses:
Course 1: 90
Course 2: 97
Course 3: 96
Course 4: 92
Course 5: 91
Internal Marks for bgh (1bm23cs019):
Course 1: 39
Course 2: 40
Course 3: 35
Course 4: 36
Course 5: 37
External Marks for bgh (1bm23cs019):
Course 1: 90
Course 2: 97
Course 3: 96
Course 4: 92
Course 5: 91
Final Marks for bgh (1bm23cs019):
Course 1: 129
Course 2: 137
Course 3: 131
Course 4: 128
Course 5: 128

Enter details for Student 3:
Enter USN: 1bm23cs017
Enter Name: lll
Enter Semester: 3
Enter Internal Marks for 5 Courses:
Course 1: 35
Course 2: 31
Course 3: 23
Course 4: 38
Course 5: 40
Enter External Marks for 5 Courses:
Course 1: 88
Course 2: 87
Course 3: 85
```

```
Enter details for Student 3:
Enter USN: 1bm23cs017
Enter Name: lll
Enter Semester: 3
Enter Internal Marks for 5 Courses:
Course 1: 35
Course 2: 31
Course 3: 23
Course 4: 38
Course 5: 40
Enter External Marks for 5 Courses:
Course 1: 88
Course 2: 87
Course 3: 85
Course 4: 90
Course 5: 91
Internal Marks for lll (1bm23cs017)
Course 1: 35
Course 2: 31
Course 3: 23
Course 4: 38
Course 5: 40
External Marks for lll (1bm23cs017)
Course 1: 88
Course 2: 87
Course 3: 85
Course 4: 90
Course 5: 91
Final Marks for lll (1bm23cs017):
Course 1: 123
Course 2: 118
Course 3: 108
Course 4: 128
Course 5: 131
PS D:\1BM23CS019>
```

Code:

```
package SEE;

import CIE.Student;

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

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

      public void printExternalMarks() {
```

```
        System.out.println("External Marks for " + name + " (" + usn + "):");
        for (int i = 0; i < externalMarks.length; i++) {
            System.out.println("Course " + (i + 1) + ": " + externalMarks[i]);
        }
    }
}

package CIE;

public class Internals extends Student {
    public int[] internalMarks;


    public Internals(String usn, String name, int sem, int[] internalMarks) {
        super(usn, name, sem);
        this.internalMarks = internalMarks;
    }


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

import CIE.Internals;
import SEE.External;

import java.util.Scanner;

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


        System.out.print("Enter number of students: ");
        int n = scanner.nextInt();
        scanner.nextLine();  // Consume the newline
t
        for (int i = 0; i < n; i++) {
            // Read student details
            System.out.println("\nEnter details for Student " + (i + 1) + ":");
```

```
        System.out.print("Enter USN: ");
        String usn = scanner.nextLine();

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

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

        int[] internalMarks = new int[5];
        System.out.println("Enter Internal Marks for 5 Courses:");
        for (int j = 0; j < 5; j++) {
            System.out.print("Course " + (j + 1) + ": ");
            internalMarks[j] = scanner.nextInt();
        }


        int[] externalMarks = new int[5];
        System.out.println("Enter External Marks for 5 Courses:");
        for (int j = 0; j < 5; j++) {
            System.out.print("Course " + (j + 1) + ": ");
            externalMarks[j] = scanner.nextInt();
        }
        scanner.nextLine();  // Consume the newline after integer input

                Internals internalStudent = new Internals(usn, name, sem,
                                        internalMarks);
        External externalStudent = new External(usn, name, sem,
                                        externalMarks);


        internalStudent.printInternalMarks();
        externalStudent.printExternalMarks();


        printFinalMarks(internalStudent, externalStudent);
    }

    scanner.close();
}

   public static void printFinalMarks(Internals internal, External
                                        external) {
```

```
        int[] internalMarks = internal.internalMarks;
        int[] externalMarks = external.externalMarks;
        int totalMarks;

        System.out.println("Final Marks for " + internal.getName() + " (" +
                                        internal.getUsn() + "):");
        for (int i = 0; i < internalMarks.length; i++) {
            totalMarks = internalMarks[i] + externalMarks[i];
            System.out.println("Course " + (i + 1) + ": " + totalMarks);
        }
    }
}
```

Program 7:

Write a program that demonstrates handling of exceptions in inherited tree. Create a base class called father and a derived class called son which extends the base class. In fathers class implement a constructor which takes the age and throws the exception wrongrage() when the input age is less tahn zero. In sons class implement a constructor that uses father and sons age and throws an exception if sons age is greater than or equal to fathers age

```
:\Users\Admin\Desktop>java FatherSon
ksha 1bm23cs019
nter Father's Age: 45
nter Son's Age: 19
ccepted Succesfully
ould you like to re-enter details (Y/n)

