VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

Skanda Mahesh (1BM23CS332)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019 Sep

2024-Jan 2025

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**OBJECT ORIENTED JAVA PROGRAMMING**" carried out by Skanda Mahesh**(1BM23CS332),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of **Object-Oriented Java Programming Lab - (23CS3PCOOJ)** work prescribed for the said degree.

Dr. Nandhini Vineeth

Associate Professor,
Department of CSE,
BMSCE, Bengaluru

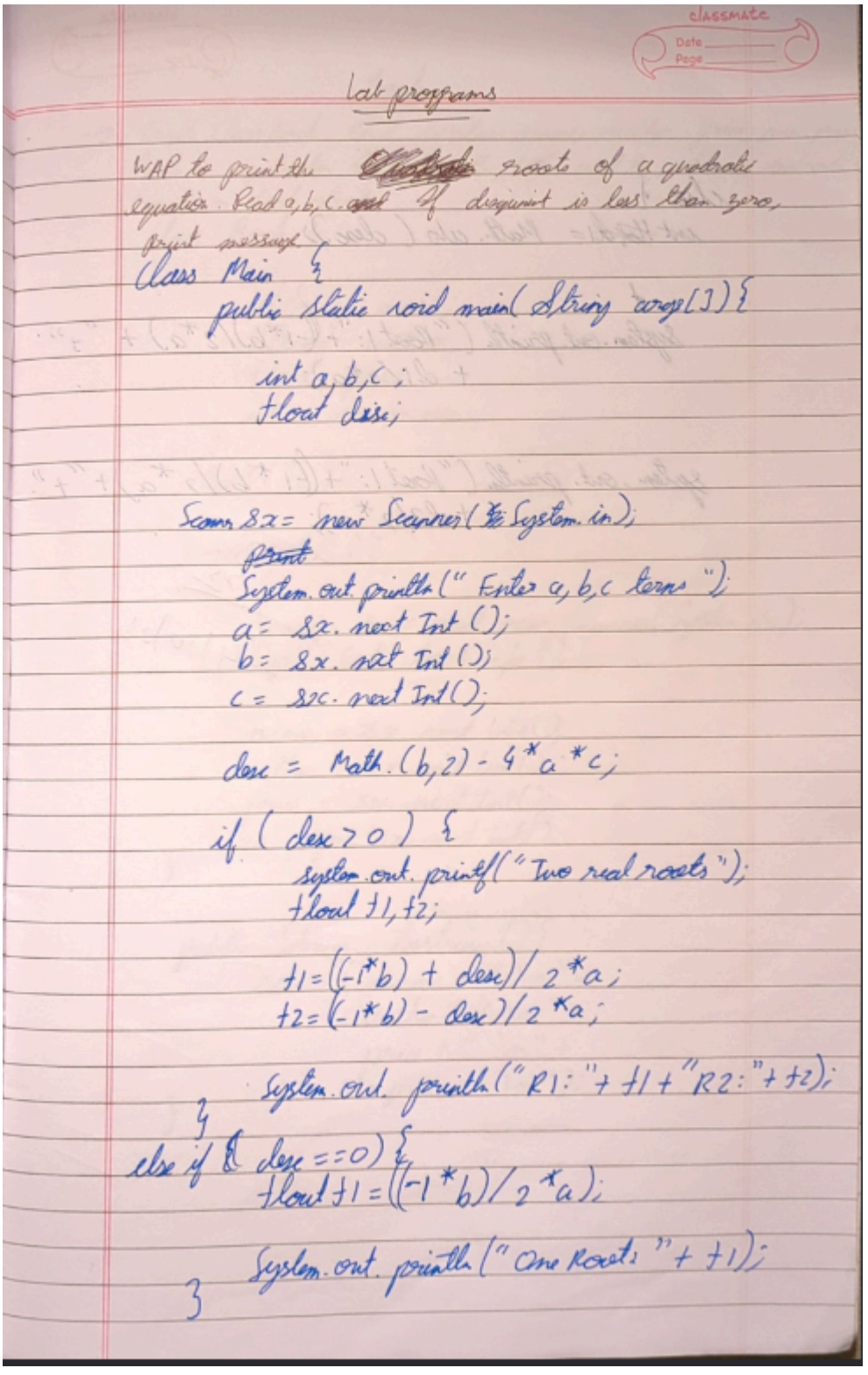
Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

INDEX

Program 1

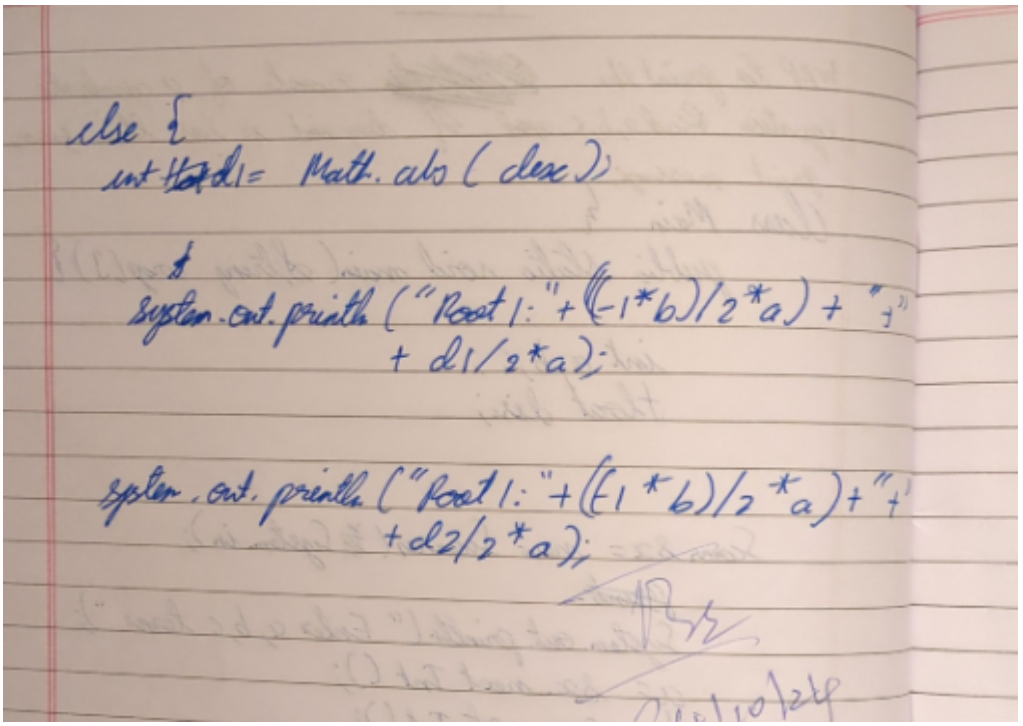
Develop a Java program that prints all real solutions to the quadratic equation ax2+bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminate b2-4ac is negative, display a message stating that there are no real solutions.



Lab programs

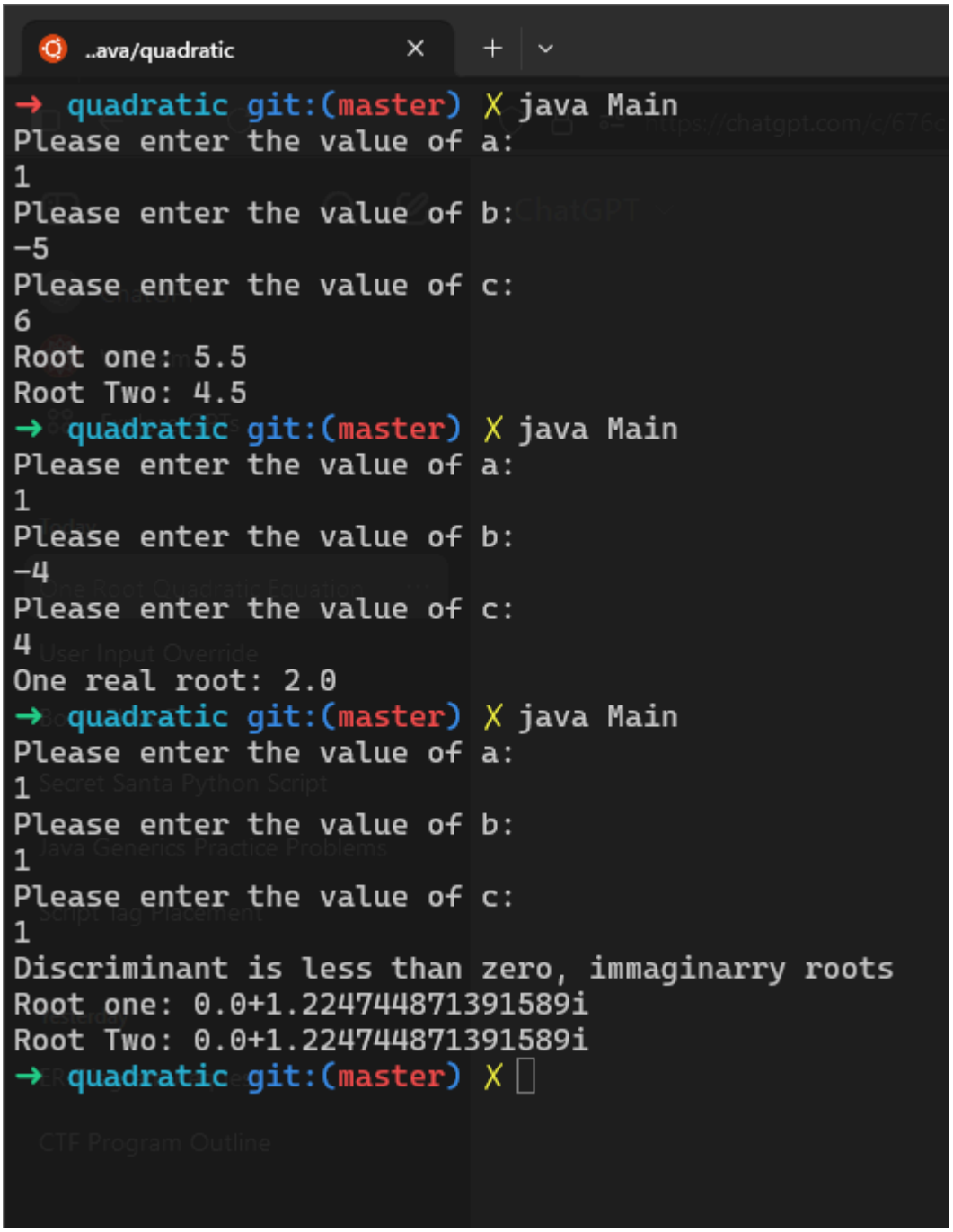
WAP to print the roots of a quadratic equation. Read a, b, c. If discriminant is less than zero, print message.

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

        int a, b, c;
        float desc;

Scanner sx = new Scanner(System.in);
        System.out.println("Enter a, b, c terms");
        a = sx.nextInt();
        b = sx.nextInt();
        c = sx.nextInt();

        desc = Math.(b,2) - 4*a*c;

        if (desc > 0) {
            system.out.printf("Two real roots");
            float f1, f2;

            f1 = ((-1*b) + desc)/2*a;
            f2 = ((-1*b) - desc)/2*a;

            System.out.println("R1: " + f1 + "R2: " + f2);
        }
else if (desc == 0) {
            float f1 = ((-1*b)/2*a);

            System.out.println("One Root: " + f1);
        }
```



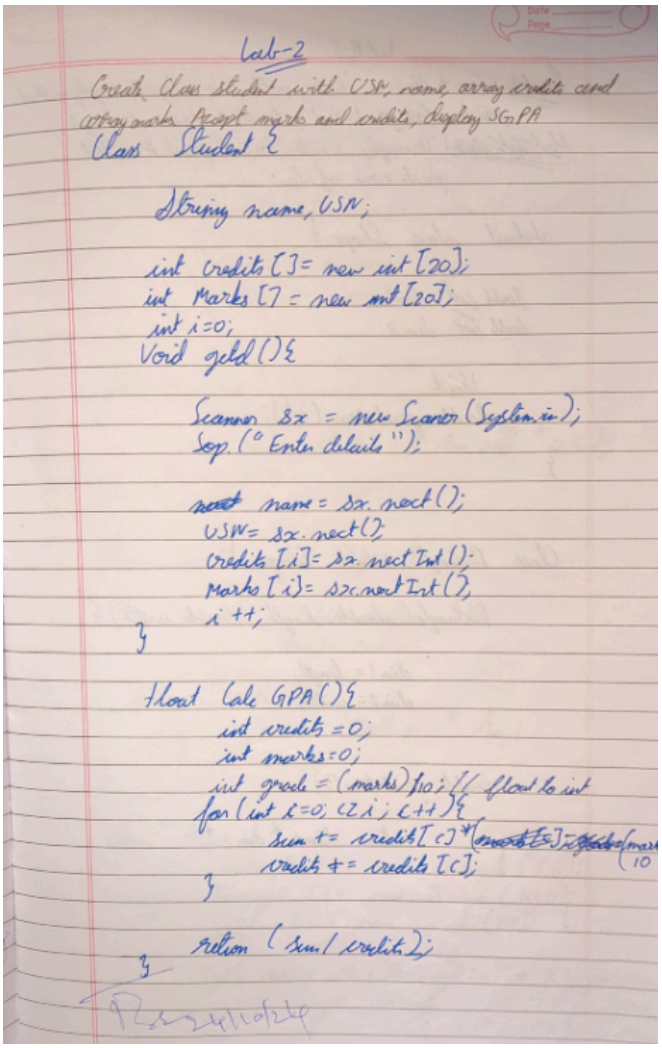

