VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

K C SAI NITHIN(1BM23CS130)

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 **K C SAI NITHIN(1BM23CS130),** 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.

Dr. Seema patil Assistant Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

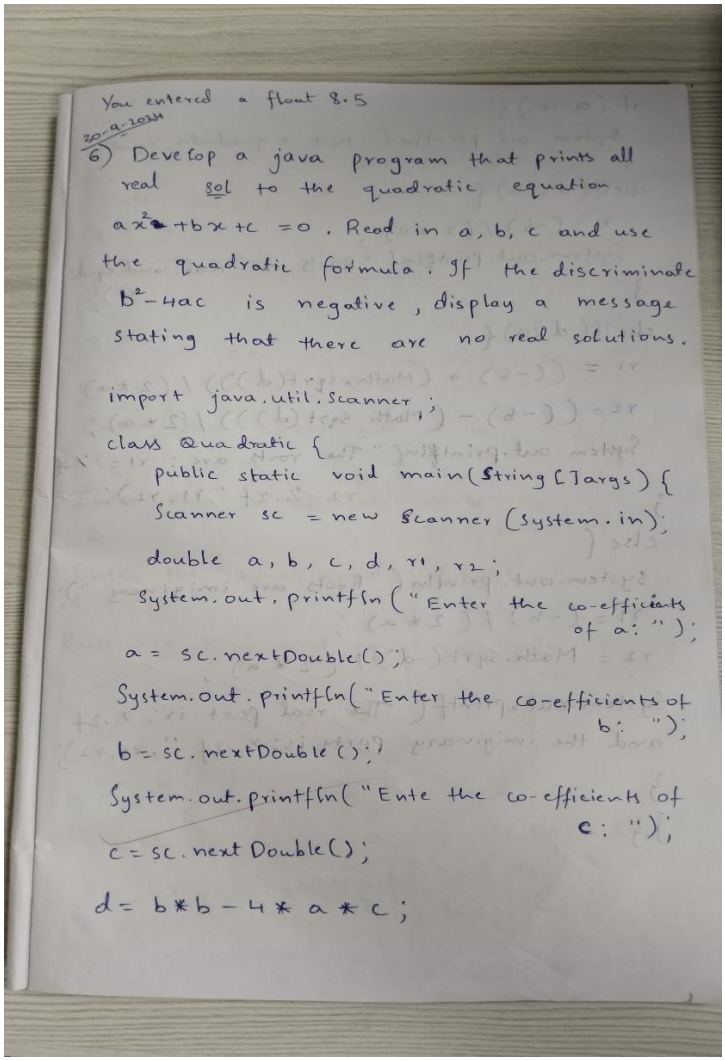
Index

Github Link:
https://github.com/KCSAINITHIN/OOJ-LAB-PROGRAMS

Program 1

Implement Quadratic Equation

Algorithm:

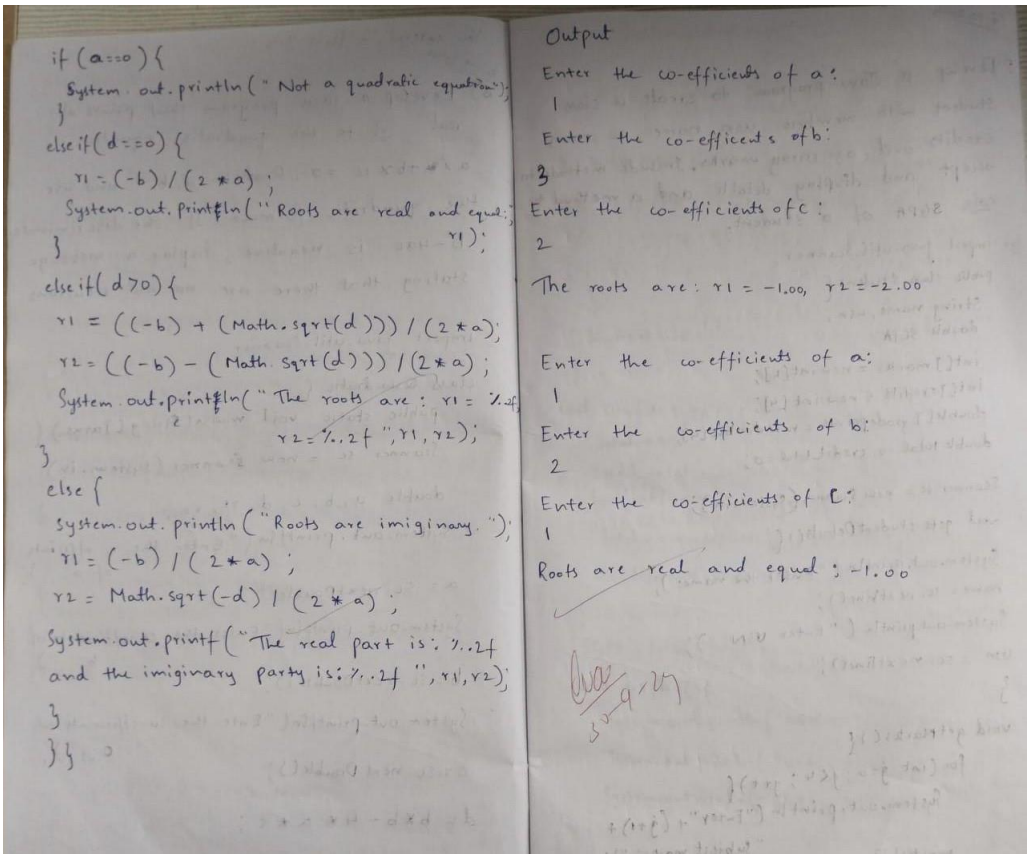


You entered a float 8.5

20-9-2024

6) Develop a java program that prints all real sol 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 {
    public static void main(String[]args) {
    Scanner sc = new Scanner(System.in);
    double a, b, c, d, r1, r2;
    System.out.printfln("Enter the co-efficients
                                of a: ");
    a = sc.nextDouble();
    System.out.printfln("Enter the co-efficients of
                                b: ");
    b = sc.nextDouble();
    System.out.printfln("Ente the co-efficients of
                                c: ");
    c = sc.next Double();
    d = b*b - 4*a*c;
```



Code:

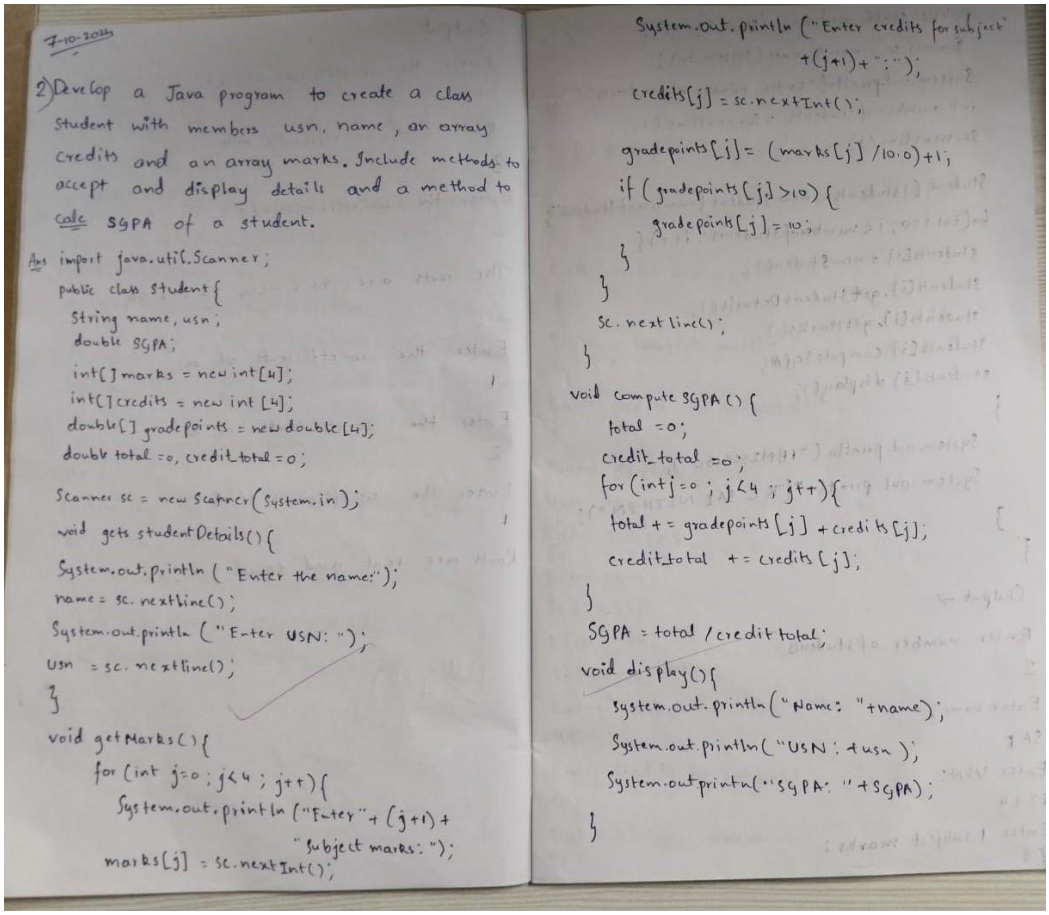
```
import java.util.Scanner;

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

      double a, b, c, d, r1, r2;
      System.out.println("Enter the coefficient a: ");
      a = sc.nextDouble();
      System.out.println("Enter the coefficient b: ");
      b = sc.nextDouble();
      System.out.println("Enter the coefficient c: ");
      c = sc.nextDouble();

      d = b * b - 4 * a * c;

      if (a == 0) {
         System.out.println("Not a quadratic equation!");
      } else if (d == 0) {
         r1 = (-b) / (2 * a);
```

```
            System.out.printf("Roots are real and equal: %.2f%n", r1);
        } else if (d > 0) {
            r1 = ((-b) + (Math.sqrt(d))) / (2 * a);
            r2 = ((-b) - (Math.sqrt(d))) / (2 * a);
            System.out.printf("The roots are: r1 = %.2f, r2 = %.2f%n", r1, r2);
        } else {  // d < 0
            System.out.println("Roots are imaginary.");
            r1 = (-b) / (2 * a);
            r2 = Math.sqrt(-d) / (2 * a);
            System.out.printf("The real part is: %.2f and the imaginary part is: %.2f i%n", r1, r2);          }

        sc.close();      }
}
```

Program 2

Calculation of CGPA of the students

Algorithm:



7-10-2024

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 calc SGPA of a student.