ksha 1bm23cs019
nter Father's Age: 45
nter Son's Age: 46
on's age cannot be greater than or equal to father's age
ould you like to re-enter details (Y/n)
```

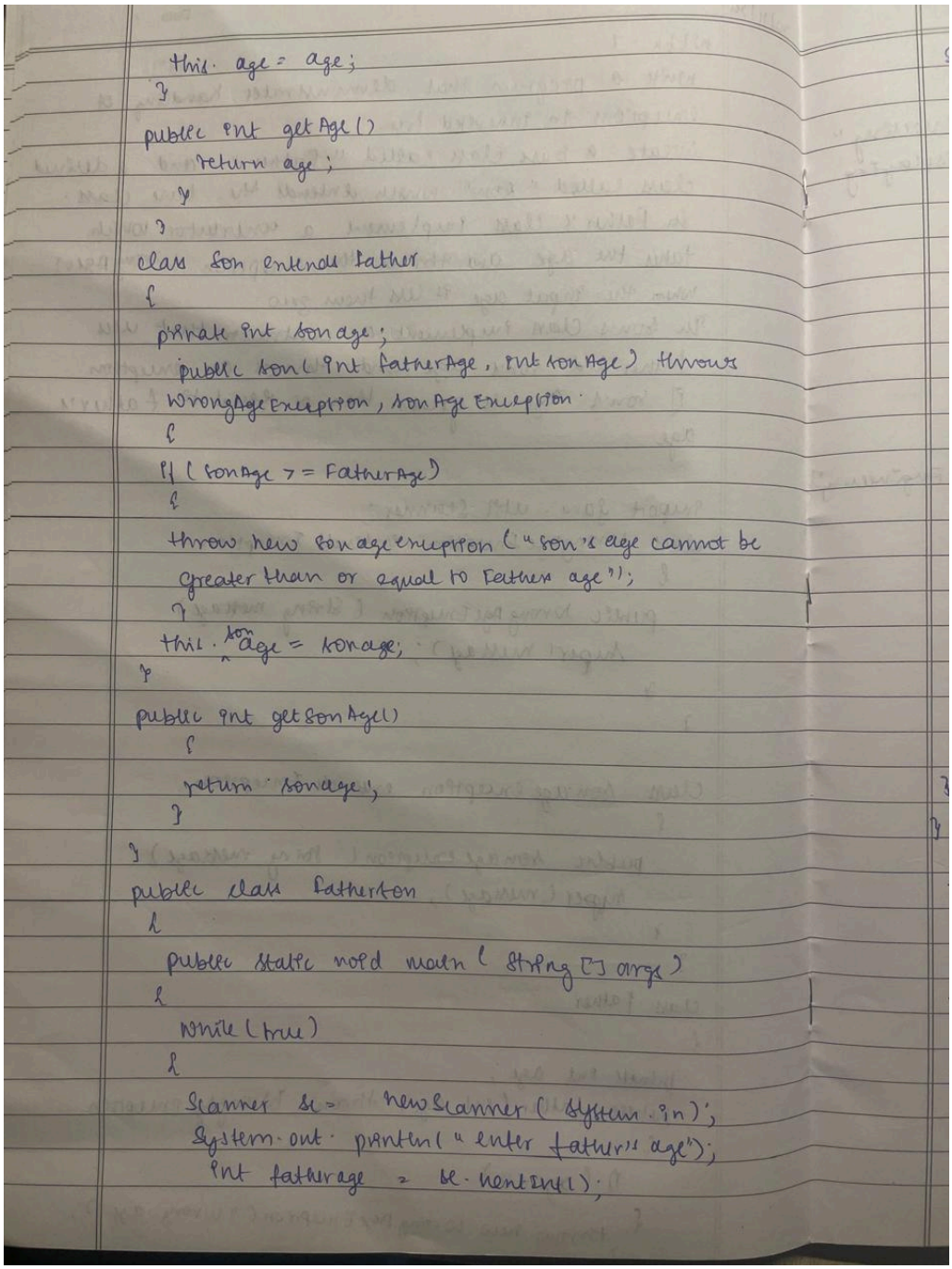
WEEK - 7

papergrid

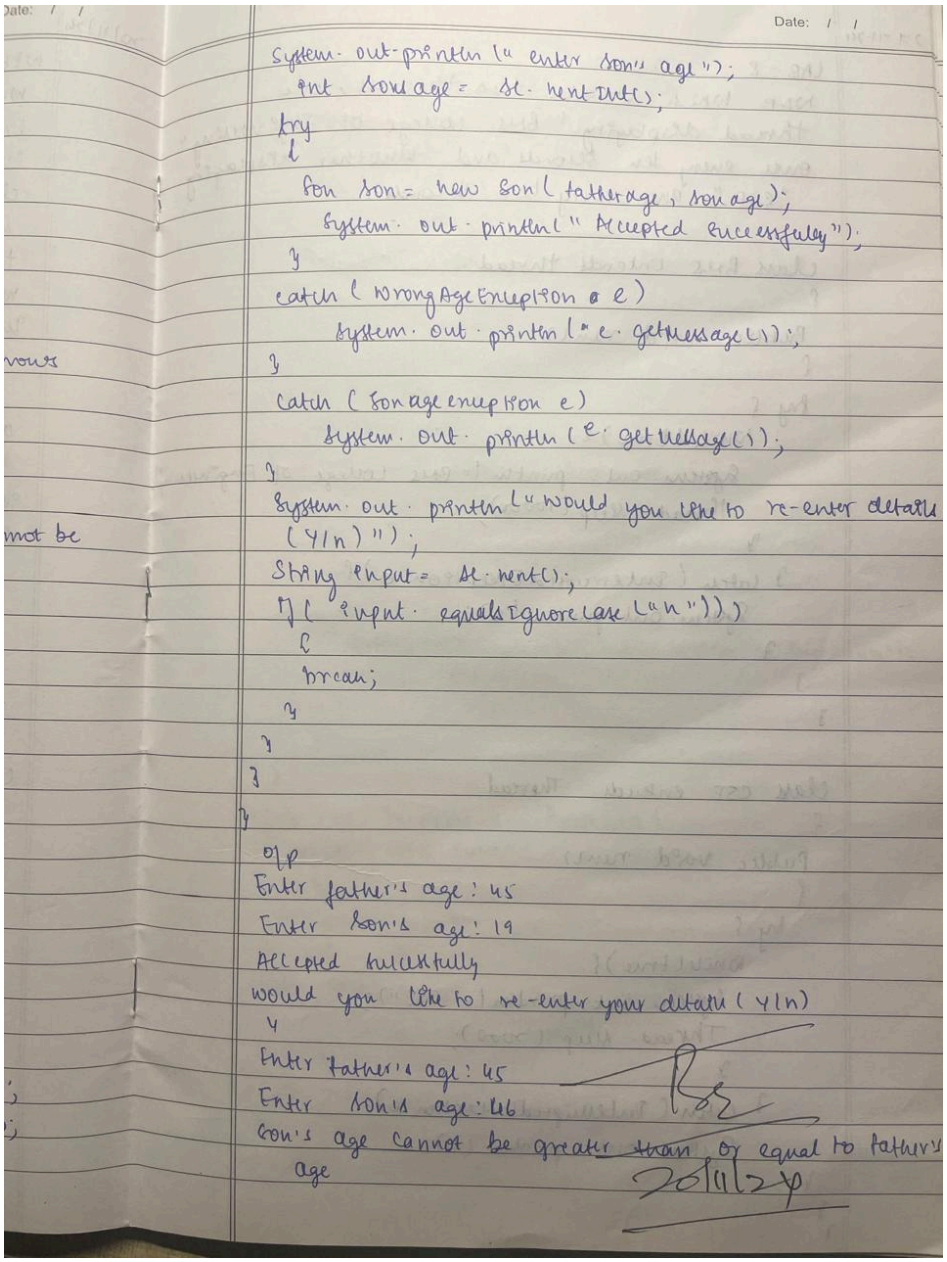
Date: / /

Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called "Father" and a derived class called "Son" which extends the base class. In Father's class implement a constructor which takes the age and throws the exception WrongAge() when the input age is less than zero. In Son's class implement a constructor that uses Father and Son's age and throws an exception if Son's age is greater than or equal to Father's age.

```
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
{
private int age;
public Father(int age) throws WrongAgeException
 {
   if (age<0)
{ throw new WrongAgeException("wrong age");
```



```
    this.age = age;
  }
public int getAge()
  {
    return age;
  }
}
class Son extends Father
{
  private int sonAge;
  public Son(int fatherAge, int sonAge) throws
  WrongAgeException, SonAgeException
  {
  if (sonAge >= FatherAge)
  {
  throw new SonAgeException("Son's age cannot be
  greater than or equal to Father age");
  }
  this.sonAge = sonAge;
}
public int getSonAge()
  {
  return sonAge;
  }
}
public class FatherSon
{
  public static void main(String[] args)
{
  while(true)
  {
  Scanner sc = new Scanner(System.in);
  System.out.println("Enter father's age");
  int fatherage = sc.nextInt();
```



```
System.out.println("enter son's age");
int sonage = sc.nextInt();
try
{
Son son = new Son(fatherage, sonage);
System.out.println("Accepted successfully");
}
catch (WrongAgeException e)
System.out.println(e.getMessage());
}
catch (SonageException e)
System.out.println(e.getMessage());
}
System.out.println("would you like to re-enter details
(y/n)");
String input = sc.next();
if (input.equalsIgnoreCase("n"))
{
break;
}
}
}
}
```

```
o/p
Enter father's age: 45
Enter son's age: 19
Accepted successfully
would you like to re-enter your details (y/n)
y
Enter father's age: 45
Enter son's age: 46
Son's age cannot be greater than or equal to father's
age
```

20/11/24

Code:

```
class Father {
    private int age;
    public Father(int age) throws WrongAgeException {
        if (age < 0) {
            throw new WrongAgeException("Wrong age");
```

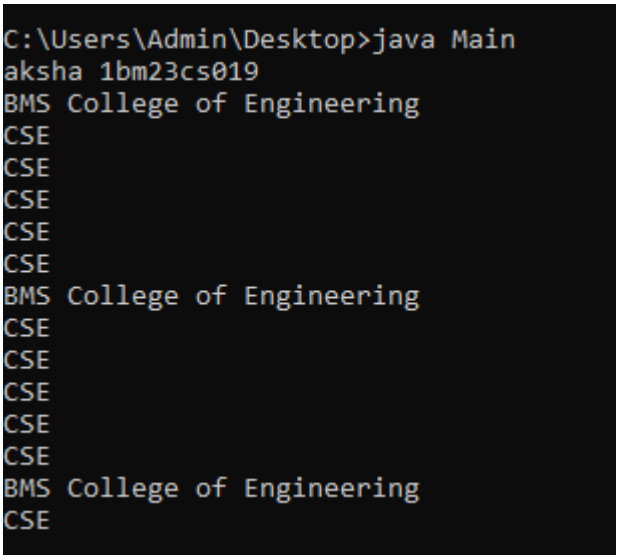
```
        }
        this.age = age;
    }
    public int getAge() {
        return age;
    }
}

class Son extends Father {
    private int sonAge;
    public Son(int fatherAge, int sonAge) throws WrongAgeException, SonAgeException {
        super(fatherAge);
        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 FatherSon{
    public static void main(String[] args) {
        while(true){
            Scanner sc = new Scanner(System.in);
            System.out.println("aksha 1bm23cs019");
            System.out.print("Enter Father's Age: ");
            int fatherAge = sc.nextInt();
            System.out.print("Enter Son's Age: ");
            int sonAge = sc.nextInt();
            try {
                Son son = new Son(fatherAge, sonAge);
                System.out.println("Accepted Succesfully");
            }
            catch (WrongAgeException e) {
                System.out.println(e.getMessage());
            }
            catch (SonAgeException e) {
                System.out.println(e.getMessage());
            }
            System.out.println("Would you like to re-enter details (Y/n)");
            String input = sc.next();
            if (input.equalsIgnoreCase("n")) {
                break;
```

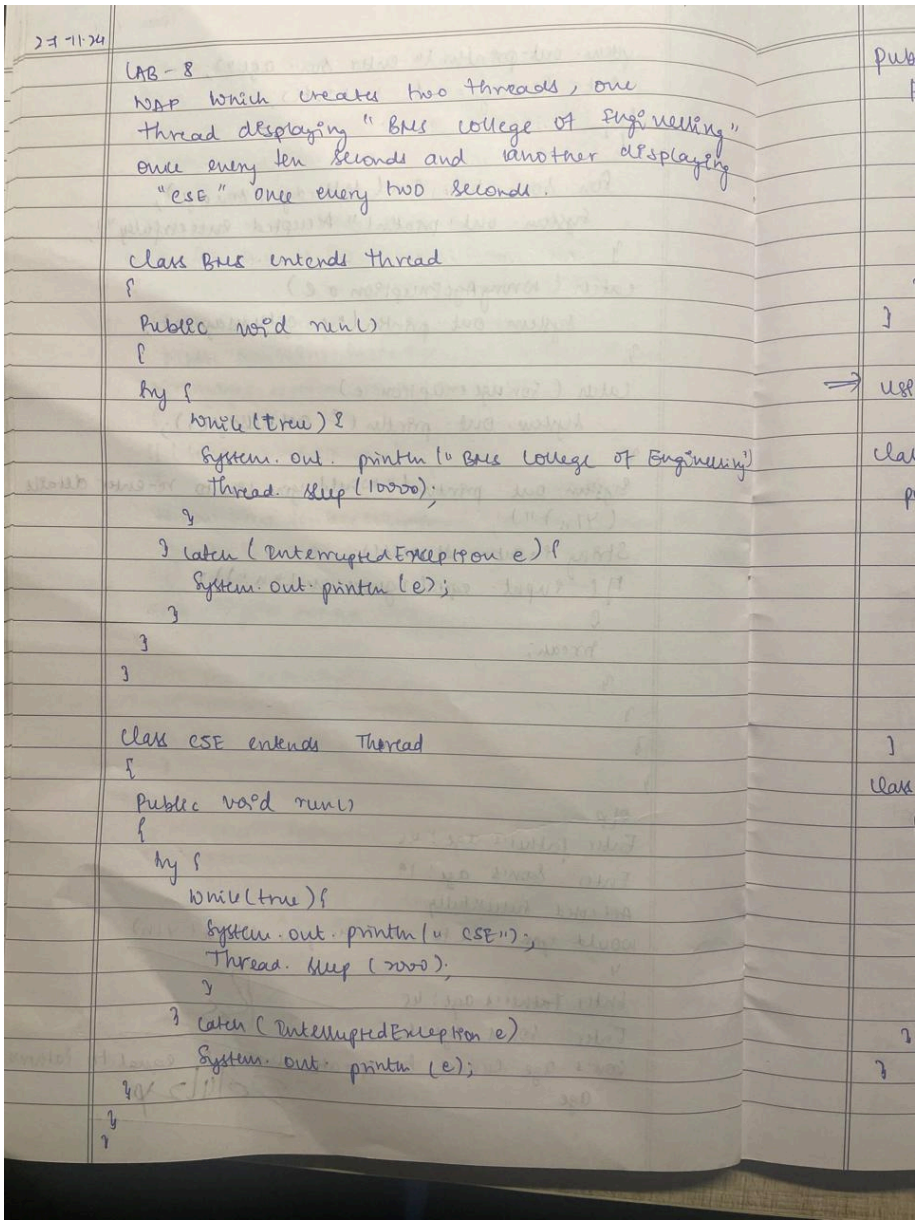
```
        }
    }
}
```

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.

Algorithm:

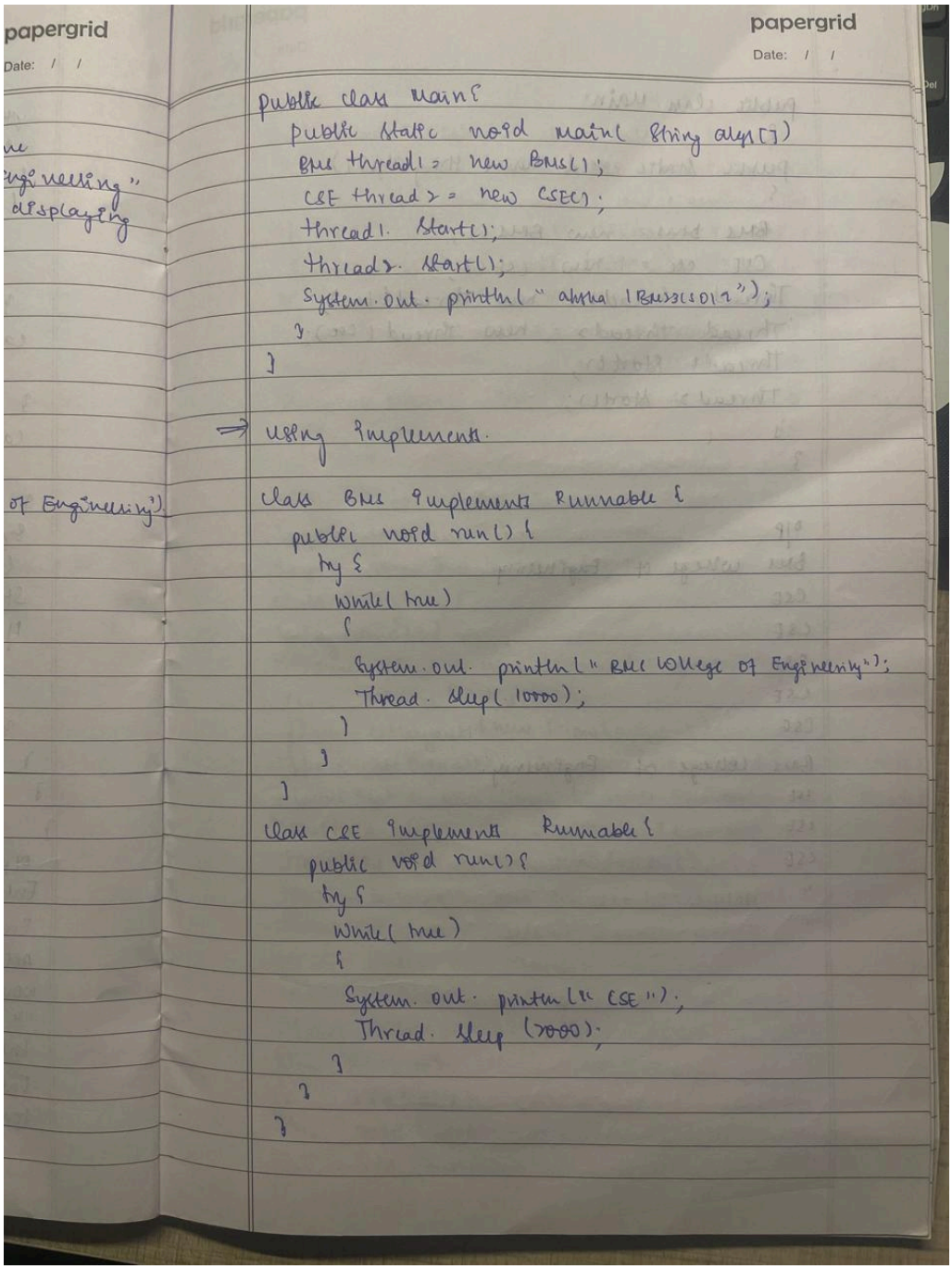




LAB - 8

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

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

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