```
import java.util.Scanner;

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

      int a, b, c;


      Scanner sn = new Scanner(System.in);

      System.out.println("Please enter the value of a: ");
      a = sn.nextInt();
      System.out.println("Please enter the value of b: ");
      b = sn.nextInt();
      System.out.println("Please enter the value of c: ");
      c = sn.nextInt();
      sn.close();

      Quadratic quad = new Quadratic(a, b, c);
      quad.return_roots();

   }
}

class Quadratic {
   int a, b, c;
   Quadratic(int x, int y, int z) {
      a = x;
      b = y;
      c = z;
   }


   void return_roots() {

      double disc = Math.pow(b, 2) - (4 * a * c);
```

```
        if (disc < 0) {
            double r1, r2;
            r1 = ((-1 * b)  / (2 * a));
            r2 = ((-1 * b)  / (2 * a));
            System.out.println("Discriminant is less than zero, immaginarry roots");

            System.out.println("Root one: " + r1 + "+" + Math.sqrt(Math.abs(disc) / (2*a)) + "i" );
            System.out.println("Root Two: " + r2 + "+" + Math.sqrt(Math.abs(disc) / (2*a)) + "i" );
        }

        else if(disc == 0) {
            double r1;
            r1 = (-1 * b) / 2*a;
            System.out.println("One real root: " + r1);
        }

        else {
            double r1, r2;
            r1 = ((-1 * b) + Math.sqrt(disc) / (2 * a));
            r2 = ((-1 * b) - Math.sqrt(disc) / (2*a));
            System.out.println("Root one: " + r1);
            System.out.println("Root Two: " + r2);
        }
    }
}
```


..ava/quadratic
→ quadratic git:(master) X java Main
Please enter the value of a:
1
Please enter the value of b:
-5
Please enter the value of c:
6
Root one: 5.5
Root Two: 4.5
→ quadratic git:(master) X java Main
Please enter the value of a:
1
Please enter the value of b:
-4
Please enter the value of c:
4
One real root: 2.0
→ quadratic git:(master) X java Main
Please enter the value of a:
1
Please enter the value of b:
1
Please enter the value of c:
1
Discriminant is less than zero, immaginarry roots
Root one: 0.0+1.2247448713915891
Root Two: 0.0+1.2247448713915891
→ quadratic git:(master) X

Program 2

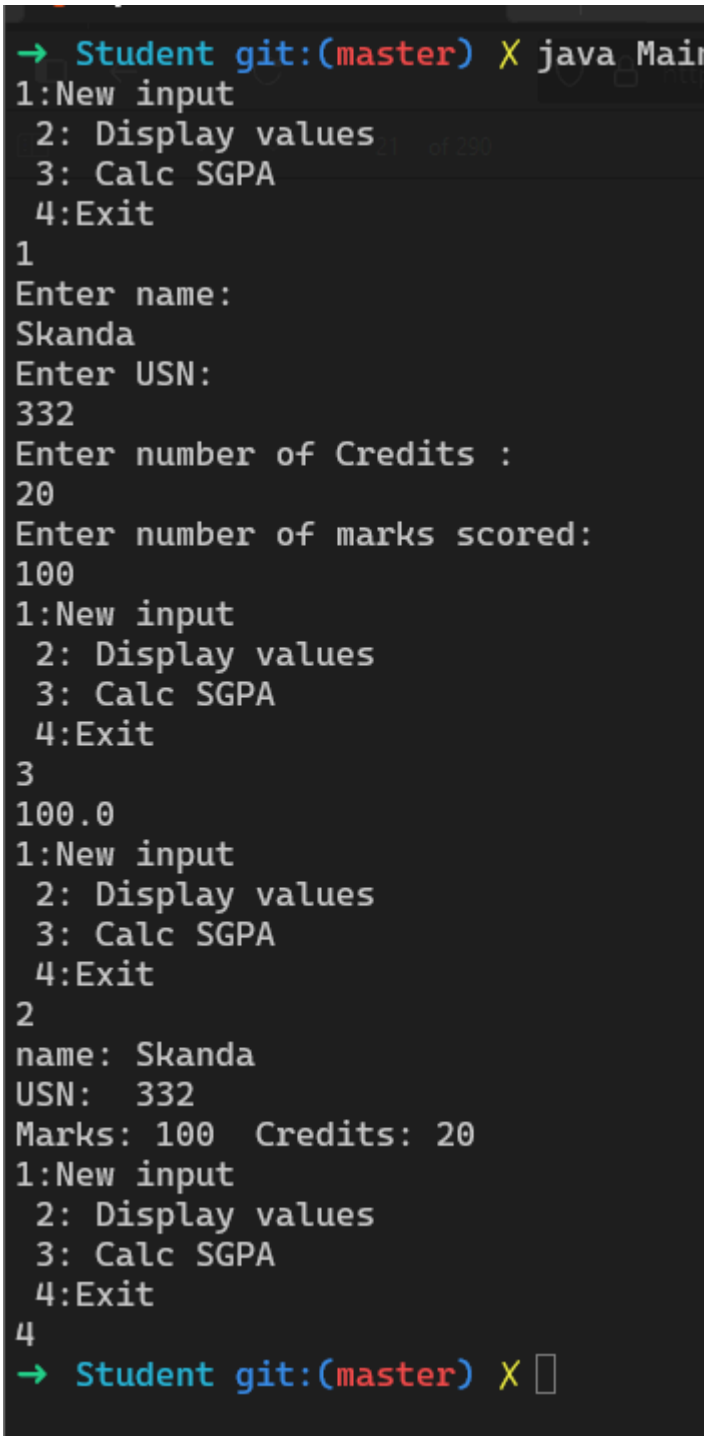
Develop a Java program to create a class Student with members usn, name, an array credits and an array marks. Include methods to accept and display details and a method to calculate SGPA of a student.




```
import java.util.Scanner;

class Student {
    String name;
    String USN;
    int credits[] = new int[20]; // Ask ma'am about dynamic mem allocation like C How to do in classes

    int marks[] = new int[20];  // If i did the mem alloc inside the func, it would be only in function scope
    int i = 0;

    void getd() {
        Scanner sx = new Scanner(System.in);
        System.out.println("Enter name: ");
        name = sx.next();
        System.out.println("Enter USN: ");
        USN = sx.next();
        System.out.println("Enter number of Credits : ");
```

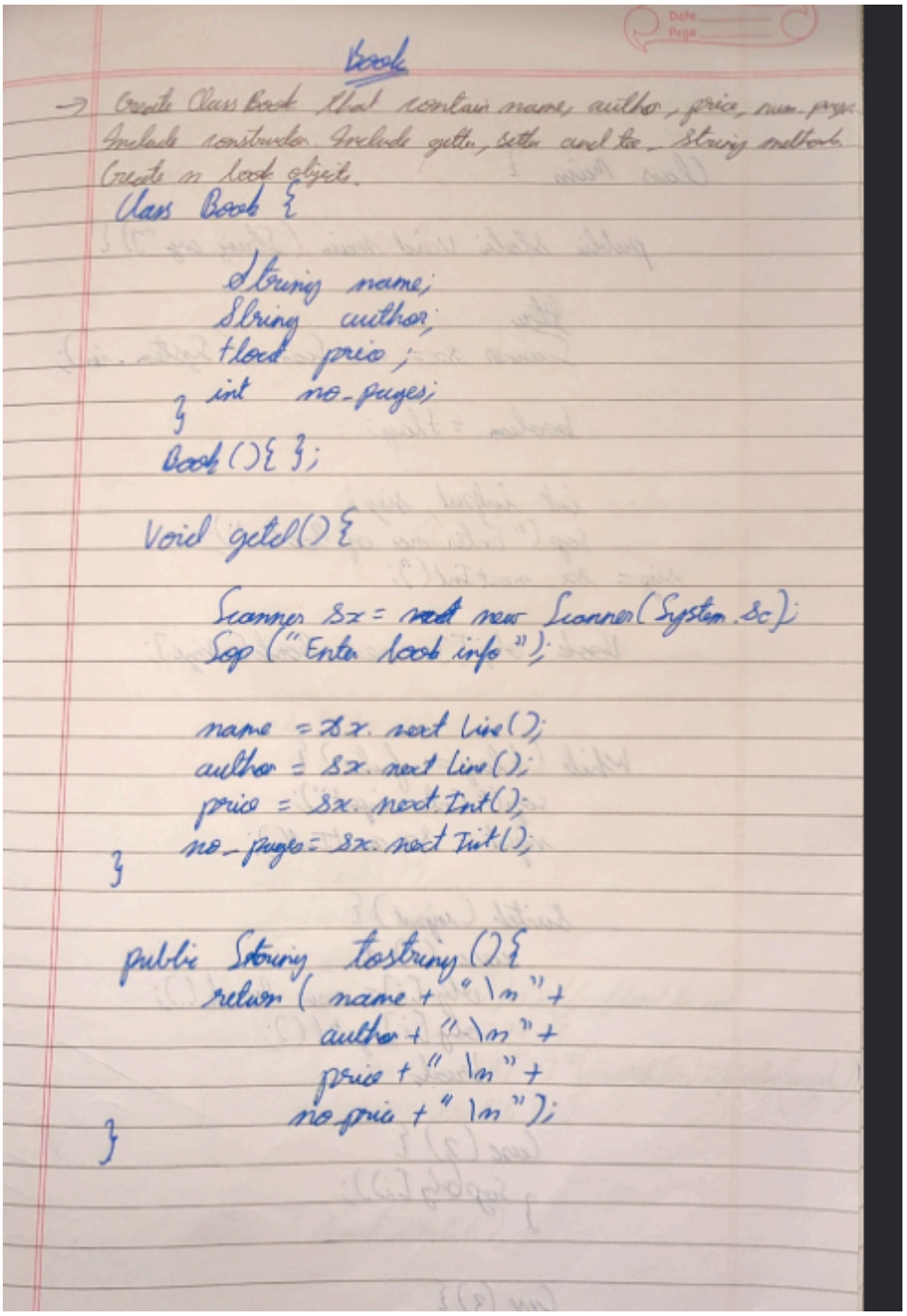
```
        credits[i] = sx.nextInt();
        System.out.println("Enter number of marks scored: ");
        marks[i] = sx.nextInt();
        i++;
    }

    float calc_SGPA() {
        int credits_count = 0;
        int sum = 0;
        for (int c = 0; c<i; c++) {
            sum += credits[c] * marks[c];
            credits_count += credits[c];
        }

        return sum/credits_count;
    }

    void putd() {
        System.out.println("name: " + this.name);
        System.out.println("USN:  " + this.USN);

        for (int c = 0; c<i; c++) {
            System.out.println("Marks: " + marks[c] + "  Credits: " + credits[c]);
        }

    }



}

class Main {
    public static void main(String[] args) {
        boolean flag = false;
        int input;

        Student s1 = new Student();

        while (flag == false) {
            Scanner sx = new Scanner(System.in);
            System.out.println("1:New input \n 2: Display values \n 3: Calc SGPA \n 4:Exit ");
            input = sx.nextInt();

            switch (input) {
                case 1:
                    s1.getd();
                    break;
                case 3:
                    System.out.println(s1.calc_SGPA());
                    break;
                case 2:
                    s1.putd();
                    break;
                case 4:
                    flag = true;