```
Ans import java.util.Scanner;
  public class Student{
    String name, usn;
    double SGPA;
    int[] marks = new int[4];
    int[] credits = new int[4];
    double[] gradepoints = new double[4];
    double total=0, credit_total=0;
    Scanner sc = new Scanner(System.in);
    void getsStudentDetails(){
    System.out.println("Enter the name:");
    name = sc.nextline();
    System.out.println("Enter USN: ");
    usn = sc.nextline();
    }
    void getMarks(){
        for(int j=0; j<4; j++){
          System.out.println("Enter"+(j+1)+
                             "Subject marks:");
        marks[j] = sc.nextInt();
        System.out.println("Enter credits for subject"
                           +(j+1)+":");
        credits[j] = sc.nextInt();
        gradepoints[j] = (marks[j]/10.0)+1;
        if(gradepoints[j]>10){
          gradepoints[j] = 10;
        }
      }
      sc.nextline();
    }
    void computeSGPA(){
      total=0;
      credit_total=0;
      for(int j=0; j<4; j++){
        total += gradepoints[j] * credits[j];
        credit_total += credits[j];
      }
      SGPA = total/credit_total;
    void display(){
      System.out.println("Name: "+name);
      System.out.println("USN: "+usn);
      System.out.println("SGPA: "+SGPA);
    }
```

```
    public static void main (String [] args) {
     Scanner sc = new Scanner (System.in);
    System.out.println("Enter number of students: ");
    int numberofStudents = sc.nextline ();
    sc.nextline();

    Student [] students = new Student [numberofStudents];
    for(int i=0; i<numberofStudents; i++){
     students[i] = new Studen();
     Student[i].getStudentDetails();
     students[i].getMarks();
     Students[i].computeSGPA;
     students[i].display();
    }
      System.out.println ("1BM23CS130");
      System.out.println ("K C SAI NITHIN");
    }
}
```

Output →

```
Enter number of students:
2
Enter name:
SAI
Enter USN:
1234
Enter 1 subject marks:
78
Enter credits for subject 1:
4
Enter 2 subject marks:
87
Enter credits for subject 2:
3
Enter 3 subject marks:
67
Enter credits for subject 3:
3
Enter 4 subjects marks:
94
Enter credits for subject 4:
3
Name: SAI
USN: 1234
SGPA: 9.0307
Enter name:
NITHIN
Enter USN:
12345
Enter 1 subject marks:
98
Enter credits for subject 1:
4
Enter 2 subject marks:
97
Enter credits for subject 2:
3
Enter 3 subject marks:
78
Enter credits for subject 3:
3
Enter 4 subjects marks:
80
Enter credits for subject 4:
3
Name: NITHIN
USN: 12345
SGPA: 9.42307
```

Code:

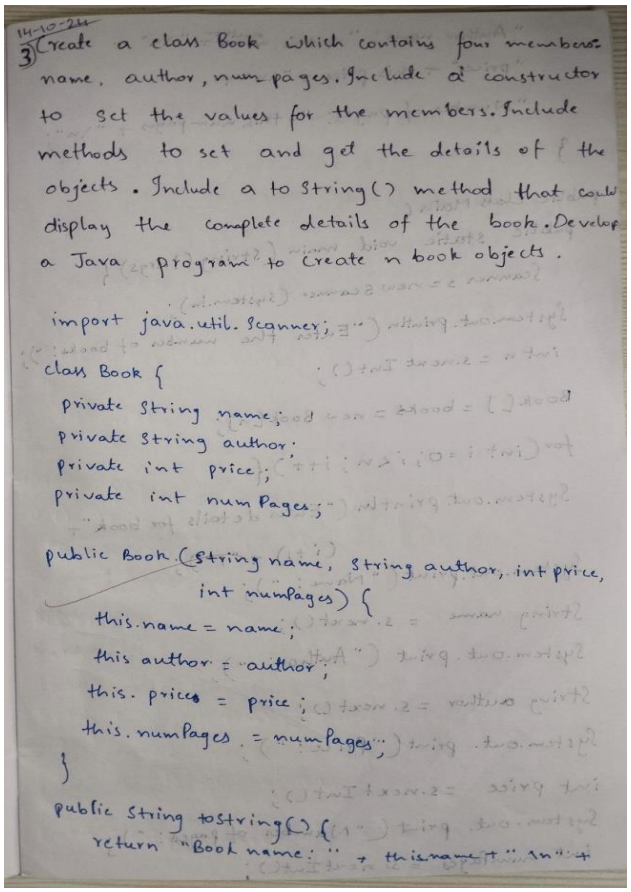
```
import java.util.Scanner;

public class Student {
    String name, usn;
    double SGPA;

    int[] marks = new int[4];
    int[] credits = new int[4];
    double[] gradepoints = new double[4];
    double total = 0, credittotal = 0;

    Scanner sc = new Scanner(System.in);

    void getStudentDetails() {
        System.out.println("Enter name:");
        name = sc.nextLine();
        System.out.println("Enter USN:");
        usn = sc.nextLine();
    }

    void getMarks() {
        for (int j = 0; j < 4; j++) {
            System.out.println("Enter " + (j + 1) + " subject marks:");
            marks[j] = sc.nextInt();
            System.out.println("Enter credits for subject " + (j + 1) + ":");
            credits[j] = sc.nextInt();

            gradepoints[j] = (marks[j] / 10.0) + 1;
            if (gradepoints[j] > 10) {
                gradepoints[j] = 10;
            }
        }
        sc.nextLine();
    }

    void computeSGPA() {
        total = 0;
        credittotal = 0;
        for (int j = 0; j < 4; j++) {
            total += gradepoints[j] * credits[j];
            credittotal += credits[j];
        }
        SGPA = total / credittotal;
```

```
    }

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

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

        Student[] students = new Student[numberOfStudents];

        for (int i = 0; i < numberOfStudents; i++) {
            students[i] = new Student();
            students[i].getStudentDetails();
            students[i].getMarks();
            students[i].computeSGPA();
            students[i].display();
        }
                    System.out.println("1BM23CS130");
                System.out.println("K C SAI NITHIN");

    }
}
```

Program 3

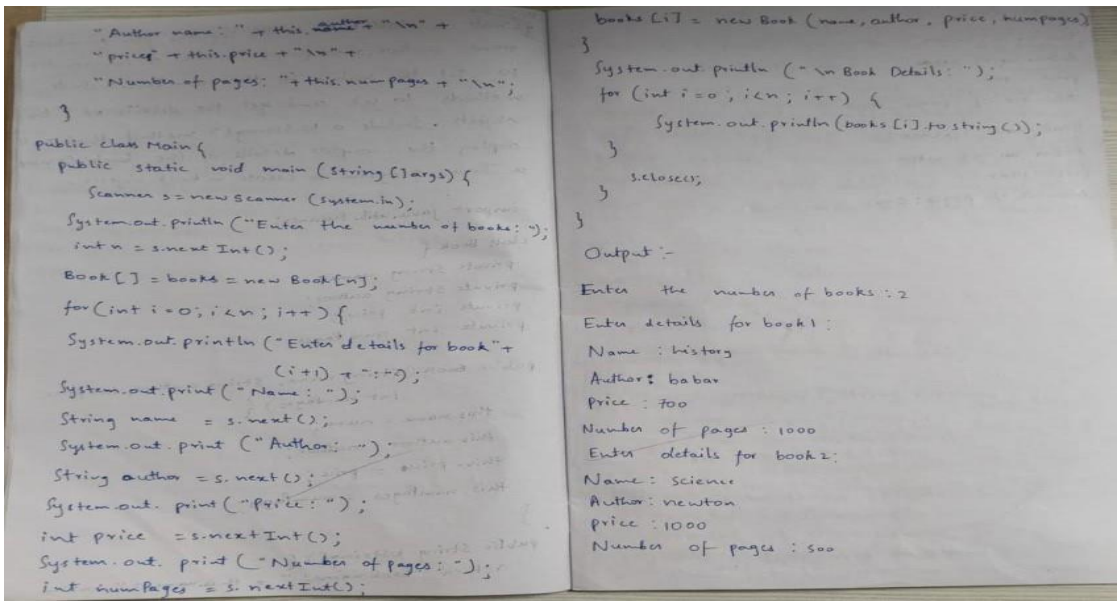
Two Strings

Algorithm:



3) Create a class Book which contains four members: name, author, 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 to String() method that could display the complete details of the book. Develop a Java program to create n book objects.

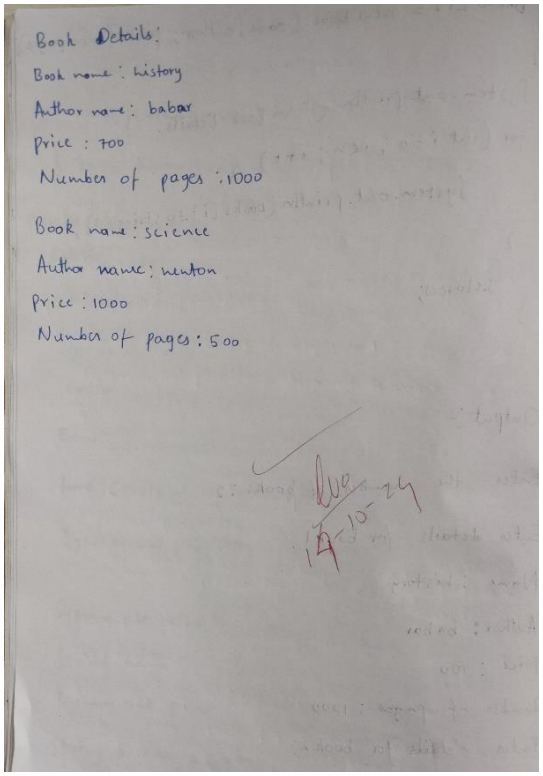
```
import java.util.Scanner;
class Book {
 private String name;
 private String author;
 private int price;
 private int numPages;

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



```
    "Author name: " + this.author + "\n" +
    "price" + this.price + "\n" +
    "Number of pages: " + this.numPages + "\n";
 }
public class Main {
  public static void main (String[] args) {
    Scanner s = new Scanner (System.in);
    System.out.println ("Enter the number of books: ");
    int n = s.nextInt();
    Book[] = books = new Book[n];
    for (int i=0; i<n; i++) {
    System.out.println ("Enter details for book" +
                        (i+1) + ":");
    System.out.print ("Name: ");
    String name = s.next();
    System.out.print ("Author: ");
    String author = s.next();
    System.out.print ("Price: ");
    int price = s.nextInt();
    System.out.print ("Number of pages: ");
    int numPages = s.nextInt();
    books[i] = new Book (name, author, price, numpages);
  }
    System.out.println ("\n Book Details: ");
    for (int i=0; i<n; i++) {
        System.out.println (books[i].toString());
    }
    s.close();
  }
}
```

Output:-

```
Enter the number of books: 2
Enter details for book1:
Name: history
Author: babar
Price: 700
Number of pages: 1000
Enter details for book2:
Name: science
Author: newton
price: 1000
Number of pages: 500
```



Code:

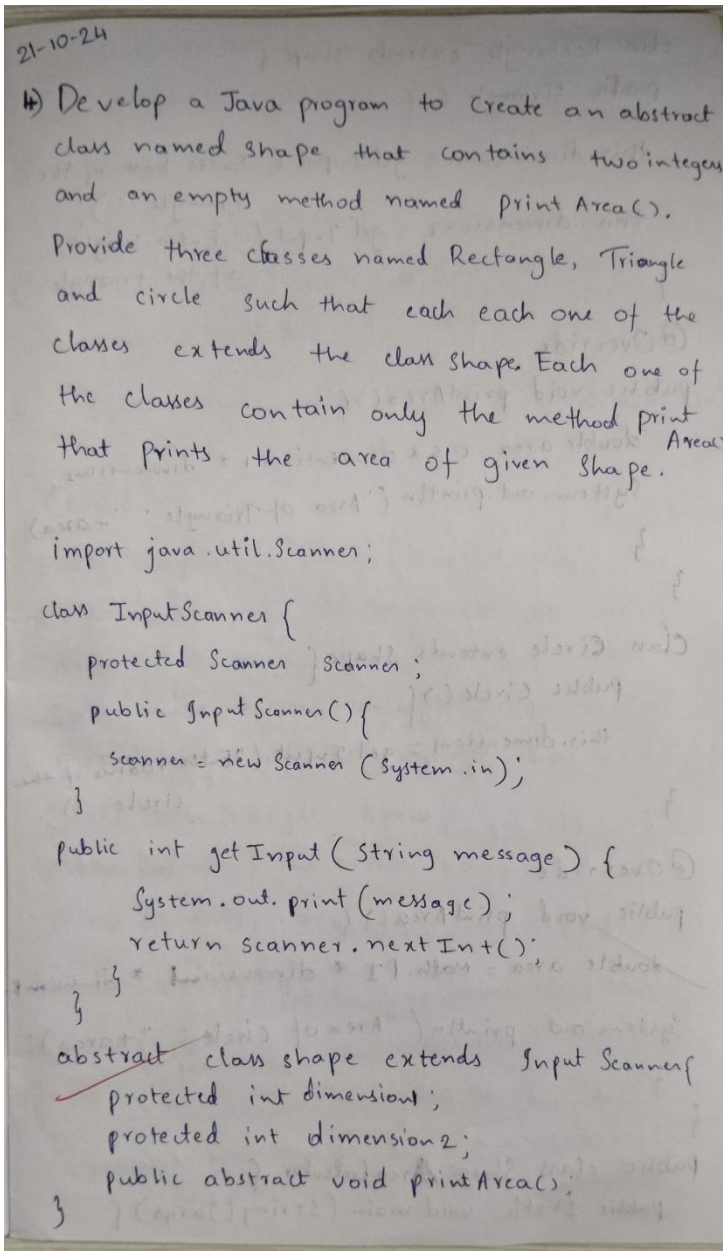
```
import java.util.Scanner;

class Book {


private String name;

private String author;

private int price; private

int numPages;


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

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

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

        System.out.print("Enter the number of books: ");
        int n = Integer.parseInt(s.nextLine());

        Book[] books = new Book[n];

        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for book " + (i + 1) + ":");
            System.out.print("Name: ");
            String name = s.nextLine();
            System.out.print("Author: ");
            String author = s.nextLine();
            System.out.print("Price: ");
            int price = Integer.parseInt(s.nextLine());
            System.out.print("Number of pages: ");
            int numPages = Integer.parseInt(s.nextLine());

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

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

Program 4

Shape Area calculator

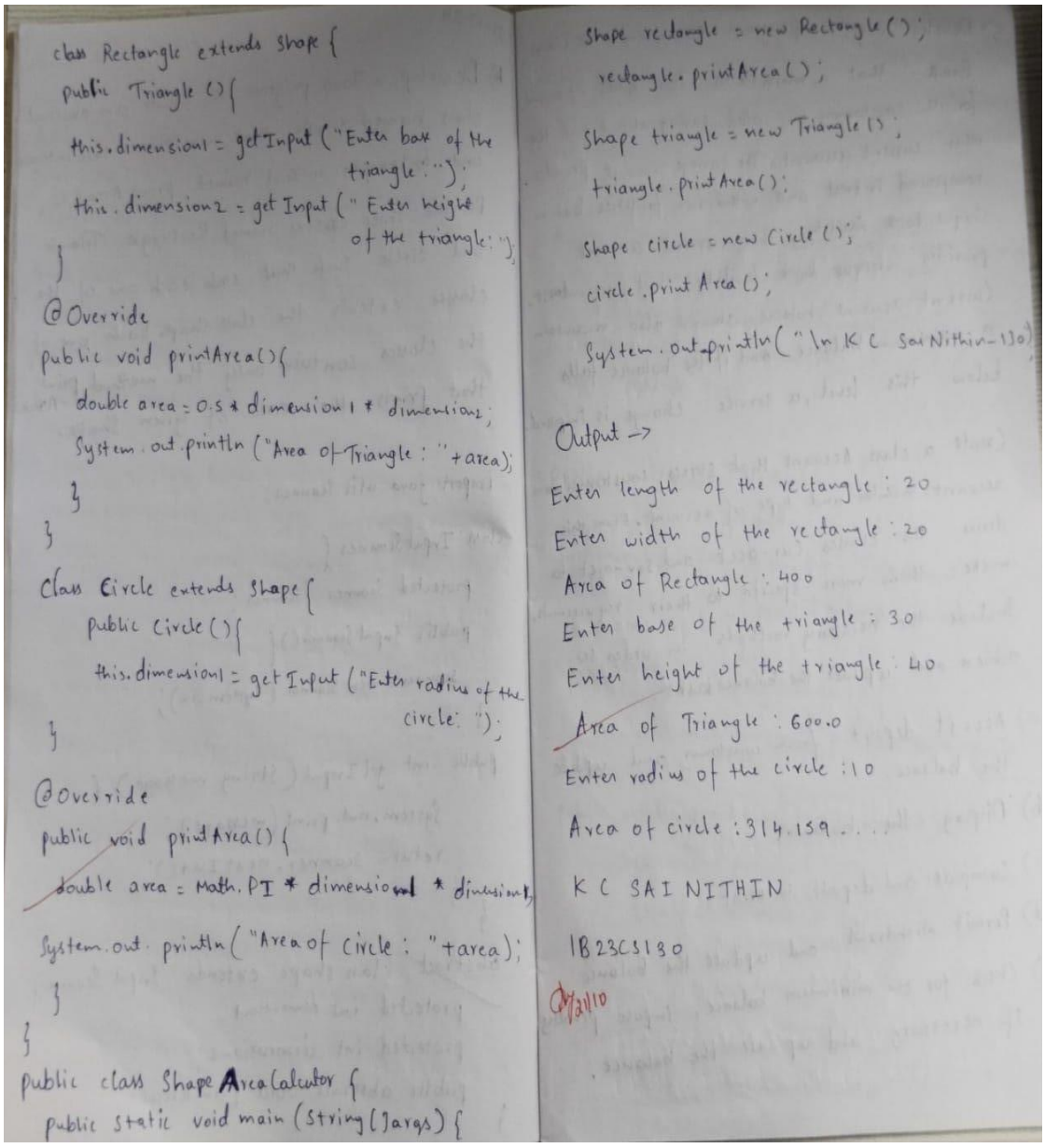
Algorithm:



21-10-24

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 each one of the classes extends the class Shape. Each one of the classes contain only the method printArea that prints the area of given Shape.

```
import java.util.Scanner;

class InputScanner {
    protected Scanner scanner;
    public InputScanner(){
        scanner = new Scanner (System.in);
    }
    public int getInput (String message) {
        System.out.print(message);
        return scanner.nextInt();
    }
}
abstract class shape extends InputScanner{
    protected int dimension1;
    protected int dimension2;
    public abstract void printArea();
}
```



```
class Rectangle extends Shape {
public Triangle (){
this.dimension1 = getInput ("Enter base of the
                  triangle: ");
this.dimension2 = getInput (" Enter height
                  of the triangle: ");
}
@Override
public void printArea(){
double area = 0.5 * dimension1 * dimension2;
System.out.println ("Area of Triangle: " +area);
}
}
Class Circle extends Shape{
public Circle (){
this.dimension1 = getInput ("Enter radius of the
                  circle: ");
}
@Override
public void printArea() {
double area = Math.PI * dimension1 * dimension1;
System.out.println("Area of circle: "+area);
}
}
public class ShapeAreaCalculator {
public static void main (String[]args) {
Shape rectangle = new Rectangle();
rectangle.printArea();
Shape triangle = new Triangle();
triangle.printArea();
Shape circle = new Circle();
circle.printArea();
System.out.println("In K C Sai Nithin-1130");
```

Output ->

```
Enter length of the rectangle: 20
Enter width of the rectangle: 20
Area of Rectangle: 400
Enter base of the triangle: 30
Enter height of the triangle: 40
Area of Triangle: 600.0
Enter radius of the circle: 10
Area of circle: 314.159...
K C SAI NITHIN
1B23CS130
```

24/10

Code:

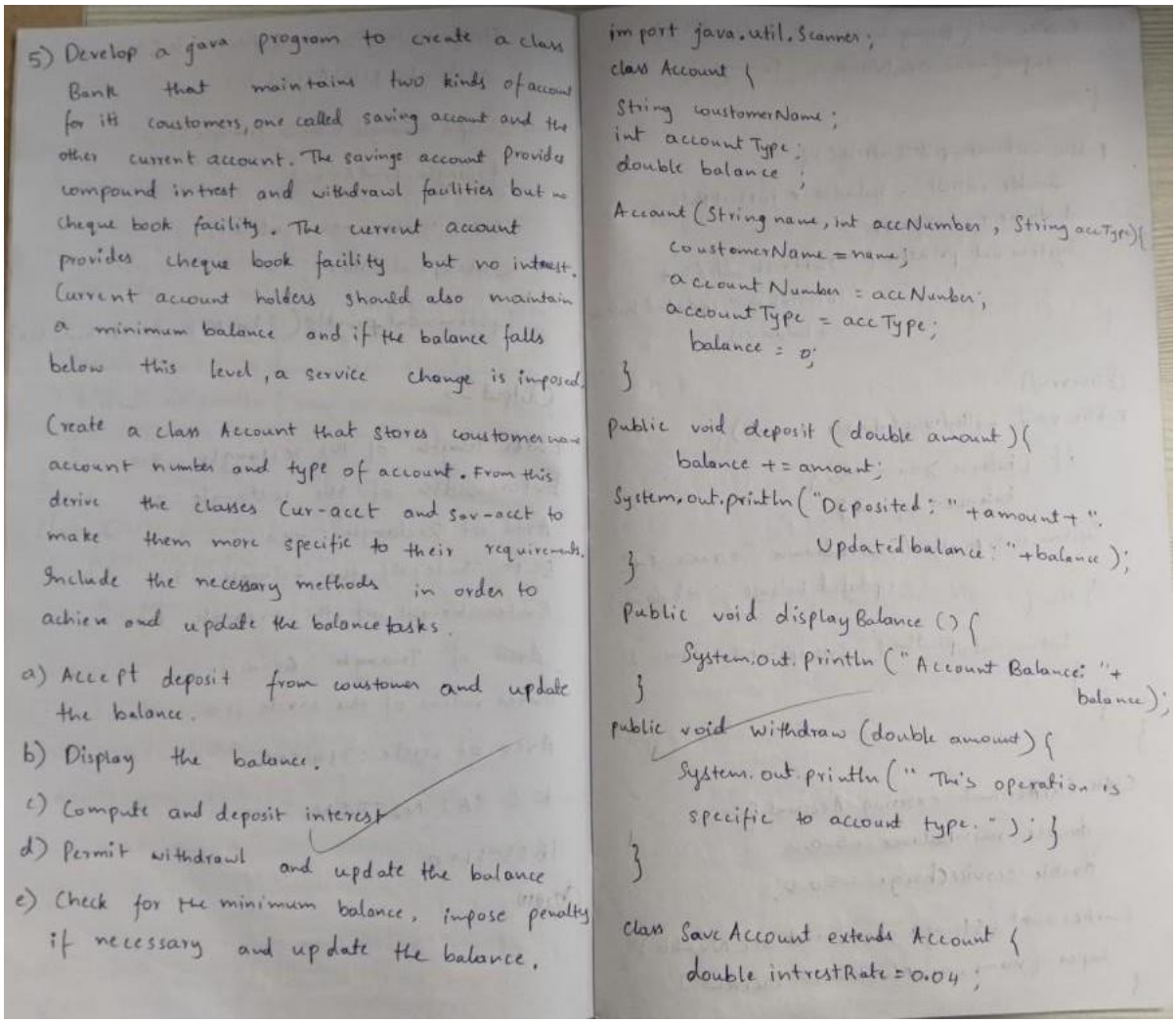
```
import java.util.Scanner;

class InputScanner {
    protected Scanner scanner;

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

    public int getInput(String message) {
        System.out.print(message);
        return scanner.nextInt();
    }
}


abstract class Shape extends InputScanner {
    protected int dimension1;
    protected int dimension2;

    public abstract void printArea();
}

class Rectangle extends Shape {
    public Rectangle() {
        this.dimension1 = getInput("Enter length of the rectangle: ");
        this.dimension2 = getInput("Enter width of the rectangle: ");
    }

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


class Triangle extends Shape {
    public Triangle() {
        this.dimension1 = getInput("Enter base of the triangle: ");
        this.dimension2 = getInput("Enter height of the triangle: ");
    }

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

```
    }
}


class Circle extends Shape {
    public Circle() {
        this.dimension1 = getInput("Enter radius of the circle: ");
    }

    public void printArea() {
        double area = Math.PI * dimension1 * dimension1;
        System.out.println("Area of Circle: " + area);
    }
}


public class ShapeAreaCalculator {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle();
        rectangle.printArea();

        Shape triangle = new Triangle();
        triangle.printArea();

        Shape circle = new Circle();
        circle.printArea();
    }
}
```

Program 5

Bank account

Algorithm:



5) 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 savings account provides compound intrest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no intrest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service change 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 and update the balance tasks.

a) Accept deposit from customer and update the balance.