```
public class main{
  public static void main(String args[])
  BMS thread1 = new BMS();
  CSE thread2 = new CSE();
  thread1.start();
  thread2.start();
  System.out.println("alpha | BMSCE012");
  }
}
```

⇒ Using Implements.

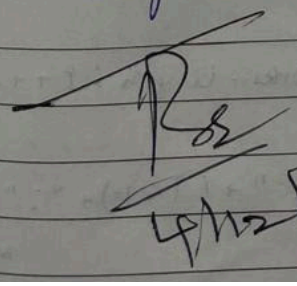
```
class BMS implements Runnable {
  public void run() {
    try {
     while(true)
     {
      System.out.println("BMS College of Engineering");
      Thread.sleep(10000);
     }
    }
}
class CSE implements Runnable{
   public void run(){
    try{
     while(true)
     {
      System.out.println("CSE");
      Thread.sleep(7000);
     }
   }
}
```

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

Deadlock() {
    Thread.currentThread().setName("MainThread");
    Thread t = new Thread(this, "RacingThread");
    t.start();
    a.foo(b);
    System.out.println("Back in main thread");
}

public void run()
 {
  b.bar(a);
  System.out.println("Back in main thread");
}
public static void main(String args[])
 {
  new Deadlock();
 }
}
```

O/P

MainThread entered A.foo

RacingThread entered B.bar.



code:

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

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

public class Main {
    public static void main(String[] args) {
        // Creating two threads
        BMS thread1 = new BMS();
        CSE thread2 = new CSE();

        // Starting the threads
        thread1.start();
        thread2.start();
        System.out.println("aksha 1bm23cs019");
    }
}
class BMS implements Runnable {
    public void run() {
        try {
            while (true) {
```

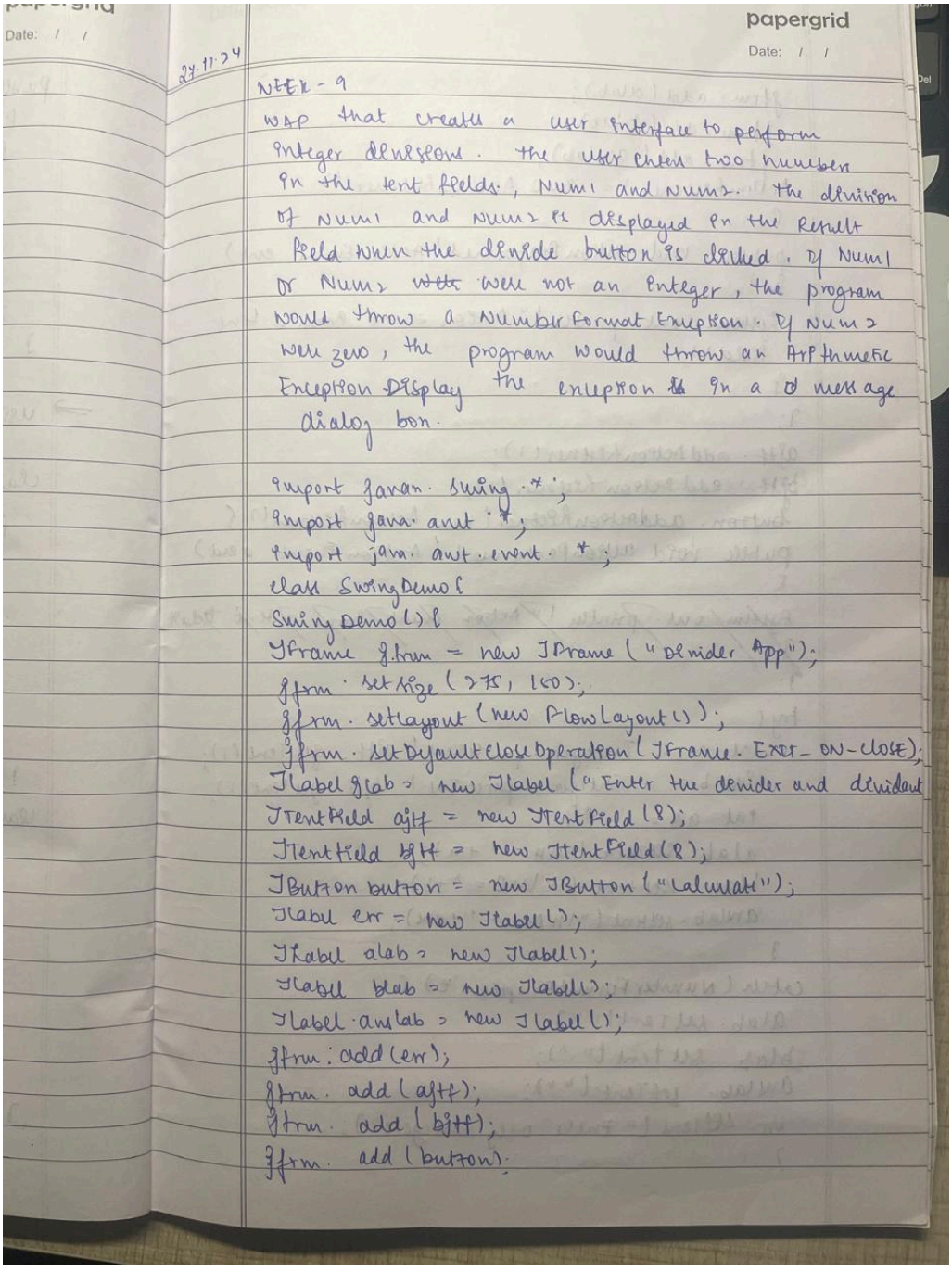
```
            System.out.println("BMS College of Engineering");
            Thread.sleep(10000); // Sleep for 10 seconds
        }
      } catch (InterruptedException e) {
        System.out.println(e);
      }
    }
}

class CSE implements Runnable {
    public void run() {
      try {
        while (true) {
          System.out.println("CSE");
          Thread.sleep(2000); // Sleep for 2 seconds
        }
      } catch (InterruptedException e) {
        System.out.println(e);
      }
    }
}

public class Demo {
    public static void main(String[] args) {
      // Create instances of Runnable classes
      BMS bms = new BMS();
      CSE cse = new CSE();