            }
        }
```

}

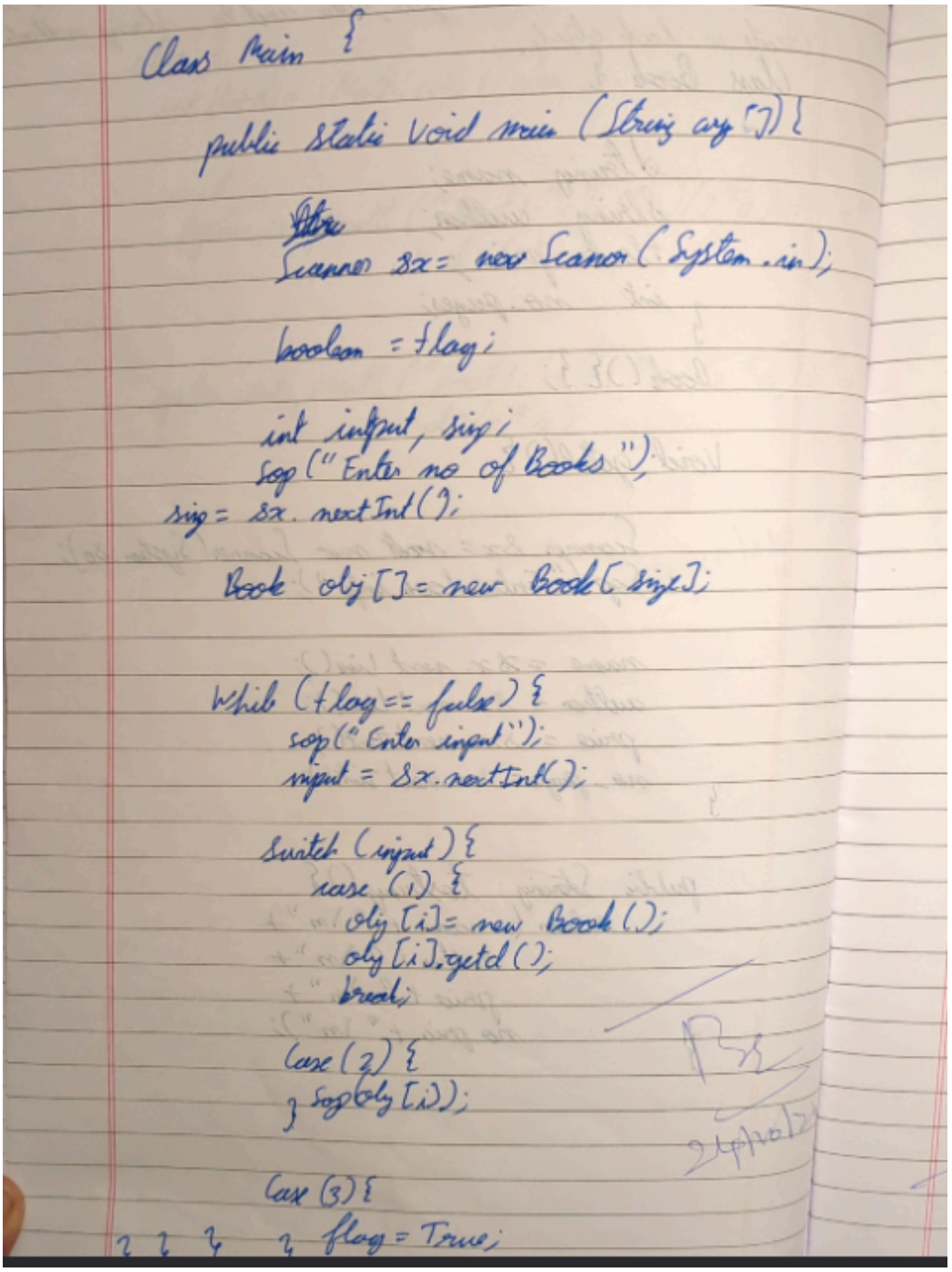

→ Student git:(master) X java Mai
1:New input
2: Display values
3: Calc SGPA
4:Exit
1
Enter name:
Skanda
Enter USN:
332
Enter number of Credits :
20
Enter number of marks scored:
100
1:New input
2: Display values
3: Calc SGPA
4:Exit
3
100.0
1:New input
2: Display values
3: Calc SGPA
4:Exit
2
name: Skanda
USN: 332
Marks: 100 Credits: 20
1:New input
2: Display values
3: Calc SGPA
4:Exit
4
→ Student git:(master) X

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.



```
Book

→ Create Class Book that contain name, author, price, num_pages
Include constructor. Include getter, setter and to_String method.
Create n book objects.
Class Book {

    String name;
    String author;
    float price;
    int no_pages;
}
Book(){};

Void getInfo(){

    Scanner sx = new Scanner(System.in);
    Sop("Enter book info");

    name = sx.next Line();
    author = sx.next Line();
    price = sx.next Int();
    no_pages = sx.next Int();
}

public String toString(){
    return( name + "\n" +
            author + "\n" +
            price + "\n" +
            no price + "\n");
}
```




```
import java.util.Scanner;

class Book {
    String name;
    String author;
    float price;
    int no_pages;

    Book() {
        name = "";
        author = "";
        price = 0;
        no_pages = 0;
    }

    Book(String name, String author, float price, int no_pages) {
```

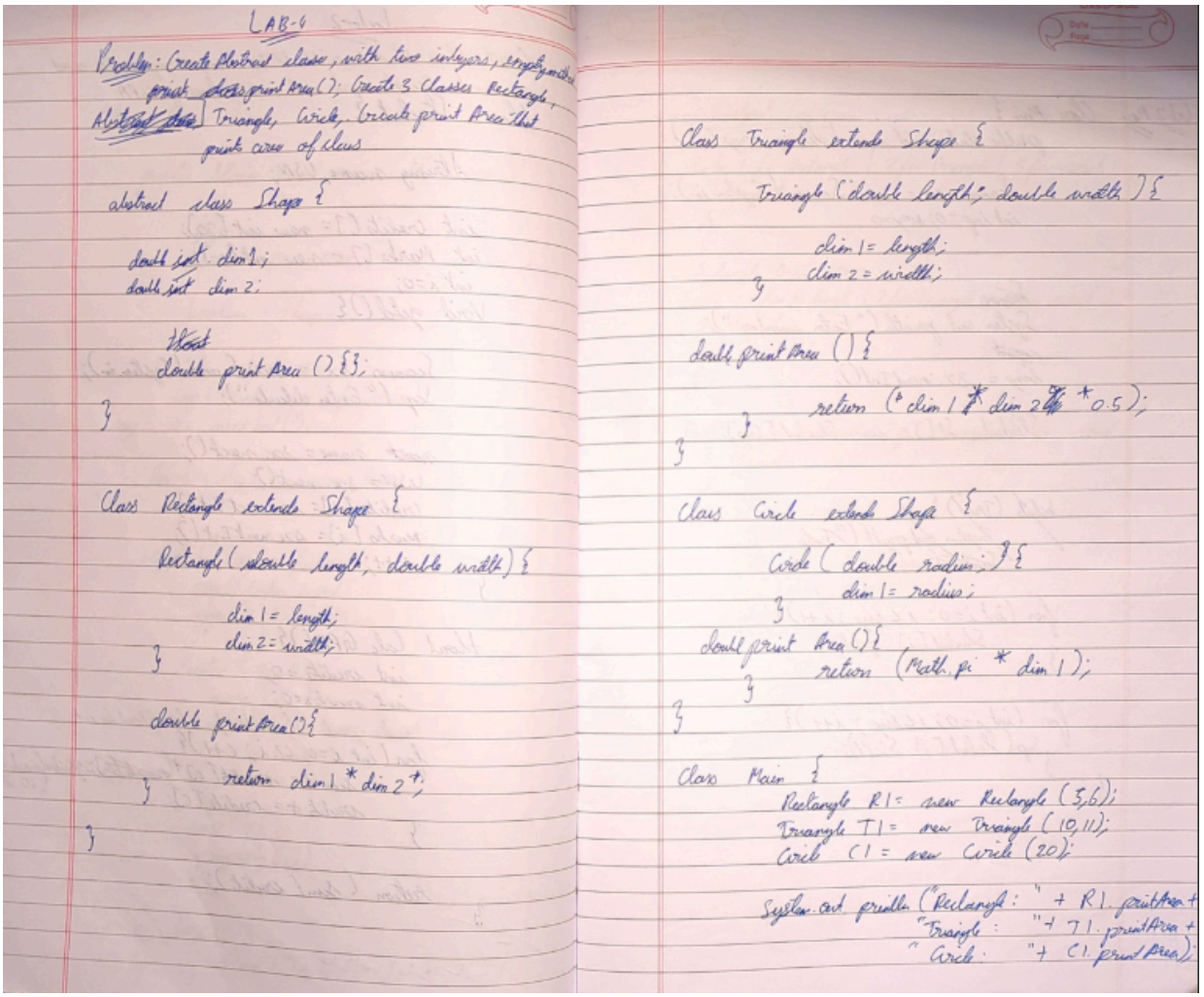
```
        this.name = name;
        this.author = author;
        this.price = price;
        this.no_pages = no_pages;
    }

    void getd() {
        Scanner sx = new Scanner(System.in);
        System.out.println("Please enter the book information in the format\n 1:Book name \n 2: Author
\n 3: Price \n 4: no_pages:  ");
        name = sx.nextLine();
        author = sx.nextLine();
        price = sx.nextFloat();
        no_pages = sx.nextInt();

    }

    void putd() {
        System.out.println("Name: " + name);
        System.out.println("Author: " + author);
        System.out.println("Price:" + price);
        System.out.println("Number of pages: " + no_pages);

    }

    public String tostring() {
        return (name + " \n " + author + " \n " + price + "\n" + no_pages + "\n");
    }
}

class Main {

    public static void main(String[] args) {
        Scanner sx = new Scanner(System.in);
        boolean flag = false;
        int input = 0;
        int size = 0;
        System.out.println("Enter number of books: ");
        size = sx.nextInt();
        int i = 0;

        Book obj[] = new Book[size];


        while (flag == false) {

            System.out.println("1:Create new object \n 2: Display values \n 3:Exit ");
            input = sx.nextInt();

            switch (input) {
                case 1:
                    obj[i] = new Book();
                    obj[i].getd();
                    i++;
                    break;
                case 2:
                    for (int c = 0; c<i; c++) {
                        obj[c].putd();
```

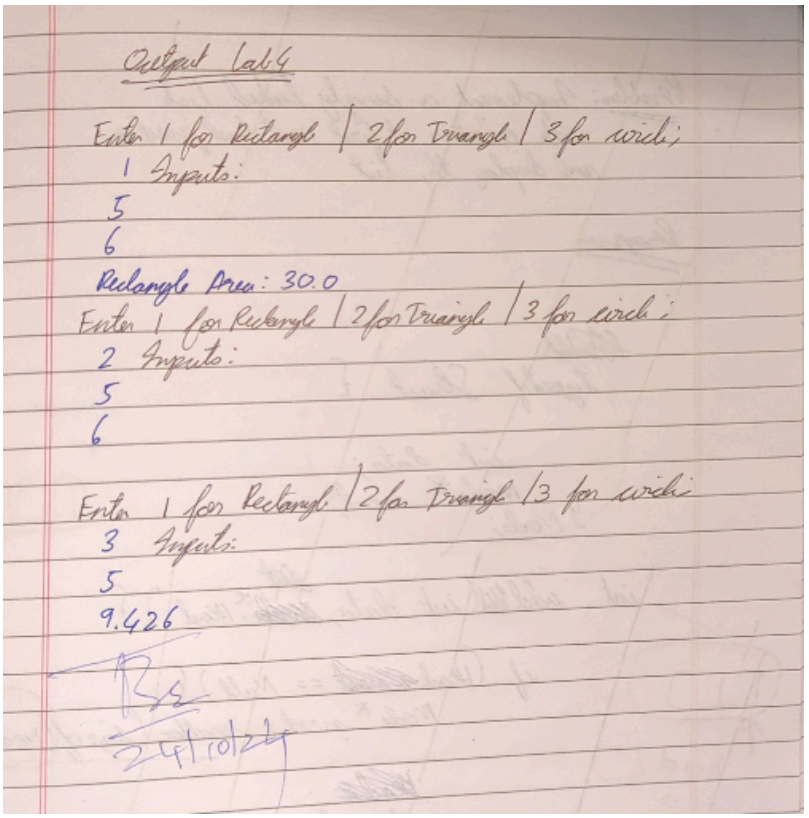
```
            }

            break;
        case 3:
            flag = true;