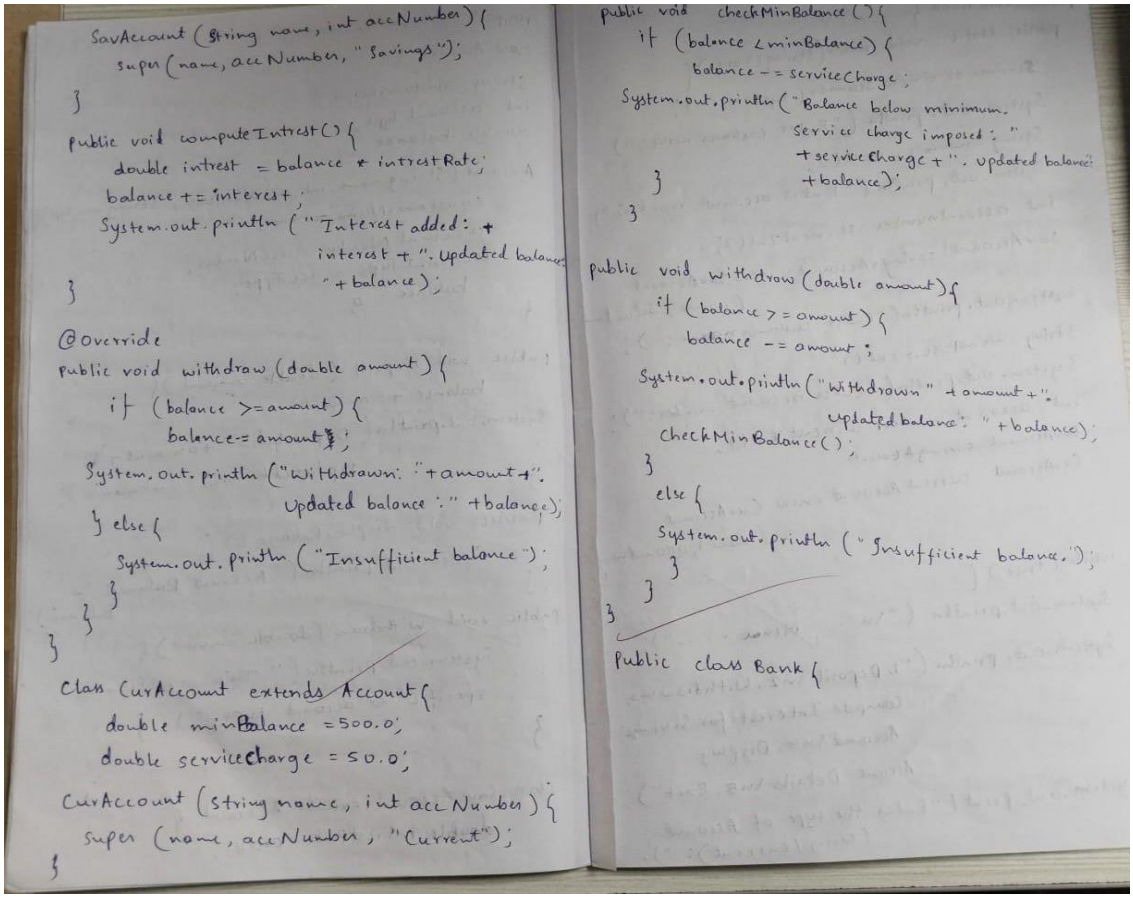
b) Display the balance.

c) Compute and deposit interest

d) Permit withdrawl and update the balance

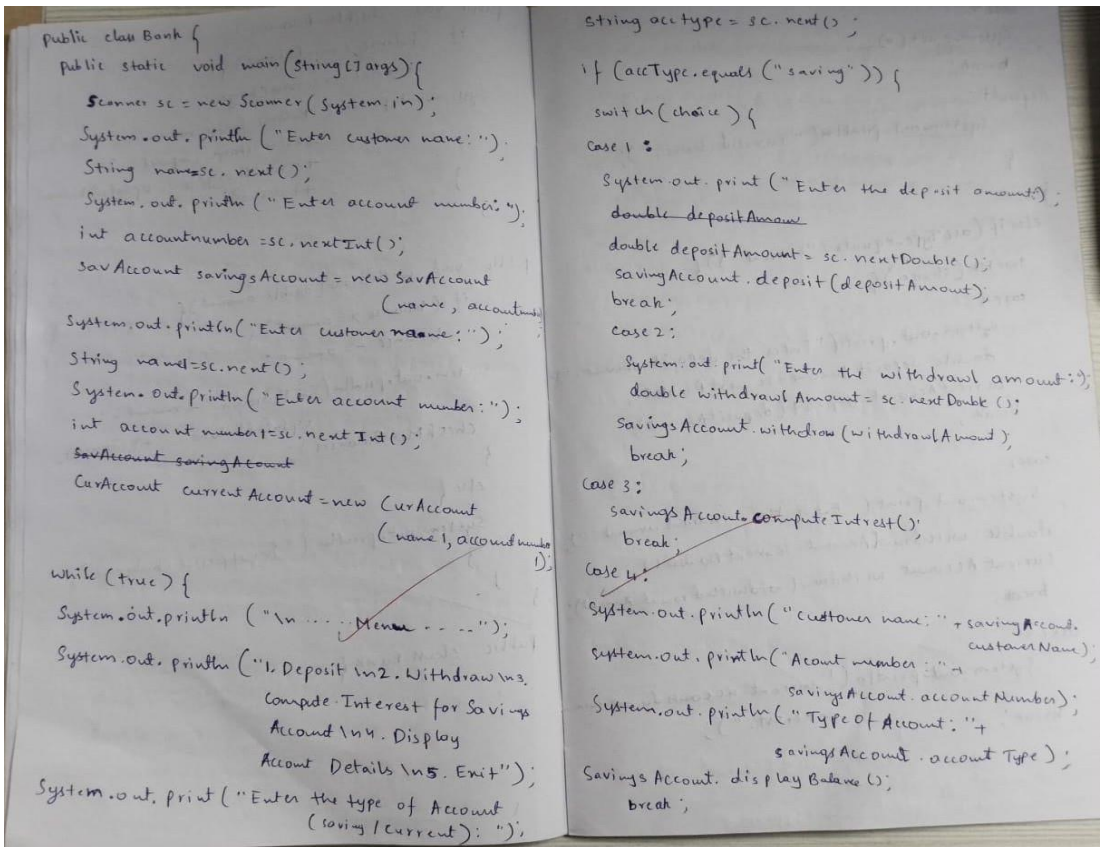
e) Check for the minimum balance, impose penalty if necessary and update the balance.

```
import java.util.Scanner;
class Account {
String customerName;
int accountType;
double balance;
Account(String name, int accNumber, String accType){
    customerName = name;
    accountNumber = accNumber;
    accountType = accType;
    balance = 0;
}
public void deposit(double amount){
    balance += amount;
System.out.println("Deposited: "+amount+".
                    Updated balance: "+balance);
}
Public void displayBalance(){
    System.out.println("Account Balance: "+
                                    balance);
}
public void withdraw(double amount){
    System.out.println("This operation is
    specific to account type."); }
}
class SaveAccount extends Account {
    double intrestRate = 0.04;
```



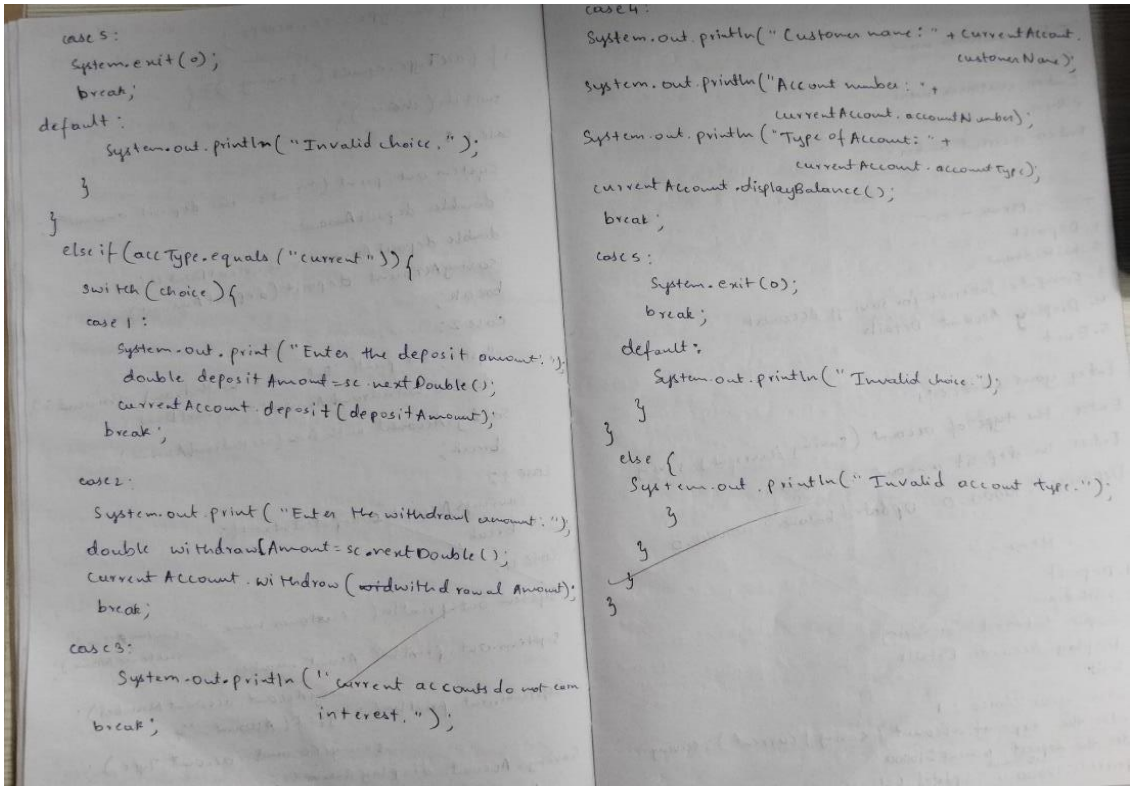
```
    SavAccount (String name, int accNumber) {
        super (name, accNumber, "Savings");
    }

    public void computeIntrest() {
        double intrest = balance * intrestRate;
        balance += interest;
        System.out.println ("Interest added: +
                            interest + ". Updated balance
                            " + balance);
    }

    @Override
    public void withdraw (double amount) {
        if (balance >= amount) {
            balance -= amount;
        System.out.println ("Withdrawn: " + amount + ".
                            Updated balance :" + balance);
        } else {
            System.out.println ("Insufficient balance");
        }
    }
}

Class CurAccount extends Account {
    double minBalance = 500.0;
    double serviceCharge = 50.0;
    CurAccount (String name, int accNumber) {
        super (name, accNumber, "Current");
}
```

```
public void checkMinBalance () {
    if (balance < minBalance) {
        balance -= serviceCharge;
    System.out.println ("Balance below minimum.
                        Service charge imposed: "
                        + serviceCharge + ". Updated balance:
                        + balance);
    }
}

public void withdraw (double amount) {
    if (balance >= amount) {
        balance -= amount;
    System.out.println ("Withdrawn " + amount + ".
                        Updated balance: " + balance);
    checkMinBalance();
    }
    else {
    System.out.println ("Insufficient balance.");
    }
}
}

Public class Bank {
```