      // Create Thread objects and pass the Runnable instances to them
      Thread thread1 = new Thread(bms);
      Thread thread2 = new Thread(cse);

      // Start the threads
      thread1.start();
      thread2.start();
      System.out.println("aksha 1bm23cs019");
    }
}
```

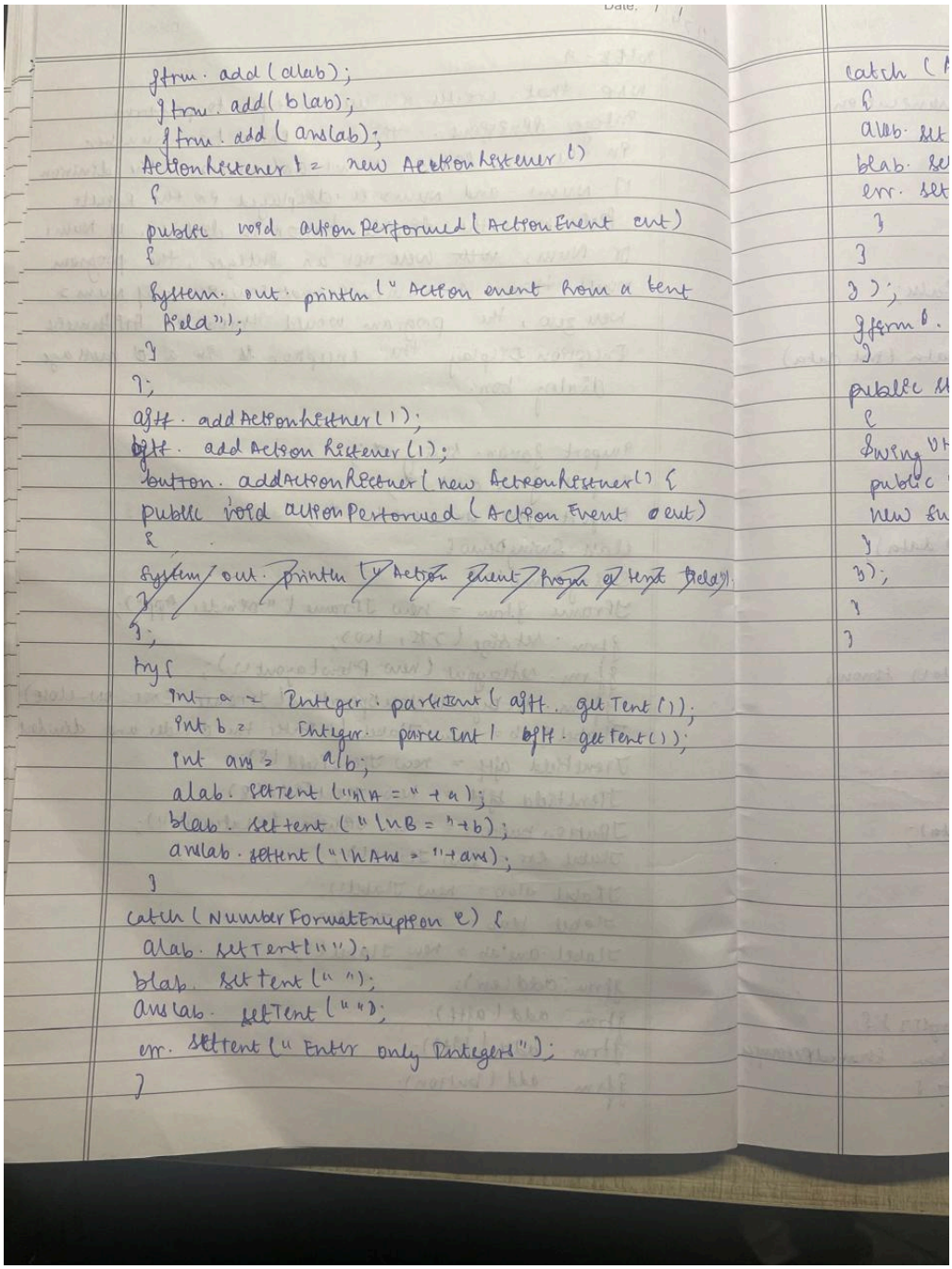
Program 9:

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.

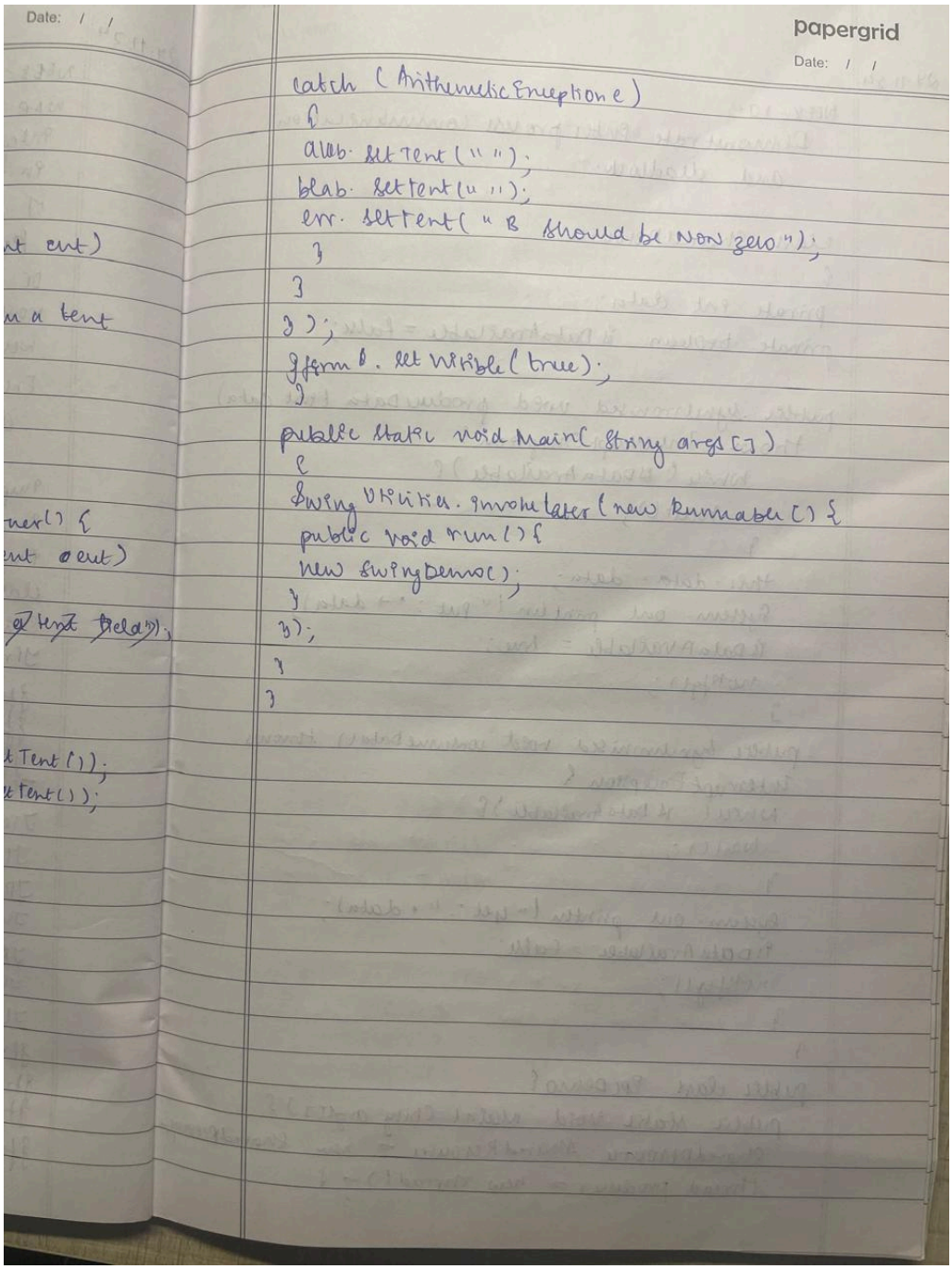
Algorithm:



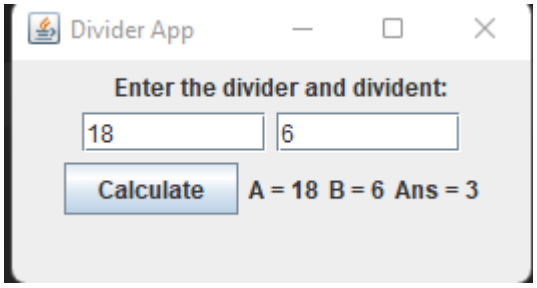