    }
}
  }

}
```

```
→ Book git:(master) X java Main
Enter number of books:
5
1:Create new object
 2: Display values
 3:Exit
1
Please enter the book information in the format
 1:Book name
 2: Author
 3: Price
 4: no_pages:
Book theif
Max
500
800
1:Create new object
 2: Display values
 3:Exit
2
Name: Book theif
Author: Max
Price:500.0
Number of pages: 800
1:Create new object
 2: Display values
 3:Exit
3
→ Book git:(master) X
```

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.





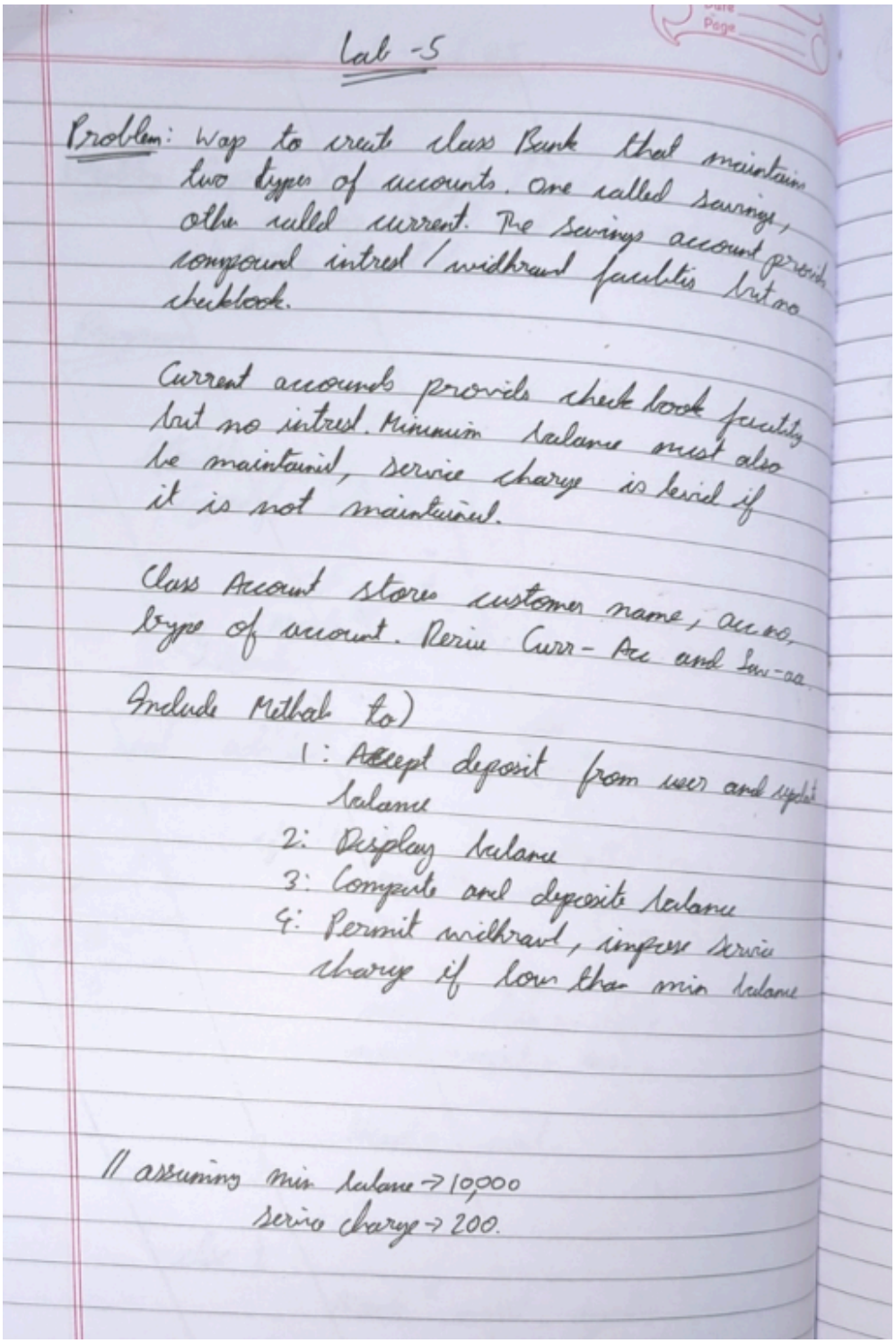

```
import java.util.Scanner;

abstract class Shape {
	double dim1;
	double dim2;

	abstract double printArea ();
}


class Rectangle extends Shape {
	Rectangle(double length, double width) {
		dim1 = length;
		dim2 = width;
	}

	double printArea () {
		return (dim1 * dim2);
	}

}

class Triangle extends Shape {
	Triangle(double length, double width) {
		dim1 = length;
		dim2 = width;
	}

	double printArea () {
		return ( 0.5 * dim1 * dim2);
	}
}

class Circle extends Shape {
```

```
	Circle(double radius) {
		dim1 = radius;
		dim2 = 0;
	}

	double printArea() {
		return (Math.PI * dim1);
	}

}


class Main {
	public static void main(String args[]) {
		Scanner sx = new Scanner(System.in);
		int input = 0;

		while (true) {
			System.out.println("Enter 1 for Rectangle | 2 for Triangle | 3 for Circle |
CTRL+C for exit ");
			input = sx.nextInt();
			int inp1, inp2;
			System.out.println("Enter inputs: ");
			switch (input) {

				case (1):
					inp1 = sx.nextInt();
					inp2 = sx.nextInt();
					Rectangle R1 = new Rectangle(inp1, inp2);
					System.out.println("Rectangle: " + R1.printArea());
					break;


				case (2):
					inp1 = sx.nextInt();
					inp2 = sx.nextInt();
					Triangle T1 = new Triangle(inp1, inp2);

					System.out.println("Triangle: " + T1.printArea());
				case (3):
					inp1 = sx.nextInt();
					inp2 = sx.nextInt();
					Circle C1 = new Circle(inp1);

					System.out.println("Circle: " + C1.printArea());









	}

	}}}
```

```
→ Lab 4 git:(master) X java Main
Enter 1 for Rectangle | 2 for Triangle | 3 for Circle | CTRL+C for exit
1
Enter inputs:
50
500
Rectangle: 25000.0
Enter 1 for Rectangle | 2 for Triangle | 3 for Circle | CTRL+C for exit
2
Enter inputs:
50
500
Triangle: 12500.0
3
500
Circle: 9.42477796076938
Enter 1 for Rectangle | 2 for Triangle | 3 for Circle | CTRL+C for exit
^C%
→ Lab 4 git:(master) X
```

Program 5

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



Lab -5

Problem: Wap to creat class Bank that maintains two types of accounts. One called savings, other called current. The savings account provide compound intrest / withdrawl facilities but no chekbook.

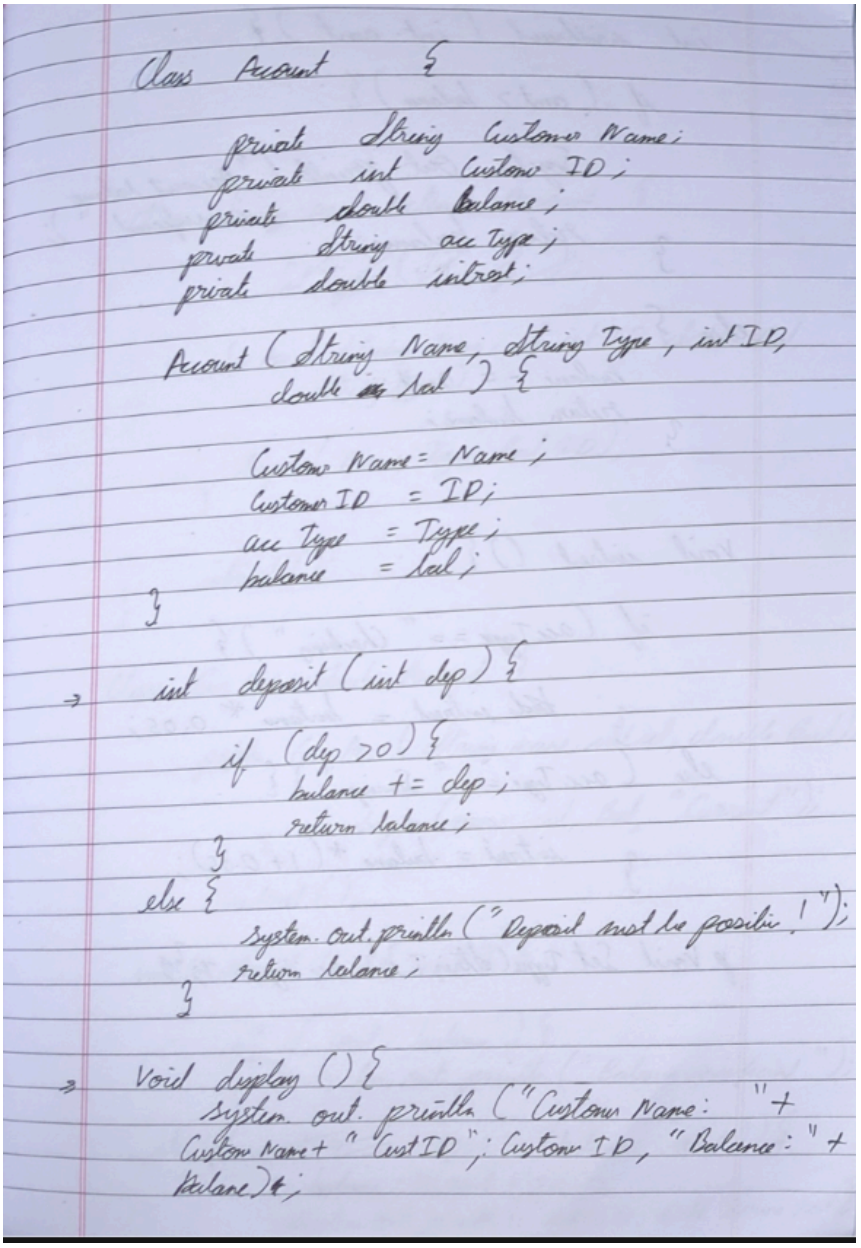
Current accounds provide chek book facility but no intrest. Minimum balance must also be maintaind, service charge is levied if it is not maintained.

Class Account store customer name, acc no, type of account. Derive Curr-Acc and Sav-acc.

Include Methods to)

1: Accept deposit from user and update balance
2: Display balance
3: Compute and deposite balance
4: Permit withdrawl, impose service charge if bow the min balance

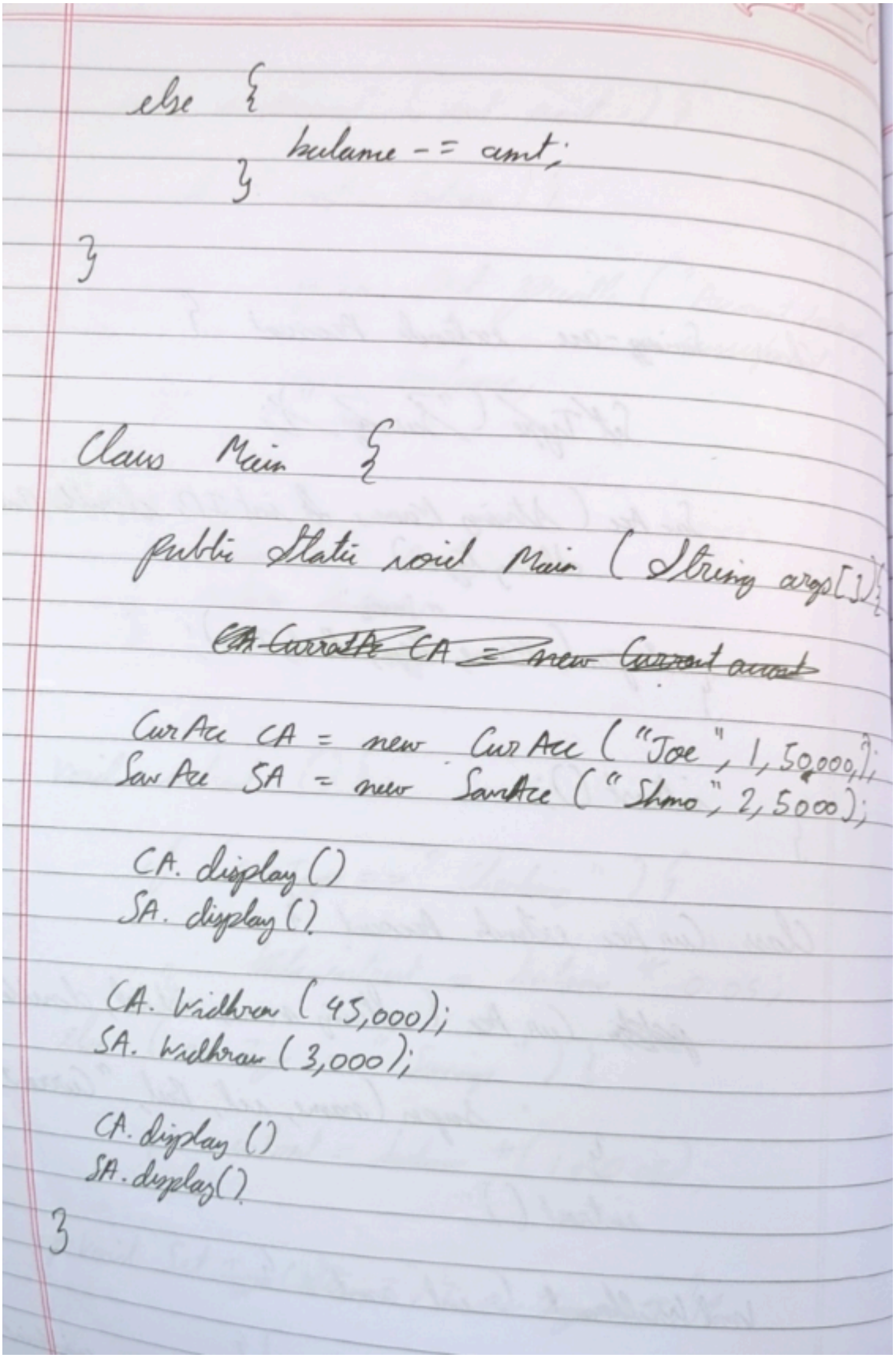
// assuming min balance → 10,000
service charge → 200.