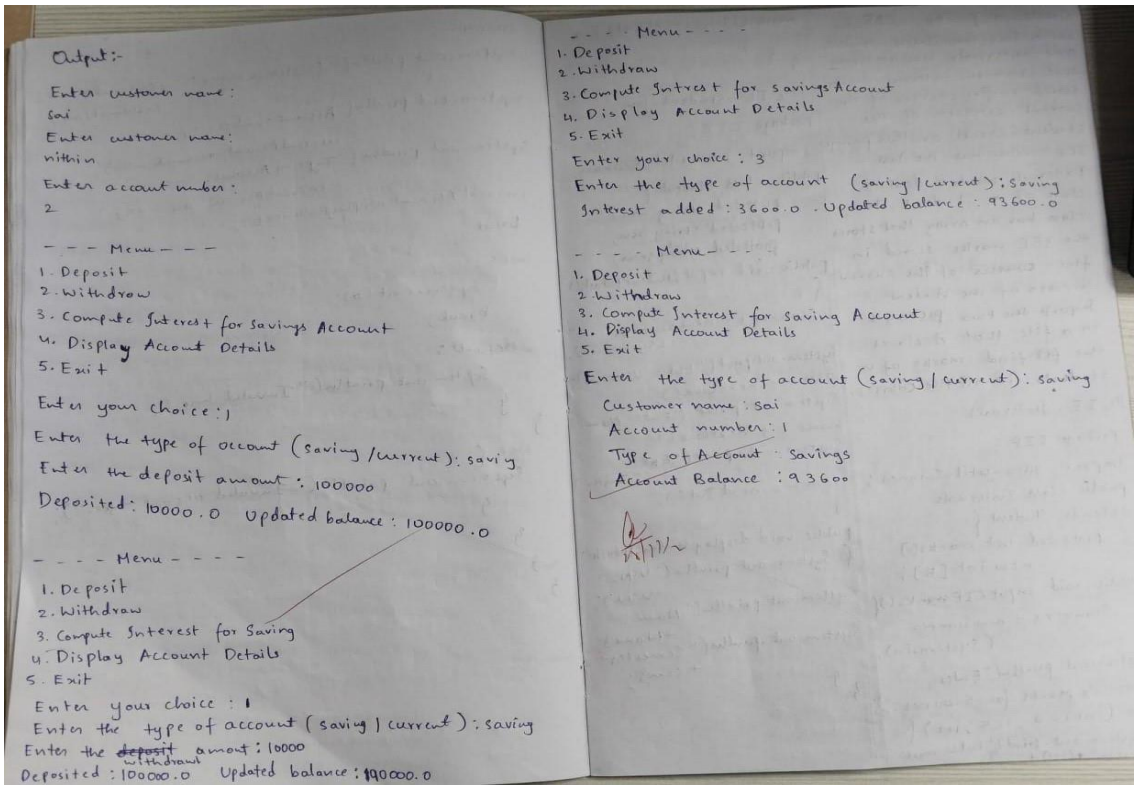
```
Public class Bank {
    public static void main (String[] args) {
        Scanner sc = new Scanner (System.in);
        System.out.println ("Enter customer name: ");
        String name=sc.next();
        System.out.println ("Enter account number: ");
        int accountnumber =sc.nextInt();
        SavAccount savingsAccount = new SavAccount
                                    (name, accountnumber);
        System.out.println("Enter customer name: ");
        String name1=sc.next();
        System.out.println("Enter account number: ");
        int accountnumber1=sc.nextInt();
        SavAccount savingAcount
        CurAccount currentAccount = new CurAccount
                                    (name1, accountnumber1);
        while (true) {
        System.out.println ("\n ... Menu ... ");
        System.out.println ("1. Deposit \n2. Withdraw \n3.
                            Compute Interest for Savings
                            Account \n4. Display
                            Account Details \n5. Exit");
        System.out.print ("Enter the type of Account
                            (saving / current): ");
```

```
String acctype = sc.next();
if (acctype.equals ("saving")) {
    switch (choice) {
    case 1 :
        System.out.print ("Enter the deposit amount");
        double depositAmount
        double depositAmount = sc.nextDouble();
        savingAccount.deposit (depositAmount);
        break;
        case 2:
        System.out.print("Enter the withdrawl amount:");
        double withdrawlAmount = sc.nextDouble();
        SavingsAccount.withdraw (withdrawlAmount);
        break;
    case 3:
        savingsAccount.computeIntrest();
        break;
    case 4:
        System.out.println("Customer name: " + savingAccount.
                            customerName);
        System.out.println("Acount number: " +
                            savingsAccount.accountNumber);
        System.out.println("Type of Account: " +
                            savingsAccount.accountType);
        SavingsAccount.displayBalance();
        break;
```



```
case 5:
    System.exit(0);
    break;
default:
    System.out.println("Invalid choice.");
  }
}
else if (accType.equals("current")) {
  switch (choice) {
  case 1:
    System.out.print("Enter the deposit amount: ");
    double depositAmount = sc.nextDouble();
    currentAccount.deposit(depositAmount);
    break;

  case 2:
    System.out.print("Enter the withdrawal amount: ");
    double withdrawalAmount = sc.nextDouble();
    currentAccount.withdraw(withdrawalAmount);
    break;

  case 3:
    System.out.println("current accounts do not earn
                        interest.");
    break;
```

```
case 4:
  System.out.println("Customer name: " + currentAccount.
                                          customerName);
  System.out.println("Account number: " +
                          currentAccount.accountNumber);
  System.out.println("Type of Account: " +
                          currentAccount.accountType());
  currentAccount.displayBalance();
  break;

  case 5:
    System.exit(0);
    break;
  default:
    System.out.println("Invalid choice.");
  }
}
else {
  System.out.println("Invalid account type.");
  }
}
```



Output:-

```
Enter customer name:
Sai
Enter customer name:
within
Enter account number:
2

--- Menu ---
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit

Enter your choice: 1
Enter the type of account (saving/current): saving
Enter the deposit amount: 100000
Deposited: 10000.0  Updated balance: 100000.0

---- Menu ----
1. Deposit
2. Withdraw
3. Compute Interest for Saving
4. Display Account Details
5. Exit
Enter your choice: 1
Enter the type of account (saving/current): saving
Enter the deposit/withdrawal amount: 10000
Deposited: 100000.0  Updated balance: 190000.0
```

```
---- Menu ----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 3
Enter the type of account (saving/current): Saving
Interest added: 3600.0. Updated balance: 93600.0

---- Menu ---
1. Deposit
2. Withdraw
3. Compute Interest for saving Account
4. Display Account Details
5. Exit
Enter the type of account (saving/current): saving
  Customer name: Sai
  Account number: 1
  Type of Account: Savings
  Account Balance: 93600
```

Code:

```
import java.util.Scanner;

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


    Account(String name, int accNumber, String accType) {
        customerName = name;
        accountNumber = accNumber;
        accountType = accType;
        balance = 0;
    }


    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: " + amount + ". Updated balance: " + balance);
    }

    public void displayBalance() {
        System.out.println("Account Balance: " + balance);
    }


    public void withdraw(double amount) {
        System.out.println("This operation is specific to account type.");
    }
}


class SavAccount extends Account {
    double interestRate = 0.04;

    SavAccount(String name, int accNumber) {
        super(name, accNumber, "Savings");
    }


    public void computeInterest() {
        double interest = balance * interestRate;
        balance += interest;
        System.out.println("Interest added: " + interest + ". Updated balance: " + balance);
```

```
    }


    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ". Updated balance: " + balance);
        } else {
            System.out.println("Insufficient balance.");
        }
    }
}
class CurAccount extends Account {
    double minBalance = 500.0;
    double serviceCharge = 50.0;

    CurAccount(String name, int accNumber) {
        super(name, accNumber, "Current");
    }

    public void checkMinBalance() {
        if (balance < minBalance) {
            balance -= serviceCharge;
            System.out.println("Balance below minimum. Service charge imposed: " + serviceCharge + ".
Updated balance: " + balance);
        }
    }


    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ". Updated balance: " + balance);
            checkMinBalance();
        } else {
            System.out.println("Insufficient 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.next();
        System.out.println("Enter account number:");
        int accountnumber=sc.nextInt();
```

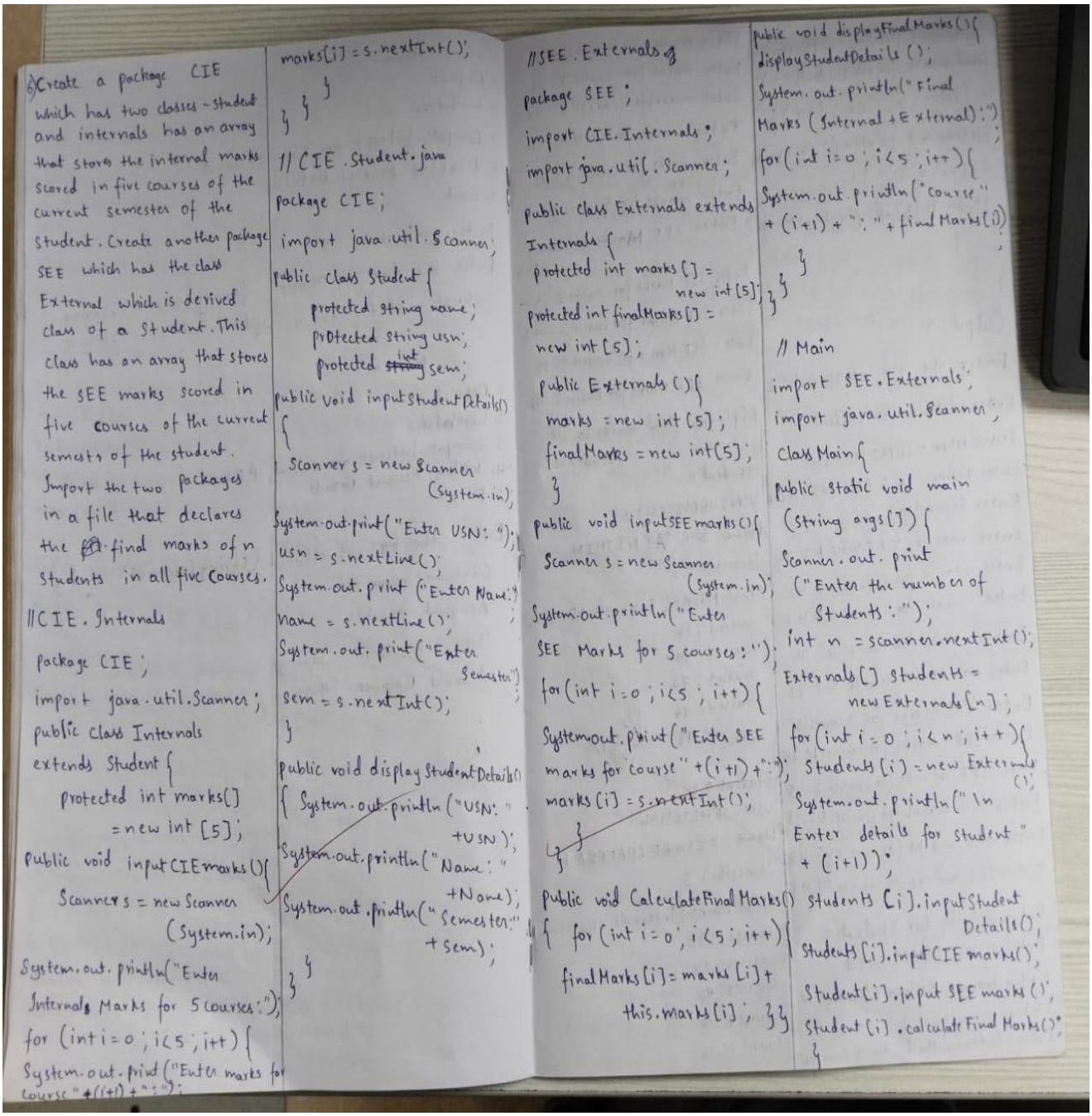
```
        SavAccount savingsAccount = new SavAccount(name, accountnumber);
System.out.println("Enter customer name:");
        String name1=sc.next();
        System.out.println("Enter account number:");
        int accountnumber1=sc.nextInt();
        CurAccount currentAccount = new CurAccount(name1, accountnumber1);


        while (true) {
            System.out.println("\n-----MENU ----- ");
            System.out.println("1. Deposit\n2. Withdraw\n3. Compute Interest for Savings Account\n4.
Display Account Details\n5. Exit");
            System.out.print("Enter your choice: ");
            int choice = sc.nextInt();

            System.out.print("Enter the type of account (saving/current): ");
            String accType = sc.next();

            if (accType.equals("saving")) {
                switch (choice) {
                    case 1:
                        System.out.print("Enter the deposit amount: ");
                        double depositAmount = sc.nextDouble();
                        savingsAccount.deposit(depositAmount);
                        break;
                    case 2:
                        System.out.print("Enter the withdrawal amount: ");
                        double withdrawalAmount = sc.nextDouble();
                        savingsAccount.withdraw(withdrawalAmount);
                        break;
                    case 3:
                        savingsAccount.computeInterest();
                        break;
                    case 4:
                        System.out.println("Customer name: " + savingsAccount.customerName);
                        System.out.println("Account number: " + savingsAccount.accountNumber);
                        System.out.println("Type of Account: " + savingsAccount.accountType);
                        savingsAccount.displayBalance();
                        break;
                    case 5:
                        System.exit(0);
                        break;
                    default:
                        System.out.println("Invalid choice.");
                }
            } else if (accType.equals("current")) {
                switch (choice) {
```