```
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);
ActionListener l = new ActionListener()
{
public void actionPerformed(ActionEvent ent)
{
System.out.println("Action event from a text field");
}
};
aff.addActionListener(l);
bff.addActionListener(l);
button.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent ent)
{
System.out.println("Action Event from text field");
}
};
try{
int a = Integer.parseInt(aff.getText());
int b = Integer.parseInt(bff.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(A
{
alab.set
blab.se
err.set
}
});
jfrm.
}
public st
{
Swing
public
new Sw
}
});
}
}
```



```
catch (ArithmeticException e)
{
alab.setText("");
blab.setText("");
err.setText("B should be NON zero");
}
}
});
jform1.setVisible(true);
}
public static void main(String args[])
{
SwingUtilities.invokeLater(new Runnable() {
public void run(){
new SwingDemo();
}
});
}
}
```



Code:

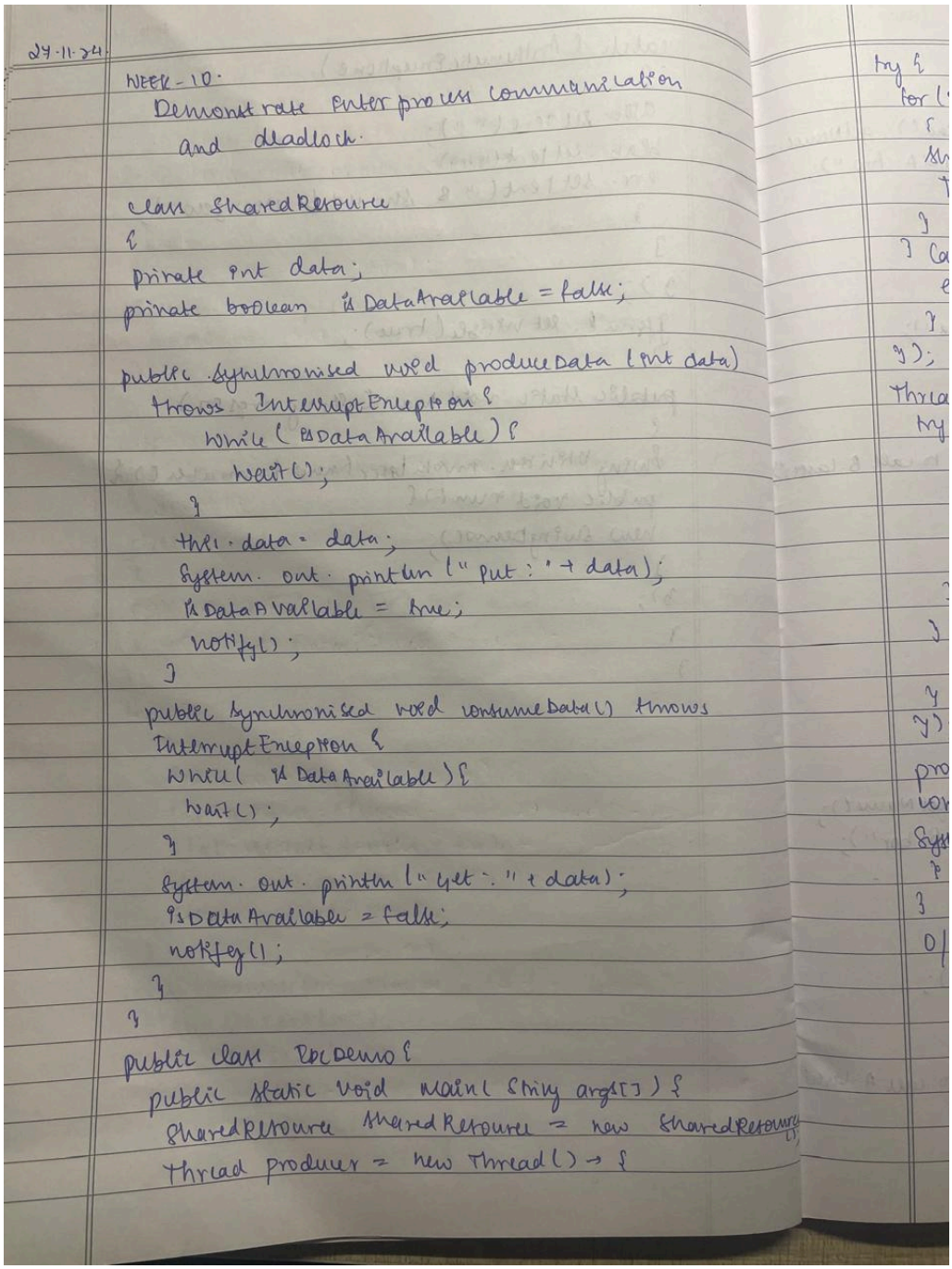
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class SwingDemo{
SwingDemo(){
// create jframe container
JFrame jfrm = new JFrame("Divider App");
jfrm.setSize(275, 150);
jfrm.setLayout(new FlowLayout());
// to terminate on close
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// 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();
}
});
}
}
```

Program 10:
Open Ended Exercise

Demonstrate Inter process Communication and deadlock

Algorithm:

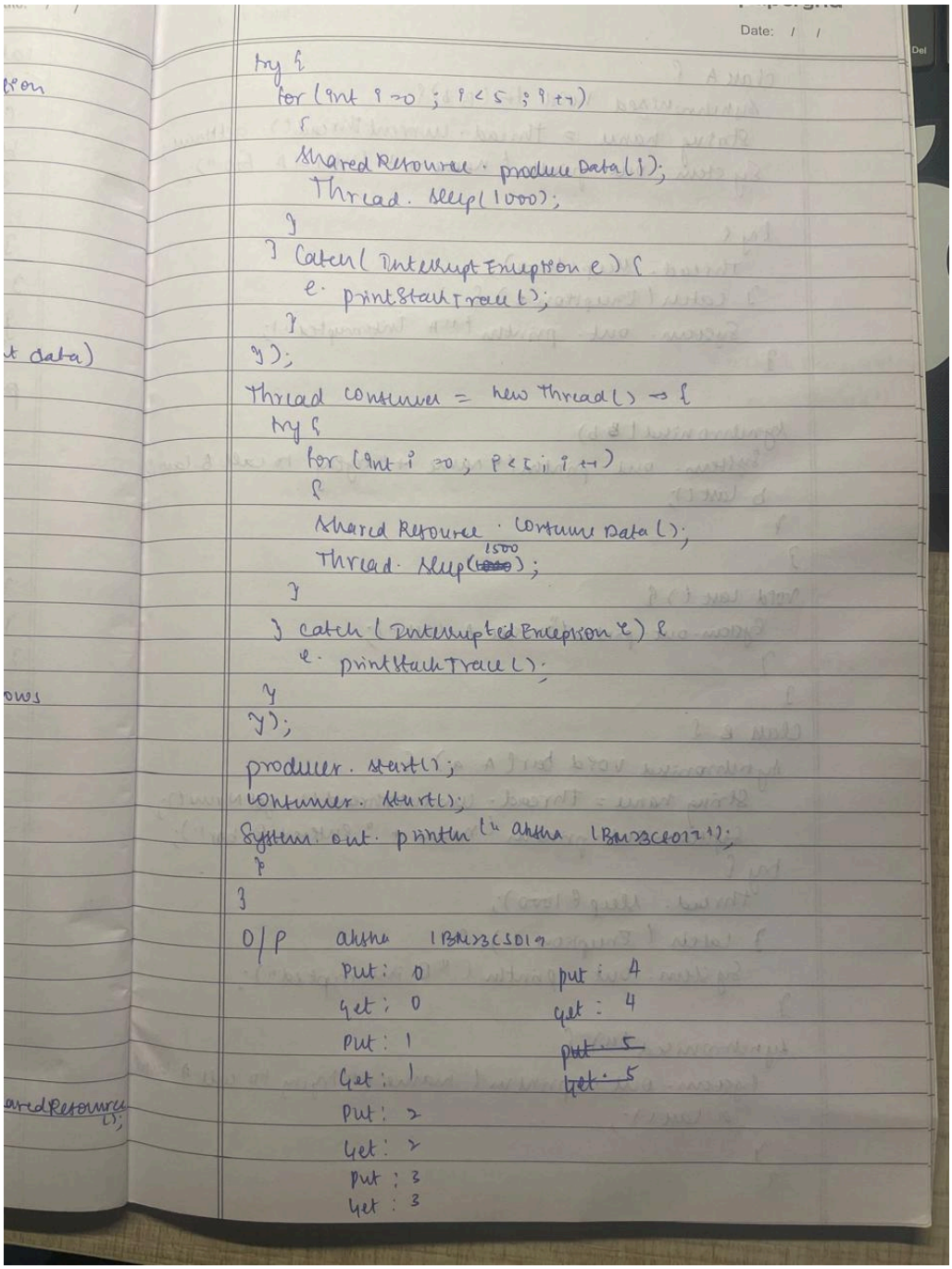


24-11-24

WEEK-10.

Demonstrate Inter process communication and deadlock.

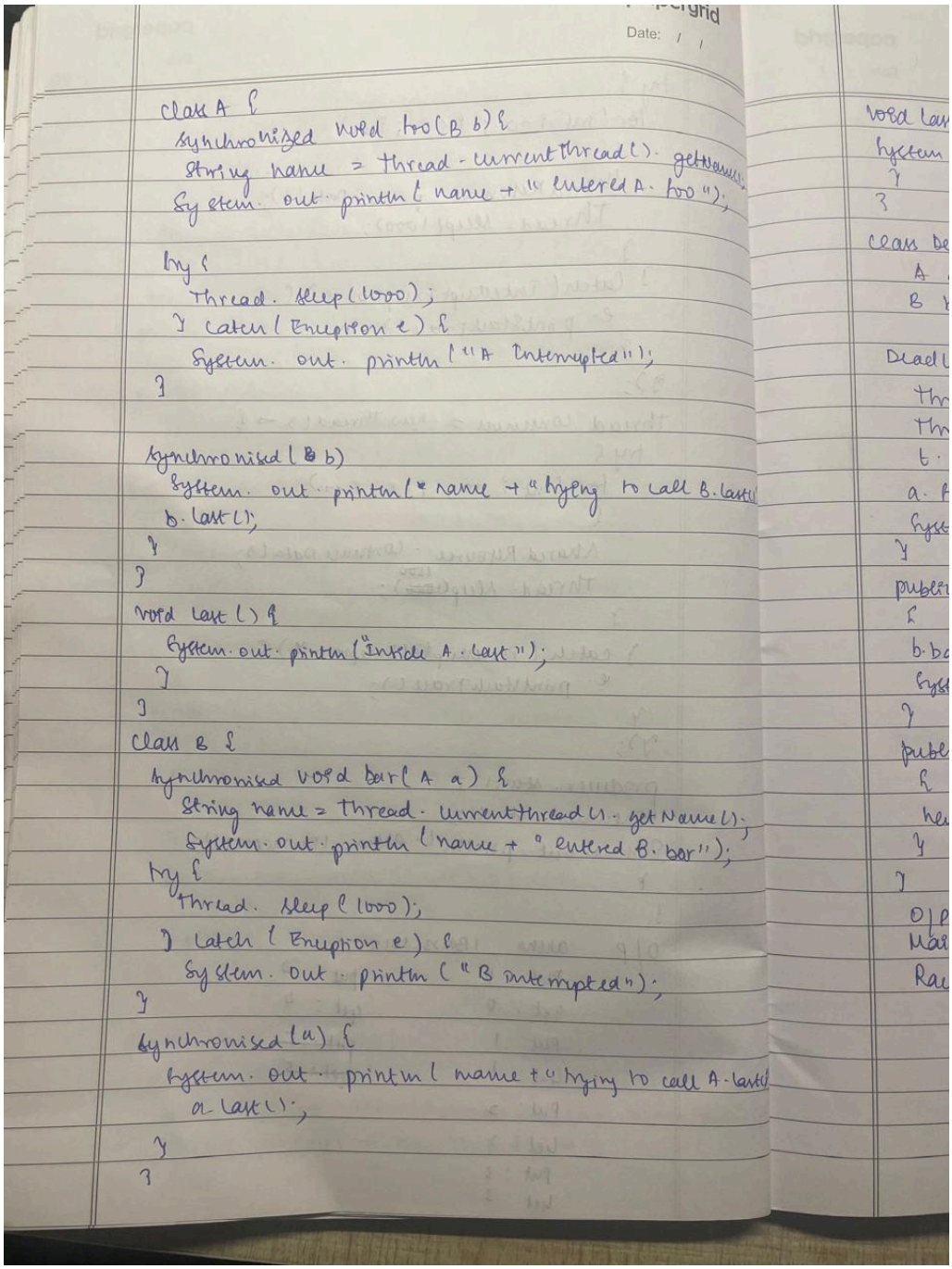
```
class SharedResource
{
private int data;
private boolean isDataAvailable = false;

public synchronised void produceData (int data)
  throws InterruptException {
      while (isDataAvailable) {
        wait();
      }
    this.data = data;
    System.out.println("Put: " + data);
    isDataAvailable = true;
    notify();
  }
  public synchronised void consumeData() throws
  InterruptException {
    while (!isDataAvailable) {
      wait();
    }
    System.out.println("Get: " + data);
    isDataAvailable = false;
    notify();
  }
}
public class PCDemo {
  public static void main(String args[]) {
    SharedResource sharedResource = new SharedResource();
    Thread producer = new Thread() -> {
```