```
class Account {

    private String CustomerName;
    private int CustomerID;
    private double balance;
    private String accType;
    private double intrest;

Account (String Name, String Type, int ID,
        double bal ) {

    CustomerName = Name;
    CustomerID   = ID;
    accType      = Type;
    balance      = bal;
}

→ int deposit (int dep) {

    if (dep >0) {
        balance += dep;
        return balance;
    }
else {
    system.out.println("Deposit must be positive!");
    return balance;
    }

→ Void display () {
    system.out.println("Customer Name: " +
CustomerName + "CustID"; CustomerID, "Balance: " +
balance);
```

```
int withdrawal (int amt ){
    if (amt > balance ){
        System.out.println("Account bal
                                insufficient
    }   return balance;

else {
        balance -= amt;
        return balance;
    }

void intrest (){
    if (accType == "Checking" ){
    }   double intrest = balance * 0.05;
else (accType == "Savings" ){
    }   intrest = balance *(1+0.05);

public void SetType(String T){ accType = T;}}
```

```
class Saving-acc extends Account {
        SetType("Savings");
    Sav Acc (String Name, int ID, double Bal,
            String type)
                    "Saving"
    { Super (Name, Type, Bal, ID);
    }
    intrest();
}
class Cur Acc extends Account {
    public Cur Acc (String name, int id, double Bal){
            super (name, id, Bal, "Current");
        }
    intrest();
void Withdrawl (int amt) {
    if (amt > balance) {
        System.out.println("Balance insufficient");
    }
else if(amt <= 10000 > (balance - amt)) {
        balance -=(amt + 200);
        System.out.println("Deducted, with service tax");
```



```
else {
        balance -= amt;
    }
}

class Main {

    public static void Main (String args[]) {

        CurrAcc CA = new CurrAcc ("Joe", 1, 50,000);
        SavAcc SA = new SavAcc ("Shmo", 2, 5000);

        CA.display()
        SA.display()

        CA.Withdraw (45,000);
        SA.Withdraw (3,000);

        CA.display()
        SA.display()
    }
```

```
class Account {
    protected String customerName;
    protected int customerID;
    protected double balance;
    protected String accType;
    protected double intrest;
```

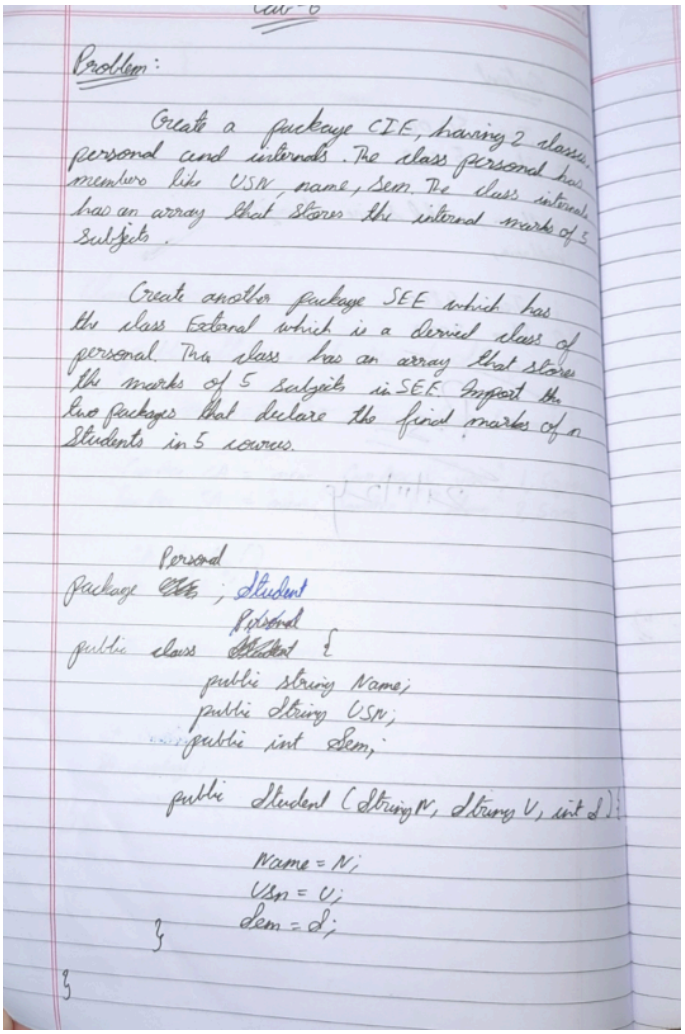
```
   Account(String Name, String type, int ID, double bal) {
      customerName = Name;
      customerID = ID;
      accType = type;
      balance = bal;
   }

   double deposit(int dep) {
      if (dep > 0) {
         balance += dep;
         return balance;
      }
      else {
         System.out.println("Deposit must be possitive!");
         return balance;
      }
   }

   void display() {
      System.out.println("Customer Name: " + customerName + " Customer ID: " + customerID + "
Balance: " + balance);;
   }

   double widhrawl (int ammount) {
      if (ammount > balance) {
         System.out.println("Account balance is insufficient");
         return balance;
      }
      else {
         balance -= ammount;
         return balance;
      }
   }

   void intrest () {
      if (accType == "Checking") {
         intrest = balance * 0.05;
      }

      else if (accType == "Savings") {
         intrest = balance * (1 + 0.05);
      }
   }

   void SetType (String T) {
      accType = T;
   }

}

class Savings extends Account {
   Savings(String Name, int ID, double bal) {
      super(Name, "Savings", ID, bal);
      intrest();
   }

}
```

```
class Current extends Account {
    Current(String Name, int ID, double bal) {
        super(Name, "Current", ID, bal);
        intrest();
    }

    @Override
    double widhrawl(int ammount) {
        if (ammount > balance) {
            System.out.println("ballance is not sufficient!");
            return balance;
        }
        else if (10000 > (balance - ammount)) {
            balance -= (balance - ammount);
            return balance;
        }
        else {
            balance -= ammount;
            return balance;
        }
    }

}


class Main {
    public static void main(String[] args) {
        Current c1 = new Current("Skanda", 1, 5000);
        Savings s1 = new Savings("Skanda", 2, 20000);

        c1.display();
        s1.display();
        c1.widhrawl(40000);
        s1.widhrawl(205);

        c1.display();
        s1.display();

    }

}
```

```
➜ Lab 5 git:(master) X ls
Account.class  Current.class  Main.class  Savings.class  lab5.java
➜ Lab 5 git:(master) X java Main
Customer Name: Skanda Customer ID: 1 Balance: 5000.0
Customer Name: Skanda Customer ID: 2 Balance: 20000.0
ballance is not sufficient!
Customer Name: Skanda Customer ID: 1 Balance: 5000.0
Customer Name: Skanda Customer ID: 2 Balance: 19795.0
➜ Lab 5 git:(master) X
```

Program 6

Create a package CIE which has two classes - Personal and Internals. The class Personal has members like usn, name, sem. The class Internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Personal. 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.



Problem:

Create a package CIE, having 2 classes personal and internals. The class personal has members like USN, name, Sem. The class internals has an array that stores the internal marks of 5 subjects.

Create another package SEE which has the class External which is a derived class of personal. This class has an array that stores the marks of 5 subjects in SEE. Import the two packages that declare the final marks of n Students in 5 courses.

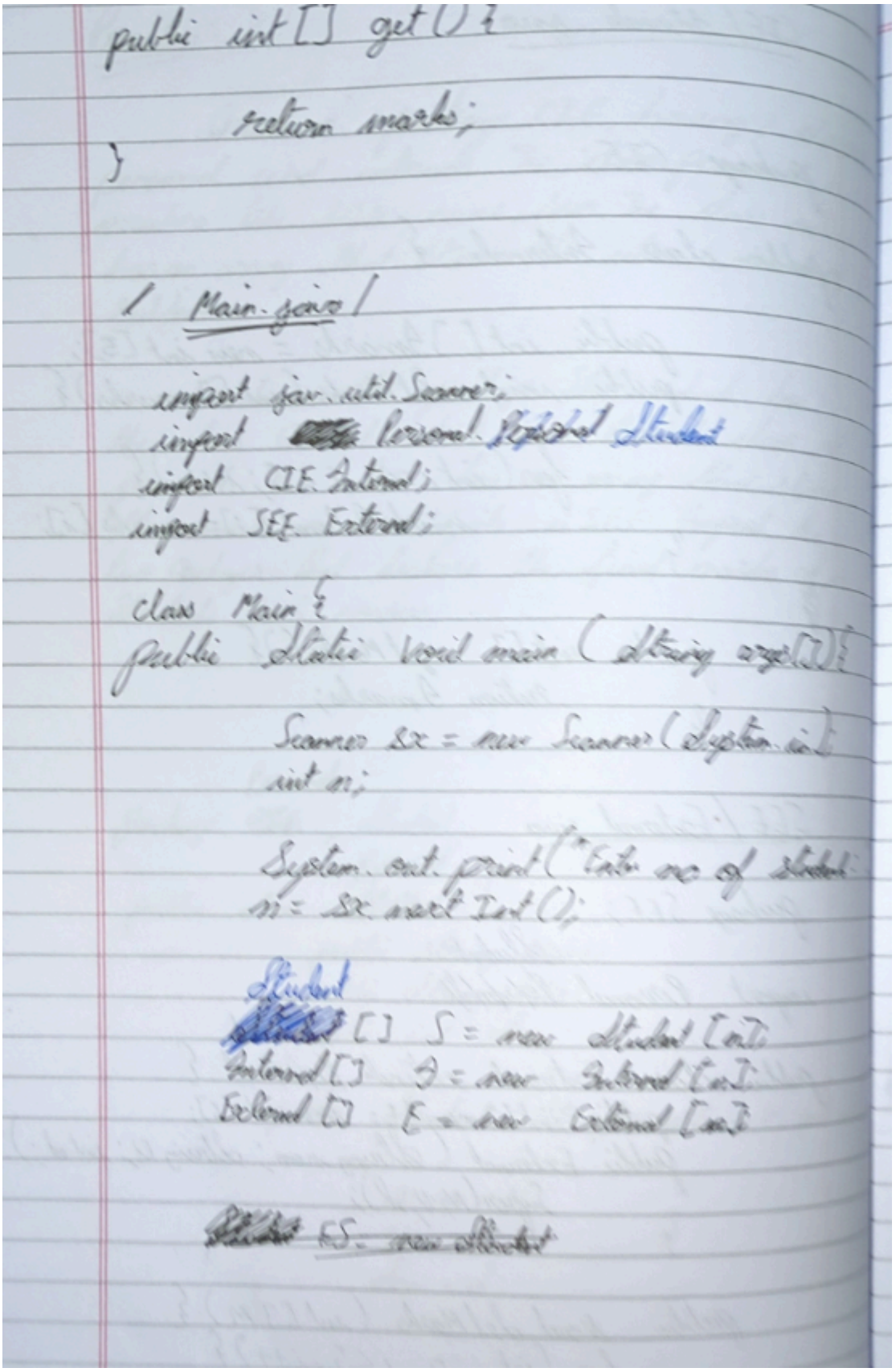
```
Personal
package CIE ; Student
              Personal
public class Student {
        public String Name;
        public String USN;
        public int Sem;

        public Student (String N, String U, int S);

                Name = N;
                Usn = U;
                Sem = S;
        }
}
```

CIE / Internals.java

```
package CIE;

public class Internals {

    public int[] Imarks = new int[5];
    public void SetMarks(int[] marks) {

        for(int i=0; i<5; i++) {
            marks Internal[i] = marks[i];
        }
    }

    public int[] getMarks() {
        return Imarks;
    }
```

SEE / External.java

```
package SEE;

import Personal.Student;

public class External extends Student {
    public int[] marksE = new int[5];
    public External(String name, String U, int d) {
        super(N, U, d);
    }

    public void SetMarks(int[] M) {
        for(int i=0; i<5; i++) {
            marksE[i] = M[i];
```