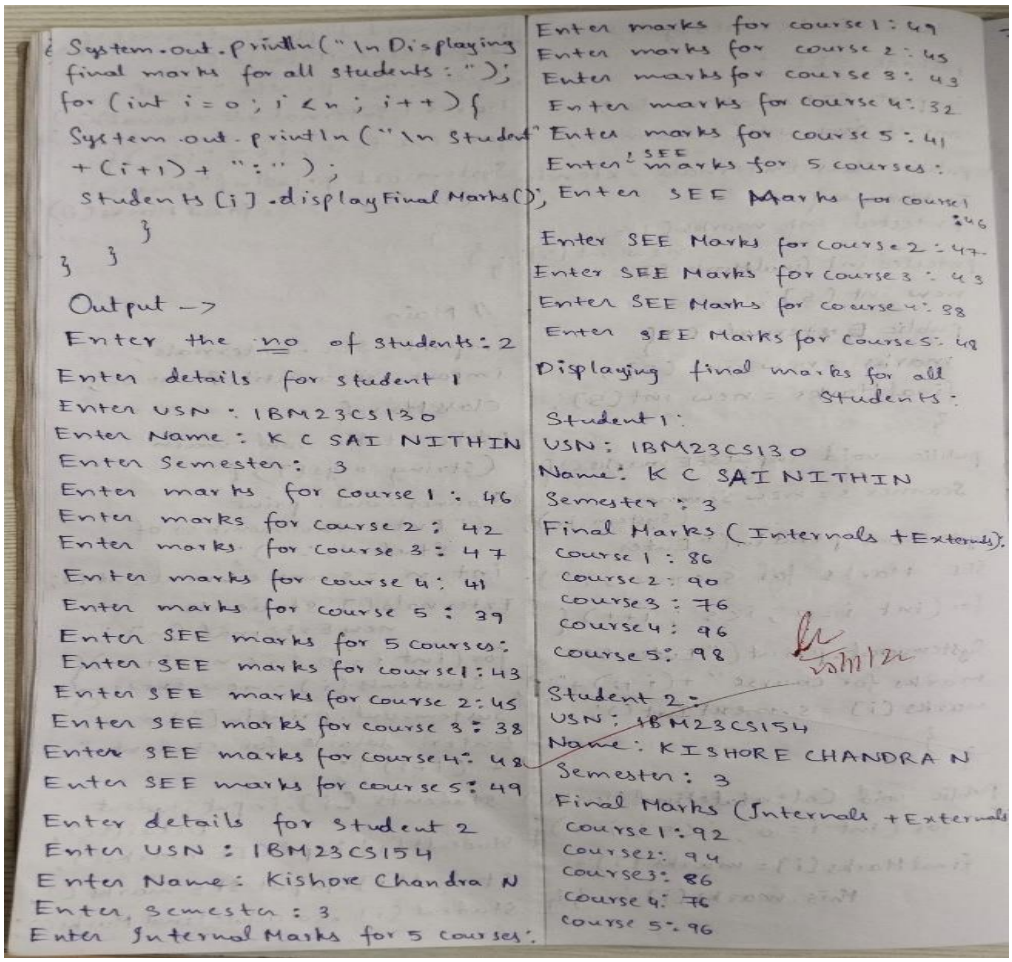
```
                case 1:
                    System.out.print("Enter the deposit amount: ");
                    double depositAmount = sc.nextDouble();
                    currentAccount.deposit(depositAmount);
                    break;
                case 2:
                    System.out.print("Enter the withdrawal amount: ");
                    double withdrawalAmount = sc.nextDouble();
                    currentAccount.withdraw(withdrawalAmount);
                    break;
                case 3:
                    System.out.println("Current accounts do not earn interest.");
                    break;
                case 4:
                    System.out.println("Customer name: " + currentAccount.customerName);
                    System.out.println("Account number: " + currentAccount.accountNumber);
                    System.out.println("Type of Account: " + currentAccount.accountType);
                    currentAccount.displayBalance();
                    break;
                case 5:
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid choice.");
            }
        } else {
            System.out.println("Invalid account type.");
        }
    }
  }
}
```

Program 6

Package CIE SEE

Algorithm:





Code:

CIE/
Inetrnals-

```
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 5 courses:");
      for (int i = 0; i < 5; i++) {
         System.out.print("Enter marks for course " + (i + 1) + ": ");
         marks[i] = s.nextInt();
      }
   }
```

```
}
```

Student-

```
package CIE;

import java.util.Scanner;

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

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

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

SEE/

Externals-

```
package SEE;

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

public class Externals extends Internals {
    protected int marks[] = new int[5];
    protected int finalMarks[] = new int[5];

    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 5 courses:");
```

```
        for (int i = 0; i < 5; i++) {
            System.out.print("Enter SEE marks for course " + (i + 1) + ": ");
            marks[i] = s.nextInt();
        }
    }

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

    public void displayFinalMarks() {
        displayStudentDetails();
        System.out.println("Final Marks (Internal + External):");
        for (int i = 0; i < 5; i++) {
            System.out.println("Course " + (i + 1) + ": " + finalMarks[i]);
        }
    }
}
```

CIE SEE/
Main-

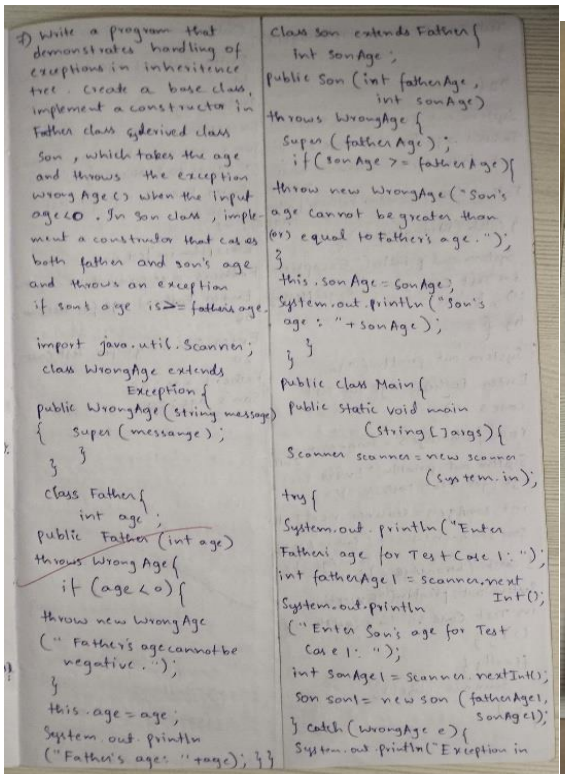
```
import SEE.Externals;
import java.util.Scanner;

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

        Externals[] students = new Externals[n];

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


            students[i].inputStudentDetails();

            students[i].inputCIEmarks();

            students[i].inputSEEmarks();

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

```
        System.out.println("\nDisplaying final marks for all students:");
        for (int i = 0; i < n; i++) {
            System.out.println("\nStudent " + (i + 1) + ":");
            students[i].displayFinalMarks();
        }
    }
}
```

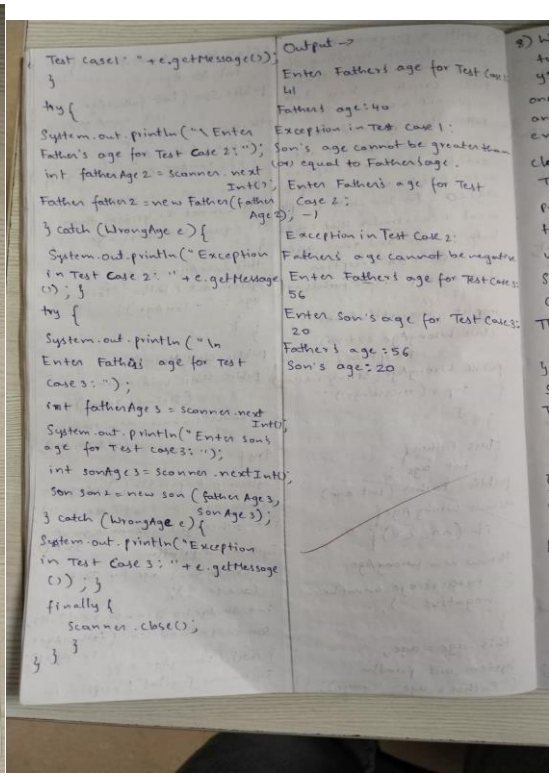
Program 7

Exception Father and Son's ages

Algorithm:



```
1) Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class, implement a constructor in Father class & derived class Son, which takes the age and throws the exception WrongAge() when the input age<0. In Son class, implement a constructor that takes 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 Father{
    int age;
public Father(int age)
throws WrongAge{
    if (age<0){
throw new WrongAge
("Father's age cannot be negative.");
}
this.age=age;
System.out.println
("Father's age: "+age);}}

class Son extends Father{
    int sonAge;
public Son(int fatherAge,
            int sonAge)
throws WrongAge {
super(fatherAge);
if(sonAge>=fatherAge){
throw new WrongAge("Son's age cannot be greater than (or) equal to Father's age.");
}
this.sonAge=sonAge;
System.out.println("Son's age : "+sonAge);
}}
public class Main{
public static void main
        (String[]args){
Scanner scanner=new Scanner
                (System.in);
try{
System.out.println("Enter Father's age for Test Case 1: ");
int fatherAge1 = scanner.nextInt();
System.out.println
("Enter Son's age for Test Case 1: ");
int sonAge1 = scanner.nextInt();
Son son1= new son (fatherAge1, sonAge1);
} catch(WrongAge e){
System.out.println("Exception in
```



```
Test case1: "+e.getMessage());
}
try{
System.out.println("\nEnter Father's age for Test case 2:");
int fatherAge2 = scanner.nextInt();
Father father2 = new Father(fatherAge2);
} catch (WrongAge e){
System.out.println("Exception in Test Case 2: "+e.getMessage());}
try{
System.out.println("\nEnter Father's age for Test case 3: ");
int fatherAge3 = scanner.nextInt();
System.out.println("Enter son's age for Test case 3: ");
int sonAge3 = scanner.nextInt();
Son son2 = new son (fatherAge3, sonAge3);
} catch (WrongAge e){
System.out.println("Exception in Test Case 3: "+e.getMessage());}
finally{
    scanner.close();
}}

Output->
Enter Father's age for Test Case 1:
41
Father's age: 40
Exception in Test case 1:
Son's age cannot be greater than (or) equal to Father's age.
Enter Father's age for Test Case 2:
-1
Exception in Test Case 2:
Father's age cannot be negative
Enter Father's age for Test Case 3:
56
Enter Son's age for Test Case 3:
20
Father's age: 56
Son's age: 20
```

Code:

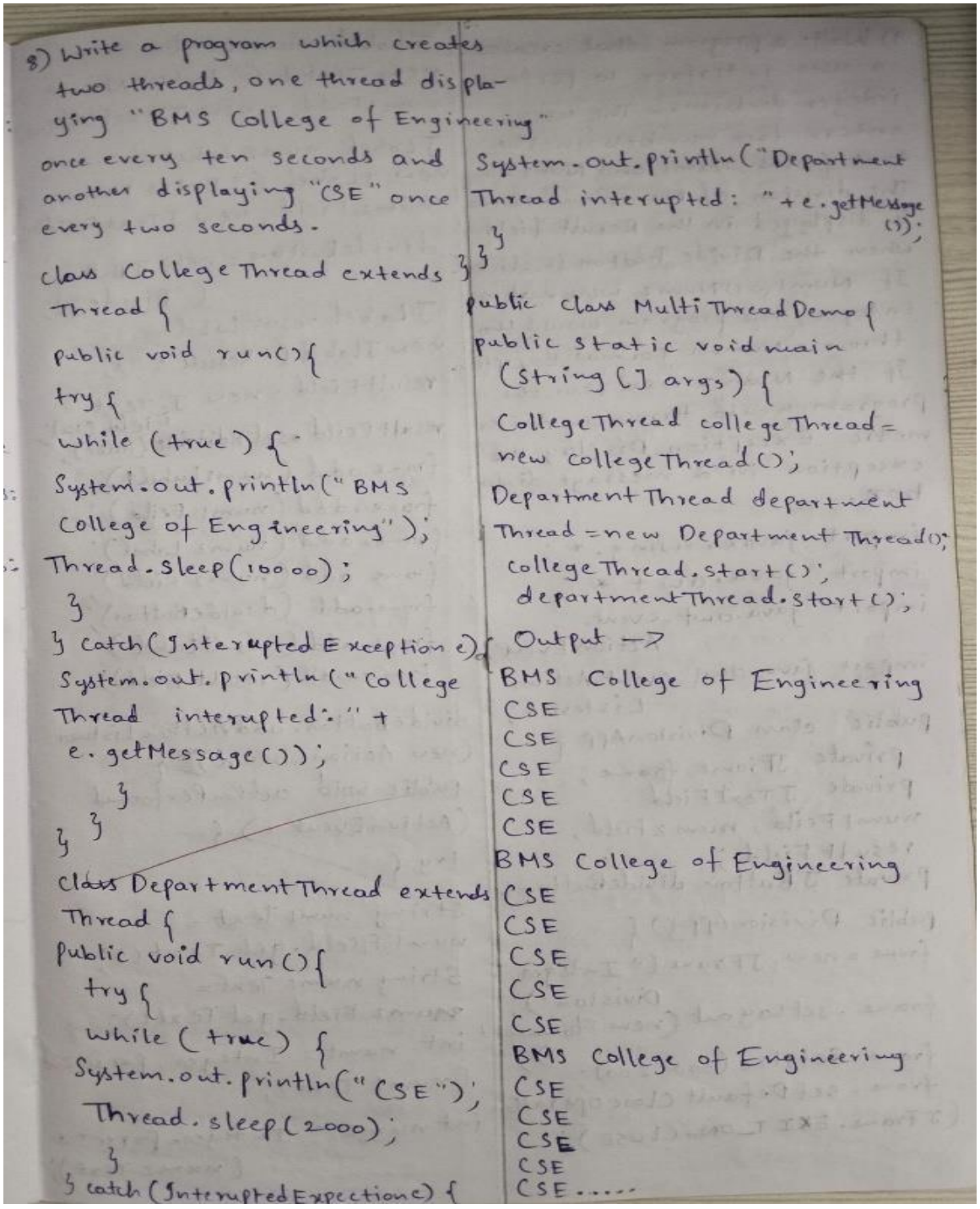
```
import java.util.Scanner;

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

class Father {
    int age;

    public Father(int age) throws WrongAge {
        if (age < 0) {
            throw new WrongAge("Father's age cannot be negative.");
        }
        this.age = age;
        System.out.println("Father's age: " + age);
    }
}

class Son extends Father {
    int sonAge;

    public Son(int fatherAge, int sonAge) throws WrongAge {
        super(fatherAge);
        if (sonAge >= fatherAge) {
            throw new WrongAge("Son's age cannot be greater than or equal to Father's age.");
        }
        this.sonAge = sonAge;
        System.out.println("Son's age: " + sonAge);
    }
}

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

        try {
            System.out.println("Enter Father's age for Test Case 1:");
            int fatherAge1 = scanner.nextInt();
            System.out.println("Enter Son's age for Test Case 1:");
            int sonAge1 = scanner.nextInt();
            Son son1 = new Son(fatherAge1, sonAge1);
            } catch (WrongAge e) {
            System.out.println("Exception in Test Case 1: " + e.getMessage());
        }
```

```
        try {
            System.out.println("\nEnter Father's age for Test Case 2:");
            int fatherAge2 = scanner.nextInt();
            Father father2 = new Father(fatherAge2);
            } catch (WrongAge e) {
            System.out.println("Exception in Test Case 2: " + e.getMessage());
        }
         try {
            System.out.println("\nEnter Father's age for Test Case 3:");
            int fatherAge3 = scanner.nextInt();
            System.out.println("Enter Son's age for Test Case 3:");
            int sonAge3 = scanner.nextInt();
            Son son2 = new Son(fatherAge3, sonAge3);
            } catch (WrongAge e) {
            System.out.println("Exception in Test Case 3: " + e.getMessage());
            } finally {
            scanner.close();
        }
    }
}
```

Program 8

Multithreading

Algorithm



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 CollegeThread extends Thread {
public void run(){
try {
while (true) {
System.out.println("BMS College of Engineering");
Thread.sleep(10000);
}
} catch(InterruptedException e){
System.out.println("College Thread interupted:" + e.getMessage());
}
}

class DepartmentThread extends Thread {
Public void run(){
  try {
  while (true) {
 System.out.println("CSE");
  Thread.sleep(2000);
  }
} catch (InterruptedExpectione) {
System.out.println("Department Thread interupted: "+e.getMessage());
}
}

public class MultiThreadDemo {
public static void main (String [] args) {
CollegeThread collegeThread = new collegeThread();
DepartmentThread department Thread =new DepartmentThread();
 collegeThread.start();
 departmentThread.start();
```

Output →

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

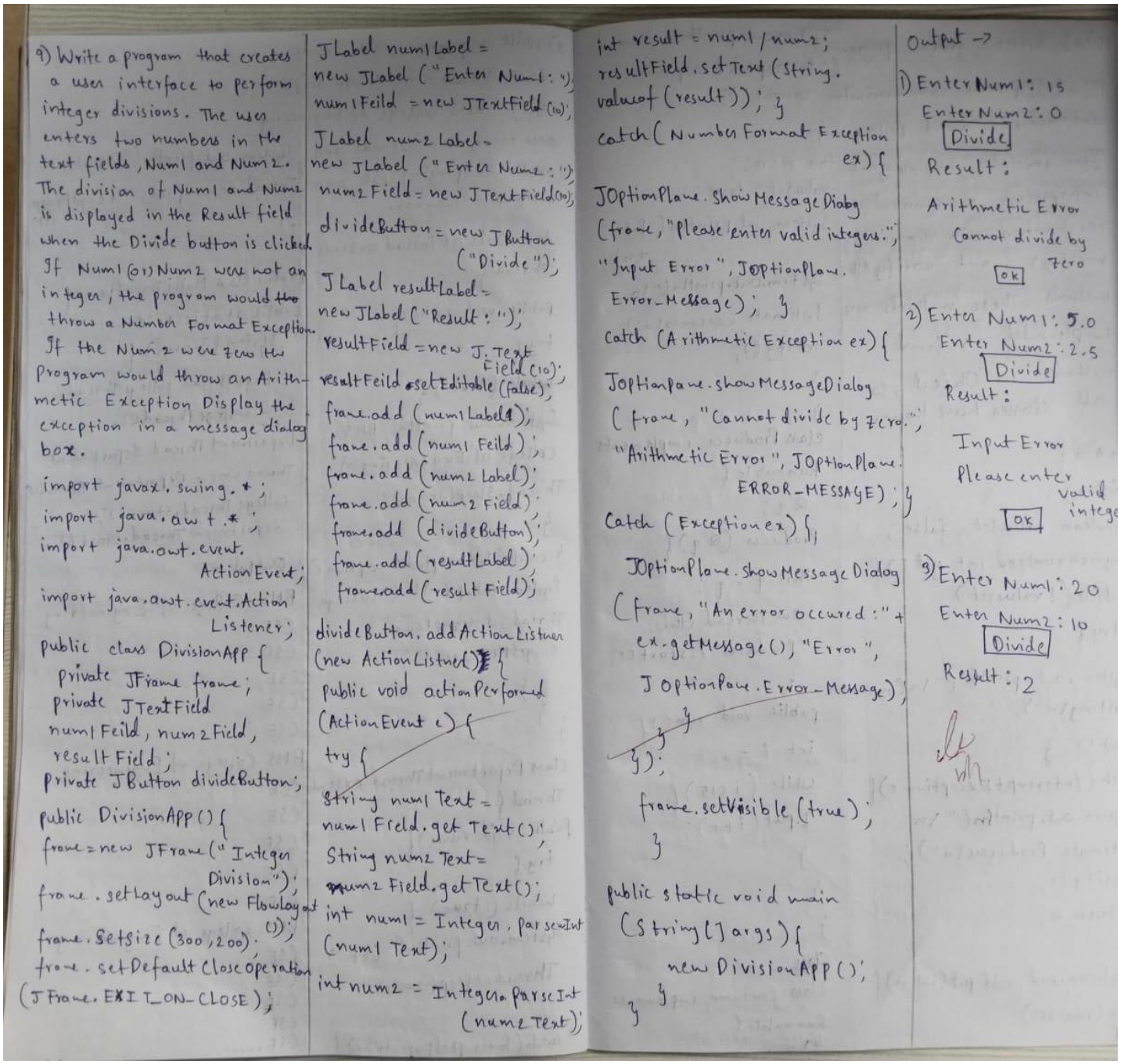
Code:

```
class CollegeThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("BMS College of Engineering");
                Thread.sleep(10000); // Pause for 10 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("CollegeThread interrupted: " + e.getMessage());
        }
    }
}


class DepartmentThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("CSE");
            }
        } catch (InterruptedException e) {
            System.out.println("DepartmentThread interrupted: " + e.getMessage());
        }
    }
}


public class MultiThreadDemo {
    public static void main(String[] args) {
        CollegeThread collegeThread = new CollegeThread();
        DepartmentThread departmentThread = new DepartmentThread();
        collegeThread.start();
        departmentThread.start();
    }
}
```

Program 9

Division app

Algorithm:



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 ~~the~~ throw a NumberFormatException. If the Num2 were zero the program would throw an ArithmeticException Display the exception in a message dialog box.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.
                ActionEvent;
import java.awt.event.Action
                Listener;
public class DivisionApp {
  private JFrame frame;
  private JTextField
  num1Feild, num2Field,
  resultField;
private JButton divideButton;
public DivisionApp(){
frame=new JFrame("Integer
                Division");
frame.setLayout(new FlowLayout
                ());
frame.setSize(300,200);
frame.setDefaultCloseOperation
(JFrame.EXIT_ON_CLOSE);
```

```
JLabel num1Label =
new JLabel("Enter Num1:");
num1Feild = new JTextField(10);
JLabel num2Label=
new JLabel("Enter Num2:");
num2Field = new JTextField(10);
divideButton = new JButton
                ("Divide");
JLabel resultLabel=
new JLabel("Result:");
resultFeild = new J.Text
                Field(10);
resultFeild.setEditable(false);
frame.add(num1Label);
frame.add(num1Feild);
frame.add(num2Label);
frame.add(num2Field);
frame.add(divideButton);
frame.add(resultLabel);
frame.add(resultField);
divideButton.addActionListner
(new ActionListner(){
public void actionPerformed
(ActionEvent e){
try{
String num1Text=
num1Field.getText();
String num2Text=
num2Field.getText();
int num1 = Integer.parseInt
(num1Text);
int num2 = Integer.parseInt
                (num2Text);
```

```
int result = num1/num2;
resultField.setText(String.
valueof(result));}
catch(NumberFormatException
                ex){
JOptionPlane.showMessageDiabg
(frame,"Please enter valid integers.",
"Input Error",JOptionPlane.
Error_Message);}
Catch(ArithmeticException ex){
JOptionPane.showMessageDialog
(frame,"Cannot divide by zero.",
"Arithmetic Error",JOptionPlane.
                ERROR_MESSAGE);}
Catch(Exception ex){
JOptionPlane.showMessageDialog
(frame,"An error occured:"+
ex.getMessage(),"Error",
JOptionPlane.Error_Message);
});
frame.setVisible(true);
}
public static void main
(String[]args){
    new DivisionApp();
}
}
```

Output →

1) Enter Num1: 15
Enter Num2: 0
[Divide]
Result:
Arithmetic Error
Cannot divide by zero
[ok]

2) Enter Num1: 5.0
Enter Num2: 2.5
[Divide]
Result:
Input Error
Please enter valid integ
[ok]

3) Enter Num1: 20
Enter Num2: 10
[Divide]
Result: 2

Code:

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

public class DivisionApp {


    private JFrame frame;
    private JTextField num1Field, num2Field, resultField;
    private JButton divideButton;

    public DivisionApp() {
        frame = new JFrame("Integer Division");
        frame.setLayout(new FlowLayout());
        frame.setSize(300, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


        JLabel num1Label = new JLabel("Enter Num1: ");
        num1Field = new JTextField(10);

        JLabel num2Label = new JLabel("Enter Num2: ");
        num2Field = new JTextField(10);

        divideButton = new JButton("Divide");