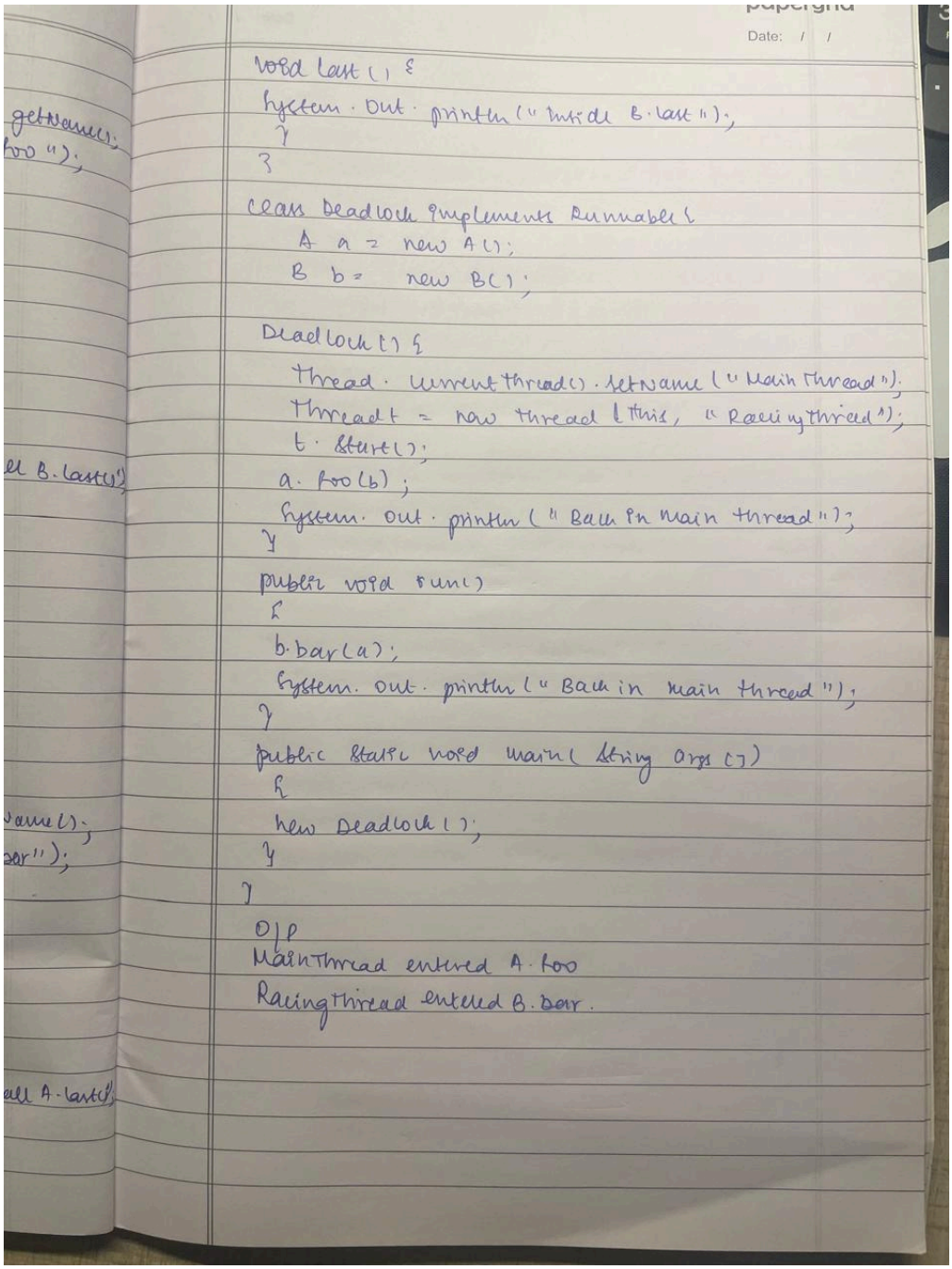
```
try {
  for (int i=0 ; i<5 ; i++)
  {
    SharedResource.produceData(i);
    Thread.sleep(1000);
  }
} catch(InterruptException e) {
   e.printStackTrace();
  }
});
Thread consumer = new Thread(() -> {
  try {
     for (int i =0; i<5; i++)
     {
       SharedResource.consumeData();
       Thread.sleep(1500);
     }
  } catch (InterruptedException e) {
     e.printStackTrace();
  }
});
producer.start();
consumer.start();
System.out.println("akshu 1BM23CS017");
}
}
```

O/P akshu 1BM23CS017

Put: 0
Get: 0
Put: 1
Get: 1
Put: 2
Get: 2
Put: 3
Get: 3
Put: 4
Get: 4
~~Put: 5~~
~~Get: 5~~



```
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");
    }

    synchronized(b)
      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");
    }
    synchronized(a) {
      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);
    System.out.println("Back in main thread");
}

public void run()
{
    b.bar(a);
    System.out.println("Back in main thread");
}
public static void main(String args[])
{
    new Deadlock();
}
}
O/P
MainThread entered A.foo
RacingThread entered B.bar.
```

```
C:\Users\Admin\Desktop>java Deadlock
MainThread entered A.foo
RacingThread entered B.bar
```

code:

IPC

```
class SharedResource {
    private int data;
    private boolean isDataAvailable = false;
```

```
    // Producer thread
    public synchronized void produceData(int data) throws InterruptedException {
        while (isDataAvailable) {
            wait();  // Wait if data is already available
        }
        this.data = data;
        System.out.println("Put: " + data);
        isDataAvailable = true;
        notify();  // Notify consumer thread that data is available
    }

    // Consumer thread
    public synchronized void consumeData() throws InterruptedException {
        while (!isDataAvailable) {
            wait();  // Wait if no data is available
        }
        System.out.println("Get: " + data);
        isDataAvailable = false;
        notify();  // Notify producer thread that data is consumed
    }
}

public class IPCDemo {
    public static void main(String[] args) {
        SharedResource sharedResource = new SharedResource();

        // Producer thread
        Thread producer = new Thread(() -> {
            try {
                for (int i = 0; i < 5; i++) {
                    sharedResource.produceData(i);
                    Thread.sleep(1000);  // Simulate some time taken to produce
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        });

        // Consumer thread
        Thread consumer = new Thread(() -> {
            try {
                for (int i = 0; i < 5; i++) {
                    sharedResource.consumeData();
```

```
                Thread.sleep(1500);  // Simulate some time taken to consume
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    });

    producer.start();
    consumer.start();
    System.out.println("aksha 1bm23cs019");
  }
}
```

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");
    }

    // Now acquire the lock on B after the lock on A
    synchronized (b) {
      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");
        }

        // Now acquire the lock on A after the lock on B
        synchronized (a) {
            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();
    }
}
```