```
import Personal.Personal;


import java.util.Scanner;

class External extends Personal {
        int Marks[] = new int[5];
```

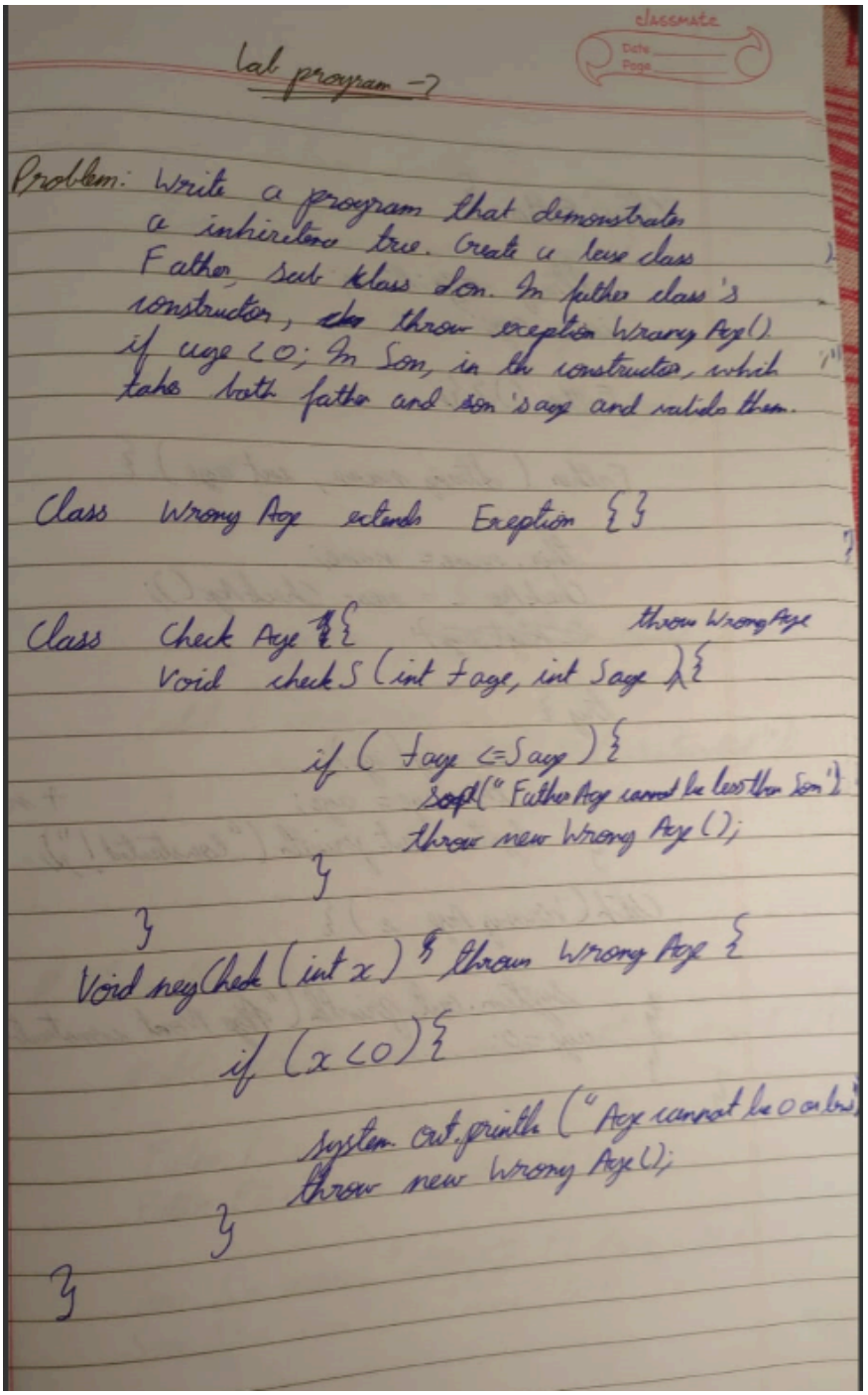
```
	public void getd() {
		Scanner sx = new Scanner(System.in);
		for (int i = 0; i<5; i++) {
			Marks[i] = sx.nextInt();
	}

	}
	public void setd() {

		for (int i = 0; i<5; i++) {
			System.out.println(Marks[i]);
			}
	}

	}package Internal;

import java.util.Scanner;
public class Internal {

	int[] Cources = new int[5];

	public void setd() {
		Scanner sx = new Scanner(System.in);
		System.out.println("Enter details of the 5 marks");

		for (int i = 0; i< 5; i++) {
			Cources[i] = sx.nextInt();
			}
	}
	public void getd() {

		for (int i = 0; i< 5; i++) {
			System.out.println(Cources[i]);
			}
}
}package Personal;

public class Personal {
	String usn;
	String name;
	int sem;
}
import Personal.Personal;
import Internal.*;


public class main {

  public static void main(String[] args) {

   Internal i1 = new Internal();
   External e1 = new External();

   i1.setd();
   i1.getd();
```

```
    e1.setd();
    e1.getd();
   }
}
```

```
→ CIE git:(master) X java main
Enter details of the 5 marks
5
6
7
8
9
5
6
7
8
9
```

Program 7

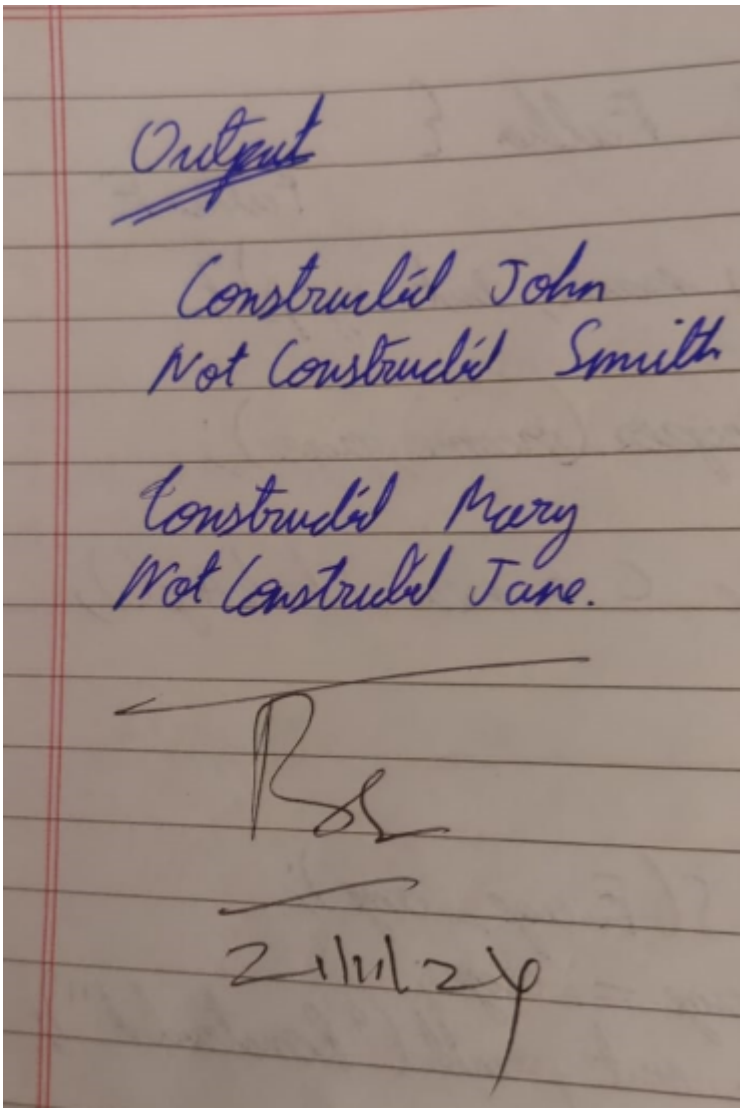
Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son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



lab program -7

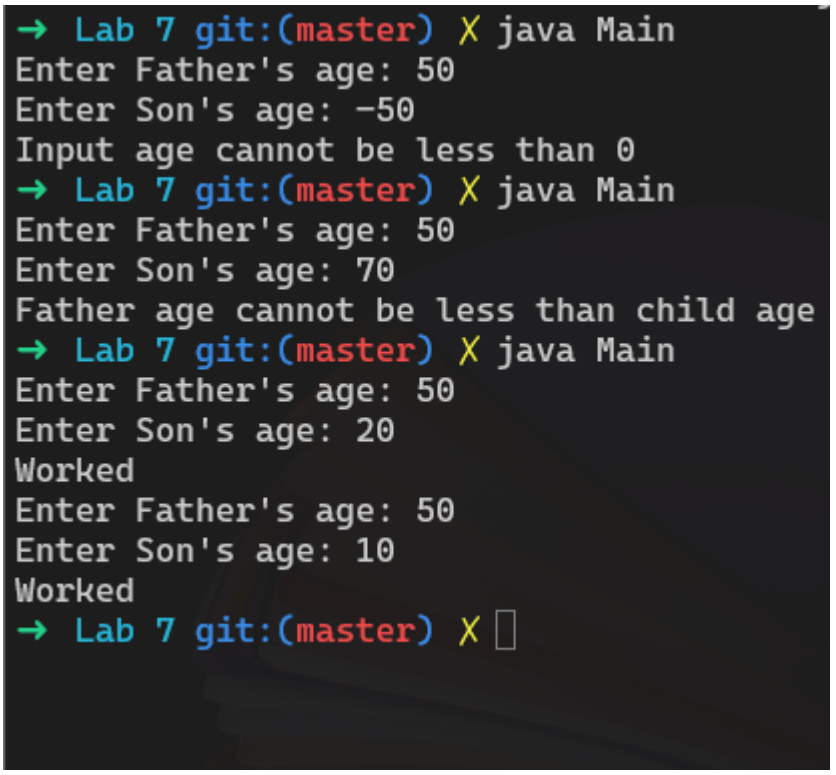
Problem: Write a program that demonstrates a inheritance tree. Create a base class Father, sub class Son. In father class's constructor, throw exception WrongAge() if age < 0; In Son, in the constructor, which takes both father and son's age and validate them.

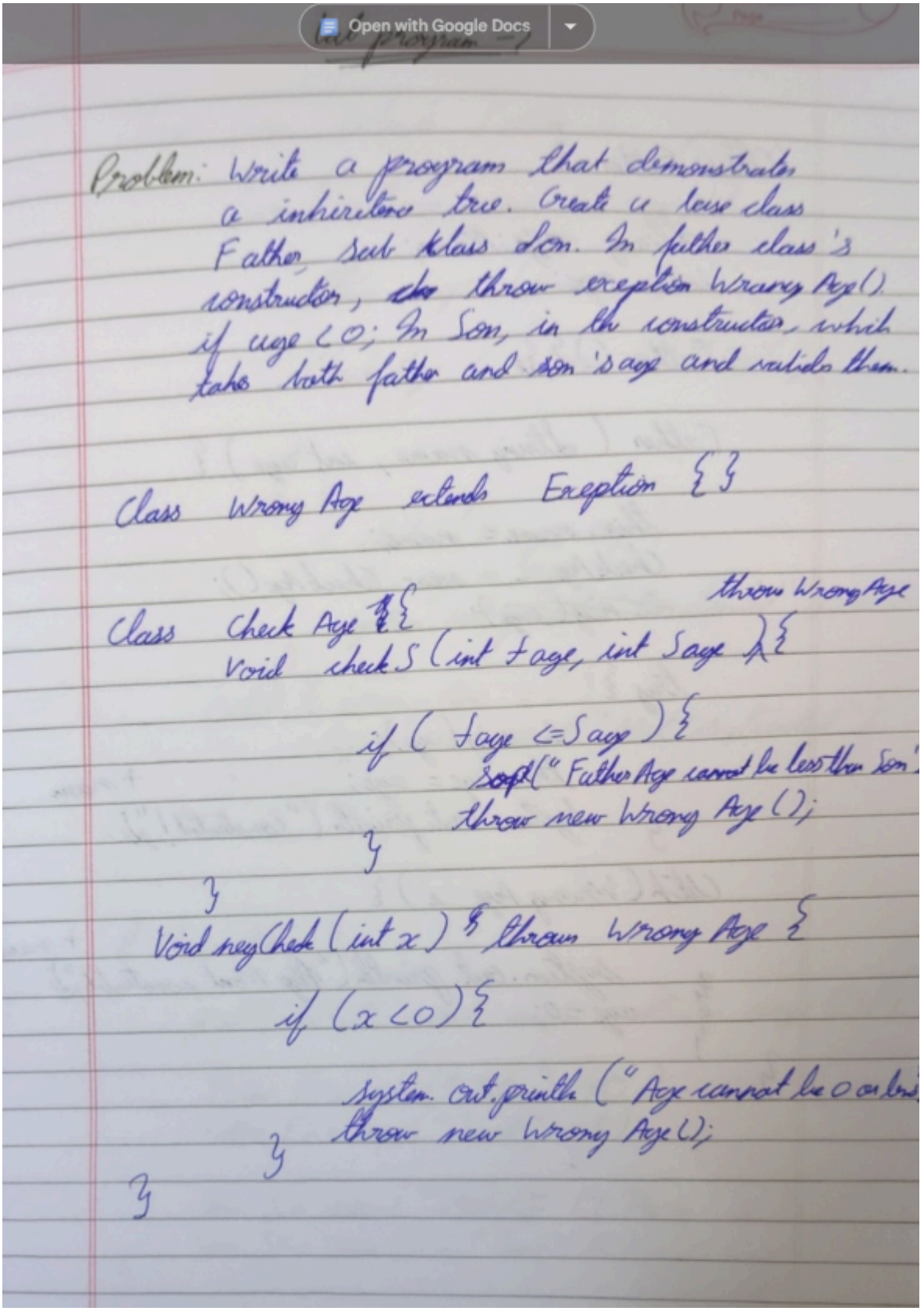
```
class WrongAge extends Exception {}