        JLabel resultLabel = new JLabel("Result: ");
        resultField = new JTextField(10);
        resultField.setEditable(false);


        frame.add(num1Label);
        frame.add(num1Field);
        frame.add(num2Label);
        frame.add(num2Field);
        frame.add(divideButton);
        frame.add(resultLabel);
        frame.add(resultField);


        divideButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                try {
```

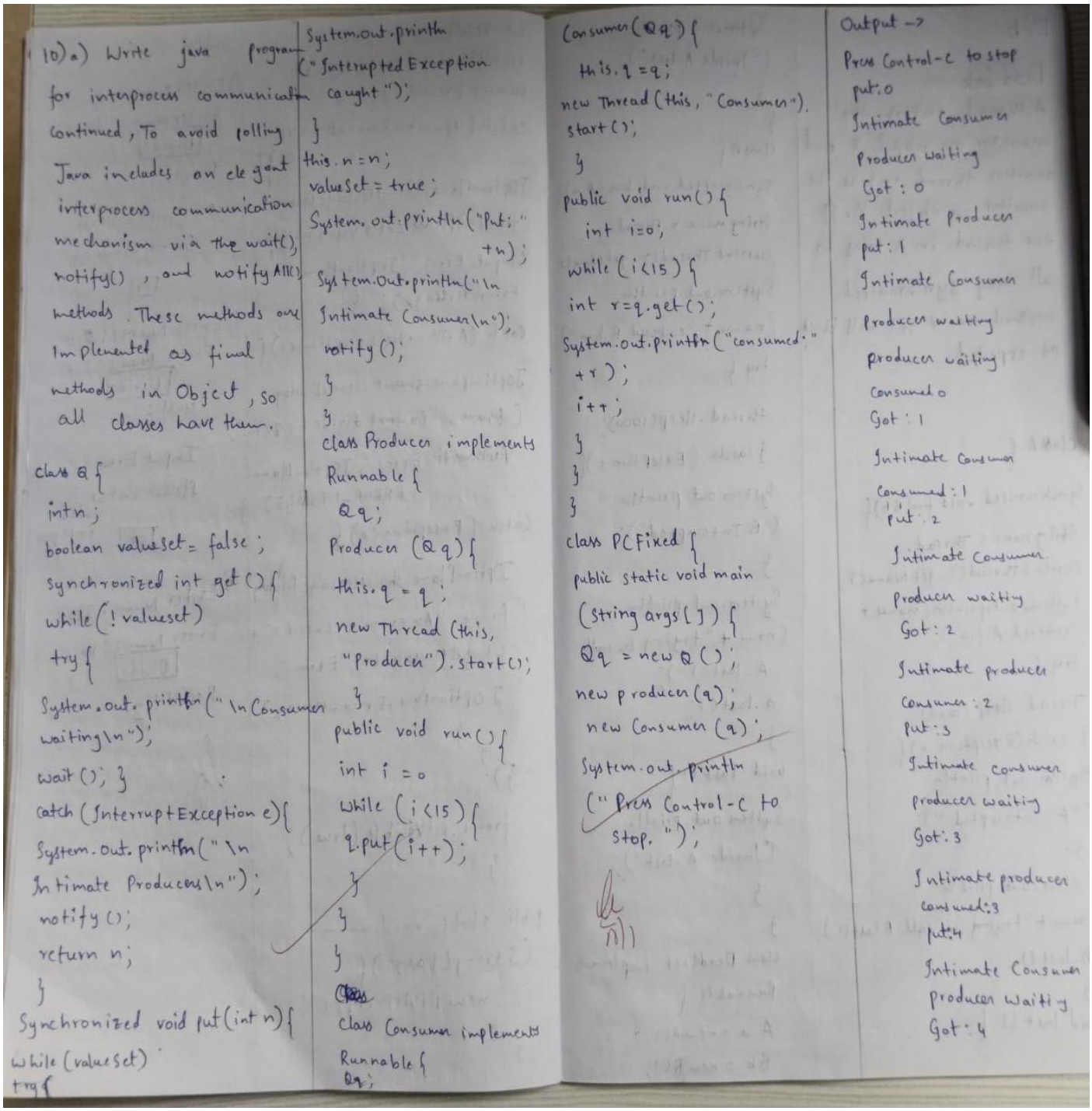
```
                String num1Text = num1Field.getText();
                String num2Text = num2Field.getText();

                int num1 = Integer.parseInt(num1Text);
                int num2 = Integer.parseInt(num2Text);

                int result = num1 / num2;
                resultField.setText(String.valueOf(result));

            } catch (NumberFormatException ex) {

                JOptionPane.showMessageDialog(frame, "Please enter valid integers.",
                                              "Input Error", JOptionPane.ERROR_MESSAGE);
            } catch (ArithmeticException ex) {

                JOptionPane.showMessageDialog(frame, "Cannot divide by zero.",
                                              "Arithmetic Error", JOptionPane.ERROR_MESSAGE);
            } catch (Exception ex) {

                JOptionPane.showMessageDialog(frame, "An error occurred: " + ex.getMessage(),
                                              "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });


    frame.setVisible(true);
  }

  public static void main(String[] args) {

    new DivisionApp();
  }
}
```

Program 10

Division app

a) IPC

Algorithm:



10) a) Write java program for interprocess communication

Continued, To avoid polling Java includes an elegant interprocess communication mechanism via the wait(), notify() , and notifyAll() methods. These methods are implemented as final methods in Object, so all classes have them.

```
class Q {
int n;
boolean valueset = false;
synchronized int get() {
while(!valueset)
try {
System.out.println("\nConsumer waiting\n");
wait(); }
catch (InterruptException e){
System.out.println("\n Intimate Producer\n");
notify();
return n;
}
Synchronized void put(int n){
while (valueset)
try{
System.out.println
("Interupted Exception caught");
}
this.n=n;
valueSet = true;
System.out.println("Put: "
+n);
System.Out.println("\n Intimate Consumer\n");
notify();
}
}
class Producer implements
Runnable {
Q q;
Producer (Q q){
this.q = q;
new Thread (this,
"Producer").start();
}
public void run(){
int i =0
while (i<15){
q.put(i++);
}
}
}
class Consumer implements
Runnable {
Q q;
Consumer(Q q){
this.q =q;
new Thread(this, "Consumer").
start();
}
public void run(){
int i=0;
while (i<15){
int r=q.get();
System.out.println("consumed:"
+r);
i++;
}
}
}
class PCFixed {
public static void main
(String args[]){
Q q = new Q();
new producer(q);
new Consumer(q);
System.out.println
("Press Control-C to
stop.");
}
```

Output ->

```
Press Control-C to stop
put:0
Intimate Consumer
Producer waiting
Got : 0
Intimate Producer
put: 1
Intimate Consumer
Producer waiting
producer waiting
Consumed 0
Got : 1
Intimate Consumer
Consumed : 1
put: 2
Intimate Consumer
Producer waiting
Got: 2
Intimate producer
Consumer : 2
Put: 3
Intimate consumer
producer waiting
Got: 3
Intimate producer
consumed:3
put:4
Intimate Consumer
Producer Waiting
Got: 4
```

Code:

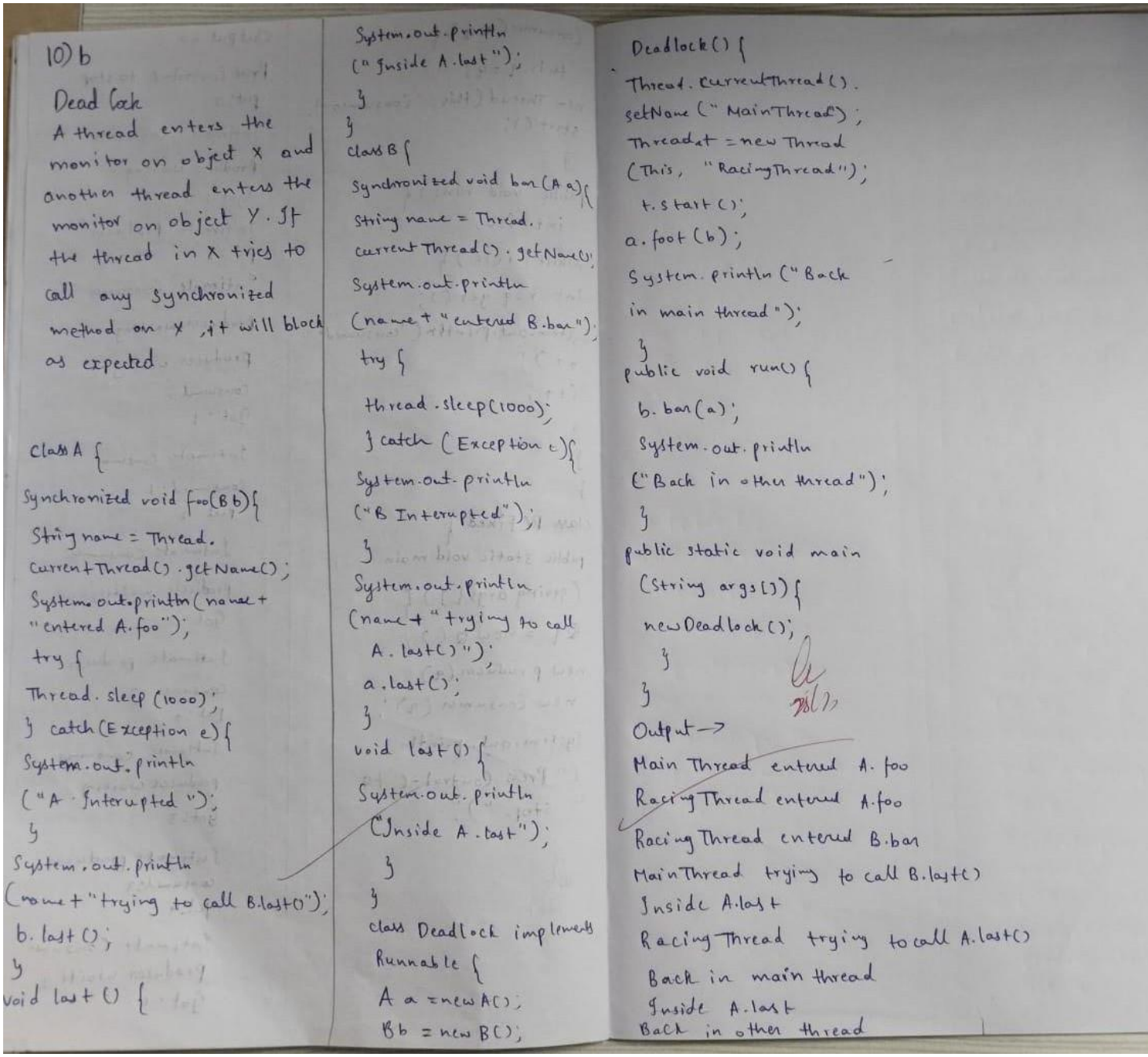
```
class Q {
int n;
boolean valueSet = false;
synchronized int get() {
while(!valueSet)
try {
System.out.println("\nConsumer waiting\n");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
System.out.println("Got: " + n);
valueSet = false;
System.out.println("\nIntimate Producer\n");
notify();
return n;
}
synchronized void put(int n) {
while(valueSet)
try {
System.out.println("\nProducer waiting\n");
wait();
} catch(InterruptedException e) {
```

```
System.out.println("InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("\nIntimate Consumer\n");
notify();
}
}
class Producer implements Runnable {
Q q;
Producer(Q q) {
this.q = q;
new Thread(this, "Producer").start();
}
public void run() {
int i = 0;
while(i<15) {
q.put(i++);
}
}
}
class Consumer implements Runnable {
Q q;
```

```
Consumer(Q q) {
this.q = q;
new Thread(this, "Consumer").start();
}
public void run() {
int i=0;
while(i<15) {
int r=q.get();
System.out.println("consumed:"+r);
i++;
}
}
}
class PCFixed {
public static void main(String args[]) {
Q q = new Q();
new Producer(q);
new Consumer(q);
System.out.println("Press Control-C to stop.");
}}
```

a) Deadlock

Algorithm:



10) b

Dead lock

A thread enters the monitor on object X and another thread enters the monitor on object Y. If the thread in X tries to call any synchronized method on X, it will block as expected

```
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 Interupted");
}
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 Interupted");
}
System.out.println
(name+"trying to call
A.last()");
a.last();
}
void last() {
System.out.println
("Inside A.last");
}
}
class Deadlock implements
Runnable {
A a = new A();
B b = new B();
Deadlock() {
Thread.currentThread().
setName("MainThread");
Thread t = new Thread
(this, "RacingThread");
t.start();
a.foo(b);
System.println("Back
in main thread");
}
public void run() {
b.bar(a);
System.out.println
("Back in other thread");
}
public static void main
(String args[]) {
new Deadlock();
}
}
```

Output ->

```
Main Thread entered A.foo
Racing Thread entered A.foo
Racing Thread entered B.bar
Main Thread trying to call B.last()
Inside A.last
Racing Thread trying to call A.last()
Back in main thread
Inside A.last
Back in other thread
```

Code:

```
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 A.last");

}

}

class Deadlock implements Runnable
{

A a = new A();

B b = new B();

Deadlock() {

Thread.currentThread().setName("MainThread");

Thread t = new Thread(this,"RacingThread");

t.start();

a.foo(b);

System.out.println("Back in main thread");

}
public void run() {

b.bar(a);

System.out.println("Back in other thread");

}

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

```
new Deadlock();
```