class CheckAge {
    void checkS(int fage, int Sage) throws WrongAge {
        if (fage <= Sage) {
            System.out.println("Father Age cannot be less than Son");
            throw new WrongAge();
        }
    }
    void negCheck(int x) throws WrongAge {
        if (x < 0) {
            System.out.println("Age cannot be 0 or less");
            throw new WrongAge();
        }
    }
}
```

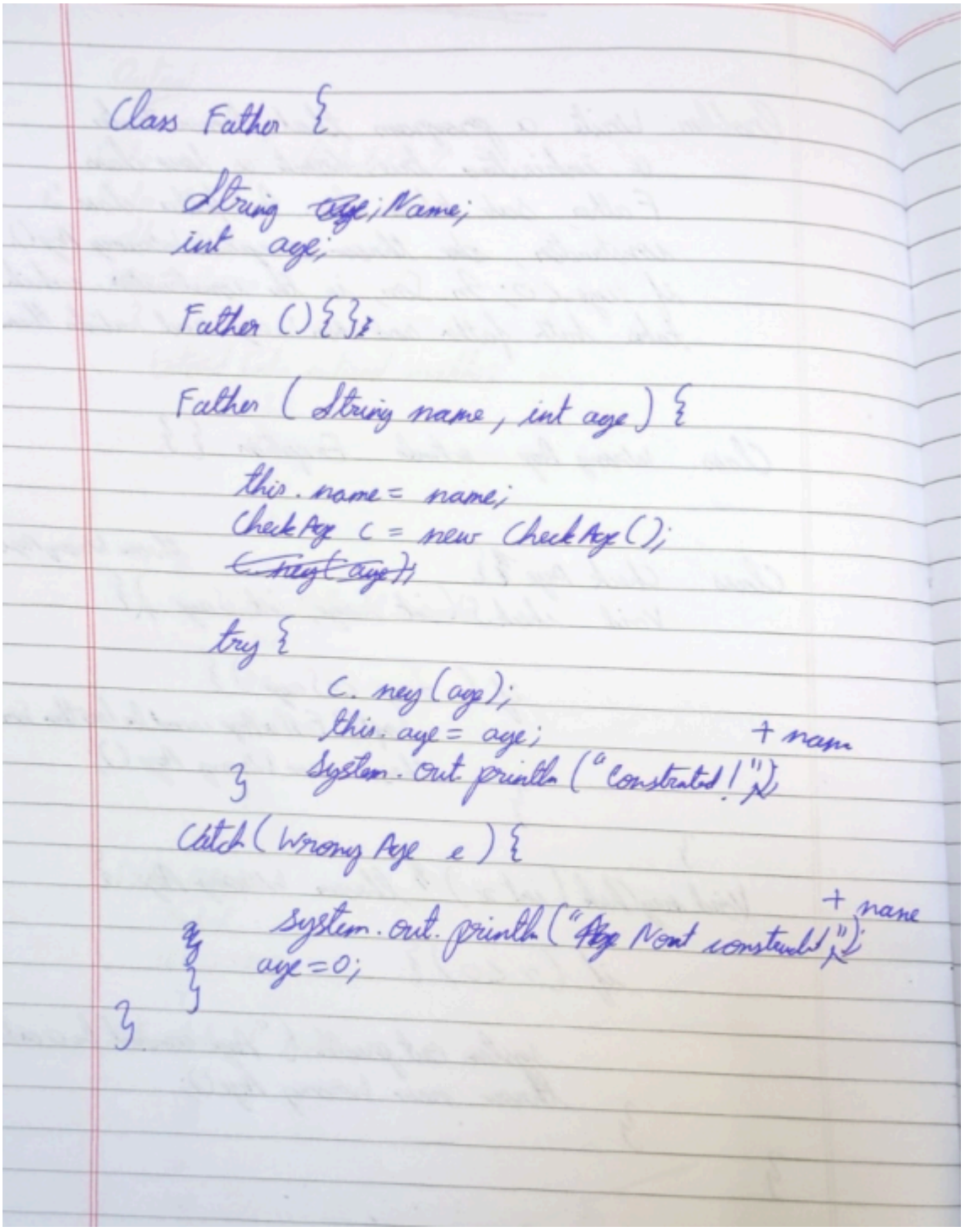



```
import java.util.Scanner;

class WrongAge extends Exception {
    public String toString() {
        return "Input age cannot be less than 0";
    }
}

class MismatchedAge extends Exception {
    public String toString() {
        return "Father age cannot be less than child age";
    }
}


class Father {
    int age;
    Father() {};
    Father(int i) throws WrongAge {
        if (i < 0) {
            throw new WrongAge();
        }
        else {
            this.age = i;
        }
```

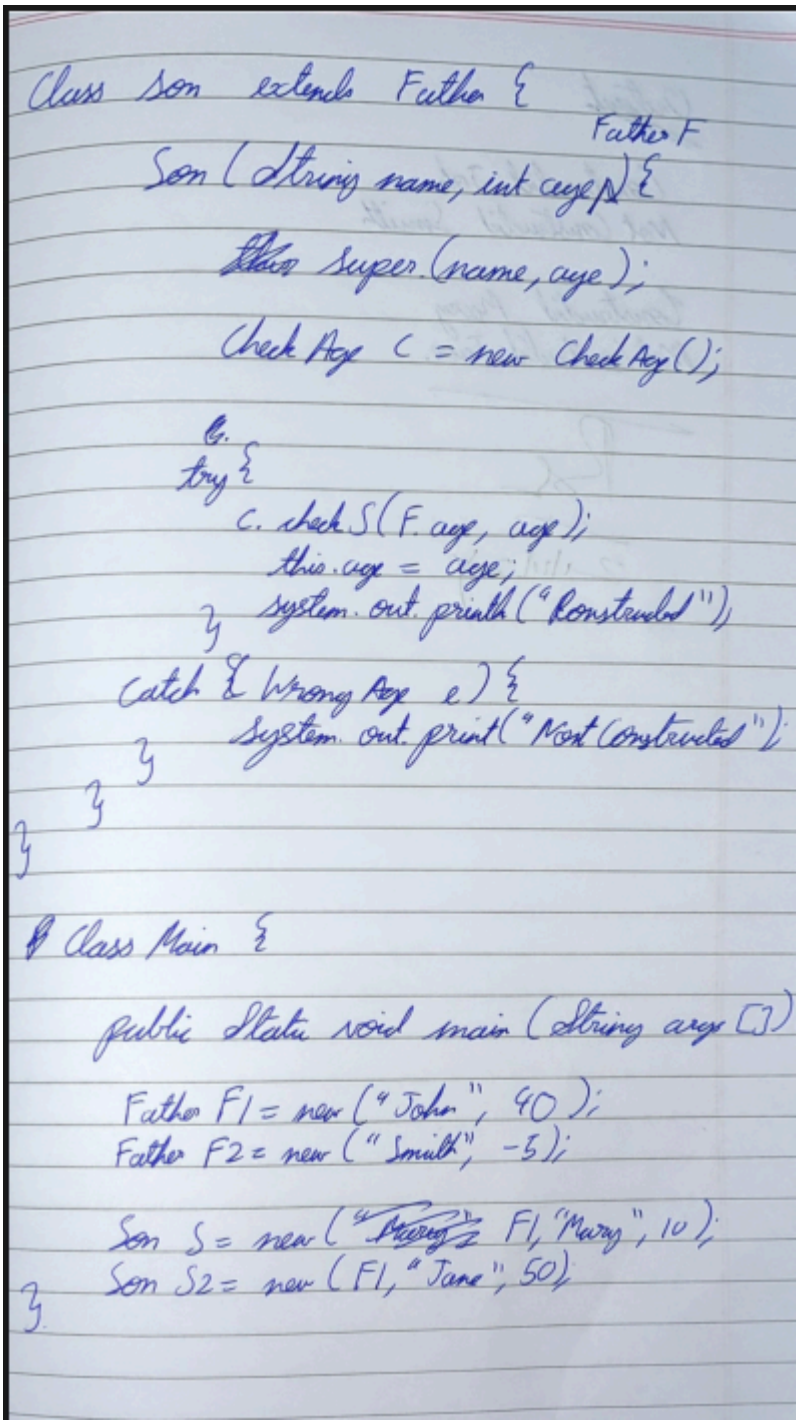
```
    }
}


class Son extends Father {
    int sonAge;
    Son(int i, int Father) throws WrongAge, MismatchedAge {
        super(Father);
        if (i < 0) {
            throw new WrongAge();
        }
        else if (i > super.age) {
            throw new MismatchedAge();
        }
        else {
            sonAge = age;
        }
    }
}

class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        try {
            System.out.print("Enter Father's age: ");
            int f1 = s.nextInt();
            Father f = new Father(f1);

            System.out.print("Enter Son's age: ");
            int s1 = s.nextInt();
            Son t = new Son(s1, f1);
            System.out.println("Worked");

            System.out.print("Enter Father's age: ");
            int f2 = s.nextInt();
            Father g = new Father(f2);

            System.out.print("Enter Son's age: ");
            int s2 = s.nextInt();
            Son u = new Son(s2, f2);
            System.out.println("Worked");
        } catch (WrongAge e) {
            System.out.println(e);
        } catch (MismatchedAge e) {
            System.out.println(e);
        }
    }
}
```


→ Lab 7 git:(master) X java Main
Enter Father's age: 50
Enter Son's age: -50
Input age cannot be less than 0
→ Lab 7 git:(master) X java Main
Enter Father's age: 50
Enter Son's age: 70
Father age cannot be less than child age
→ Lab 7 git:(master) X java Main
Enter Father's age: 50
Enter Son's age: 20
Worked
Enter Father's age: 50
Enter Son's age: 10
Worked
→ Lab 7 git:(master) X


Open with Google Docs
Problem: Write a program that demonstrates a inheritance tree. Create a base class Father, sub class Son. In father class's constructor, throw exception WrongAge() if age < 0; In Son, in the constructor, which take both father and son's age and validate them.
class WrongAge extends Exception {}
class CheckAge {
void checkS(int fage, int Sage) throws WrongAge {
if (fage <= Sage) {
throw new WrongAge();
}
}
void negCheck(int x) throws WrongAge {
if (x < 0) {
throw new WrongAge();
}
}



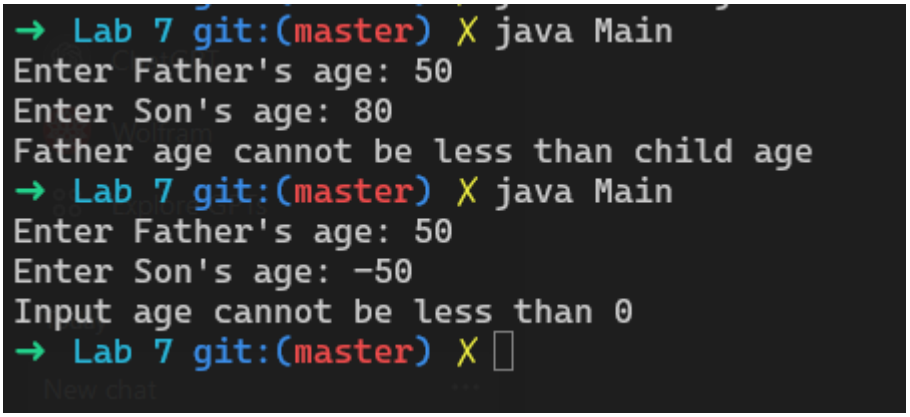
```
Class Father {

    String ~~age~~;Name;
    int age;

    Father (){};

    Father ( String name, int age ) {

        this.name = name;
        CheckAge c = new CheckAge();
        ~~c.test(age);~~

        try {
                c.test(age);
                this.age = age;                    + name
            } System.out.println("constructed!");

        Catch( WrongAge e ) {

                                                   + name
            System.out.println("Age Not constructed");
        }
        age=0;
    }
}
```




```
import java.util.Scanner;

class WrongAge extends Exception {
   public String toString() {
      return "Input age cannot be less than 0";
   }
}

class MismatchedAge extends Exception {
   public String toString() {
      return "Father age cannot be less than child age";
   }
}
```

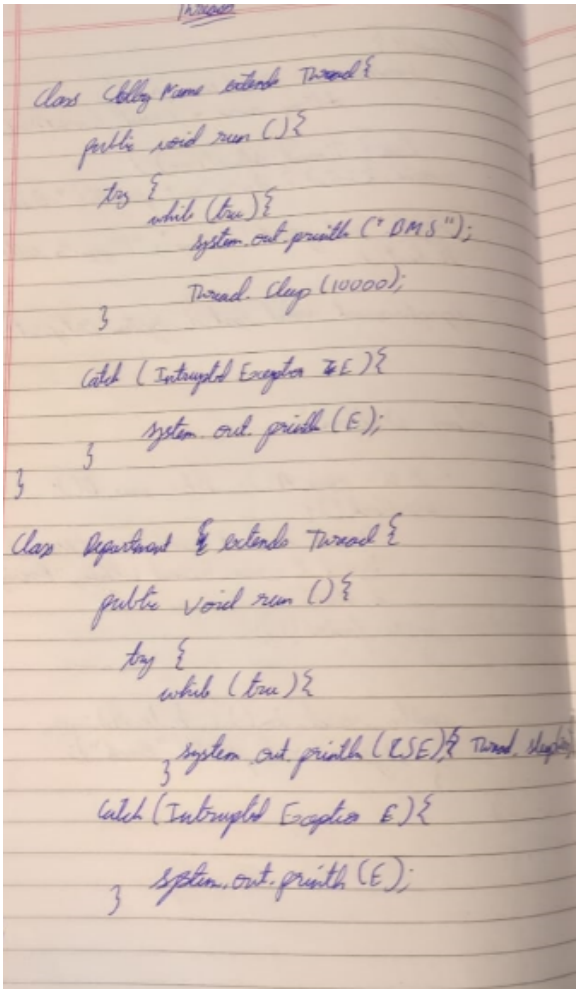
```
class Father {
   int age;
   Father() {};
   Father(int i) throws WrongAge {
      if (i < 0) {
         throw new WrongAge();
      }
      else {
         this.age = i;
      }

   }
}

class Son extends Father {
   int sonAge;
   Son(int i, int Father) throws WrongAge, MismatchedAge {
      super(Father);
      if (i < 0) {
         throw new WrongAge();
      }
      else if (i > super.age) {
         throw new MismatchedAge();
      }
      else {
         sonAge = age;
      }
   }
}

class Main {
   public static void main(String[] args) {
      Scanner s = new Scanner(System.in);
      try {
         System.out.print("Enter Father's age: ");
         int f1 = s.nextInt();
         Father f = new Father(f1);

         System.out.print("Enter Son's age: ");
         int s1 = s.nextInt();
         Son t = new Son(s1, f1);
         System.out.println("Worked");

         System.out.print("Enter Father's age: ");
         int f2 = s.nextInt();
         Father g = new Father(f2);

         System.out.print("Enter Son's age: ");
         int s2 = s.nextInt();
         Son u = new Son(s2, f2);
         System.out.println("Worked");
      } catch (WrongAge e) {
         System.out.println(e);
      } catch (MismatchedAge e) {
         System.out.println(e);
```

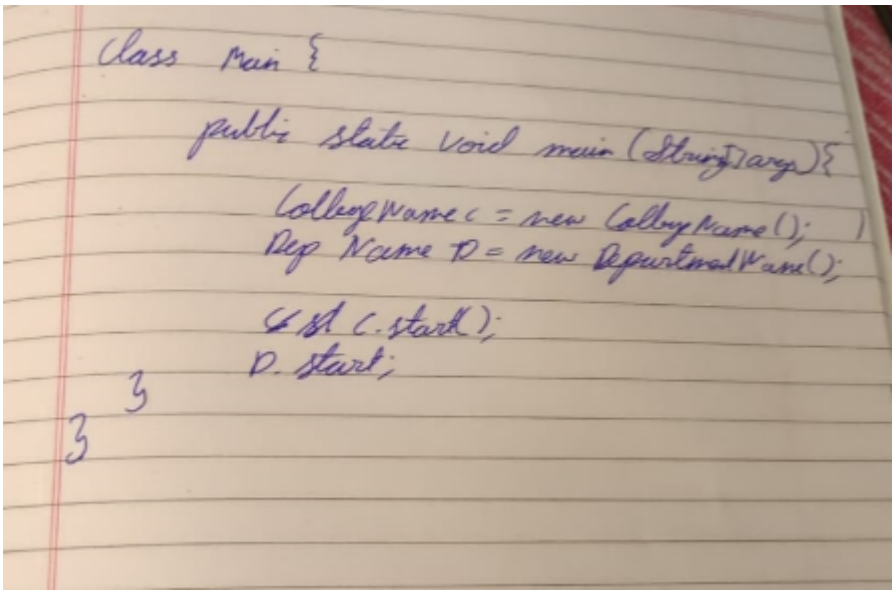
```
    }
  }
}
```


→ Lab 7 git:(master) X java Main
Enter Father's age: 50
Enter Son's age: 80
Father age cannot be less than child age
→ Lab 7 git:(master) X java Main
Enter Father's age: 50
Enter Son's age: -50
Input age cannot be less than 0
→ Lab 7 git:(master) X

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.






```
class CollegeName 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 DepartmentName extends Thread {
   public void run() {
      try {
         while (true) {


            System.out.println("CSE");
            Thread.sleep(2000);
         }
      }
      catch (InterruptedException e) {
         System.out.println(e);
      }
   }
}


class Main {
   public static void main(String[] args) {
      CollegeName c = new CollegeName();
      DepartmentName d = new DepartmentName();

      System.out.println("Printing college name then department name with delay: \n");
```

```
        c.start();
        d.start();
    }
}
```

```
→ threads git:(master) X java Main
Printing college name then department name with delay:

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
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
```

Program 9

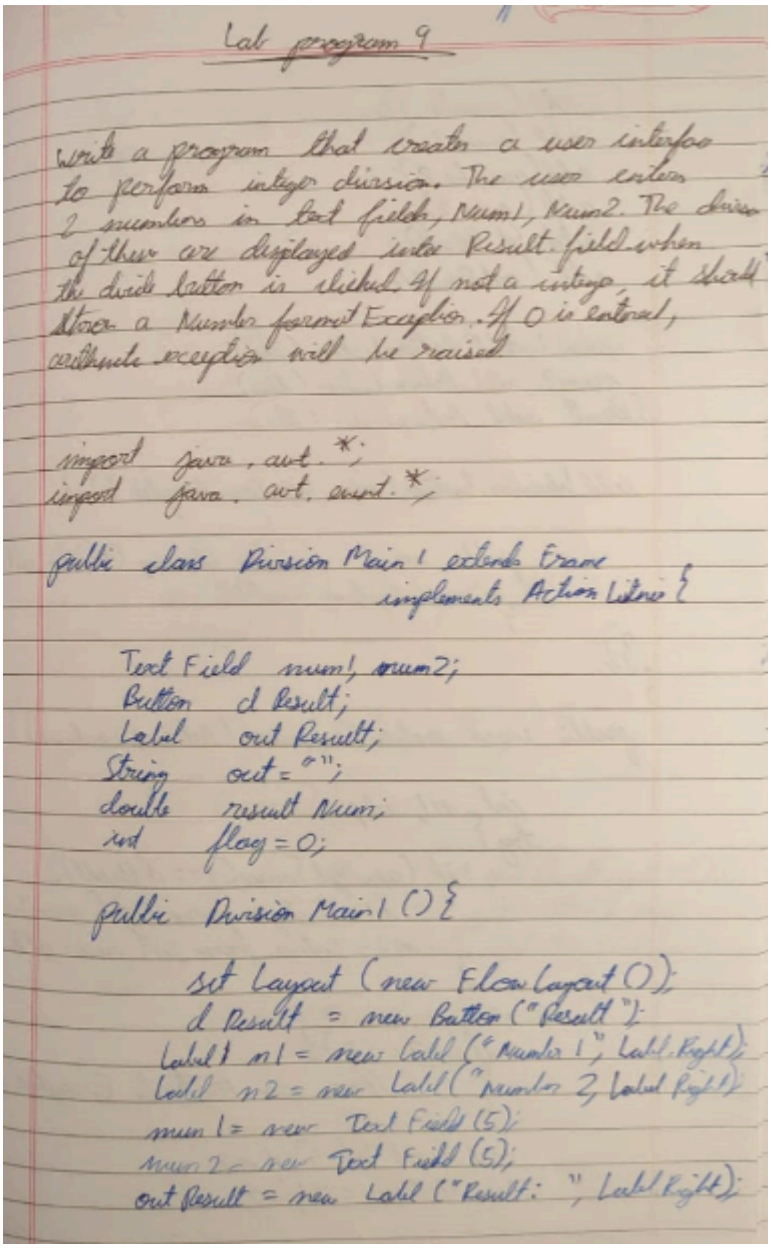
Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException.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

```
import
java.awt.*;
import
java.awt.ev
ent.*;

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

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

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

add(number1);
add(num1);
add(number2);
add(num2);
add(dResult);
add(outResult);

num1.addActionListener(this);
num2.addActionListener(this);
dResult.addActionListener(this);
addWindowListener(new WindowAdapter()
{
        public void windowClosing(WindowEvent we)
        {
         System.exit(0);
        }
});
}
public void actionPerformed(ActionEvent ae)
```

```
{
        int n1,n2;try
        {
                if (ae.getSource() == dResult)
                {
                        n1=Integer.parseInt(num1.getText());
                        n2=Integer.parseInt(num2.getText());

                        /*if(n2==0)
                                throw new
                        ArithmeticException();*/
                        out=n1+" "+n2+" ";
                        resultNum=n1/n2;
                        out+=String.valueOf
                        (resultNum);
                        repaint();

                }
        }
        catch(NumberFormatException e1)
        {
                flag=1;
                out="Number Format
                Exception! "+e1; repaint();
        }
        catch(ArithmeticException e2)
        {
                flag=1;
                out="Divide by 0
                Exception! "+e2;
```

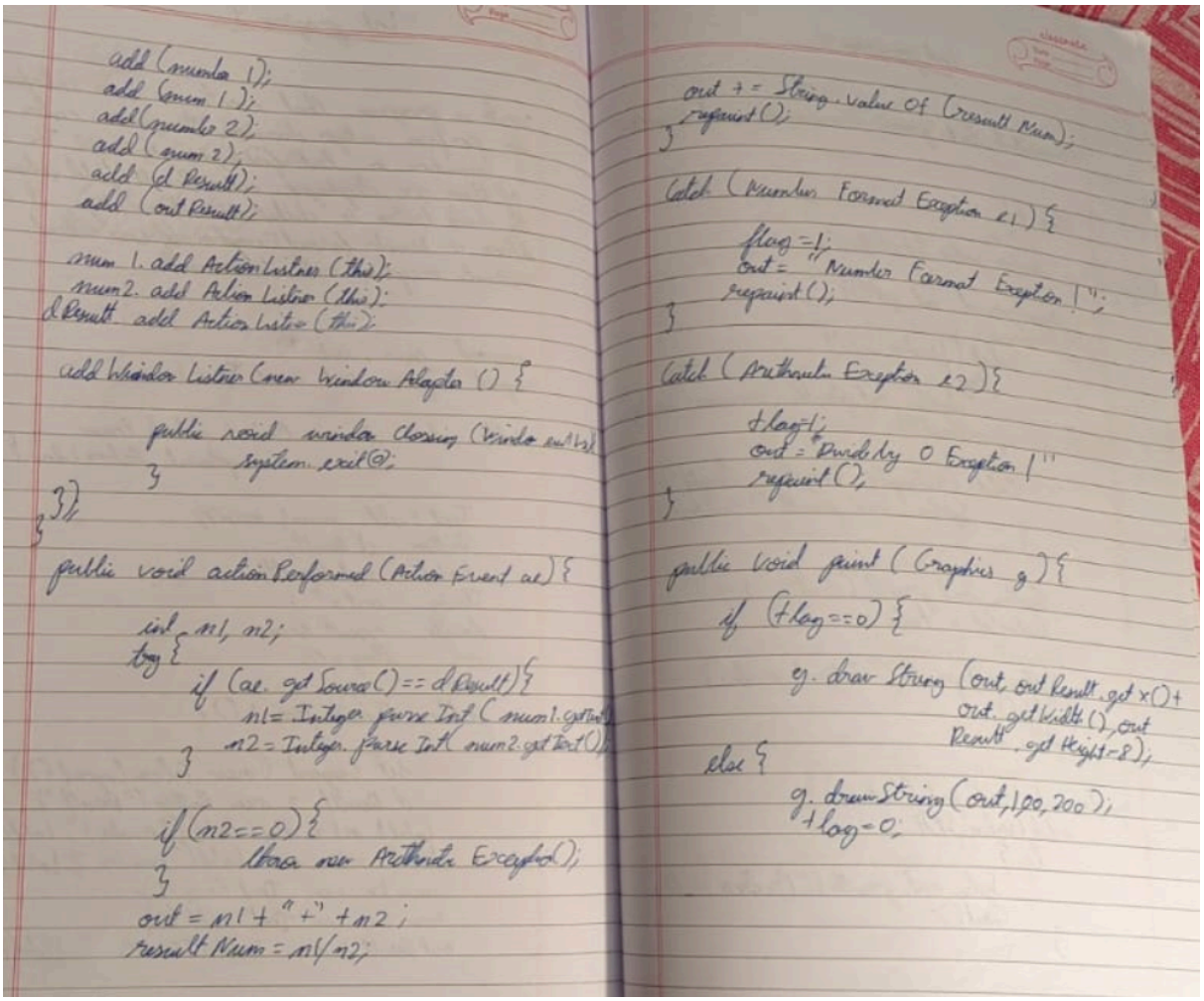
```
                repaint();
            }

        }
        public void paint(Graphics g)
        {
            if(flag==0)
            g.drawString(out,outResult.getX()+outResult.getWidth(
            ),outResult.getY()+outResult. getHeight()-8);
            else
            g.drawString
            (out,100,200)
            ; flag=0;
        }
```



Lab program 9

write a program that creates a user interface to perform integer division. The user enters 2 numbers in text fields, Num1, Num2. The division of them are displayed in Result field when the divide button is clicked. If not a integer, it should throw a NumberFormatException. If 0 is entered, an ArithmeticException will be raised

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

public class DivisionMain1 extends Frame
                    implements ActionListener{

    TextField num1, num2;
    Button d Result;
    Label outResult;
    String out = "";
    double result Num;
    int flag = 0;

    public DivisionMain1 (){

        set Layout (new FlowLayout());
        d Result = new Button ("Result");
        Label n1 = new Label ("Number 1", Label.Right);
        Label n2 = new Label ("Number 2, Label.Right);
        num1 = new TextField (5);
        num2 = new TextField (5);
        outResult = new Label ("Result: ", Label.Right);
```



```
add (number 1);
add (num 1);
add (number 2);
add (num 2);
add (d Result);
add (out Result);

num 1. add Action Listner (this);
num 2. add Action Listner (this);
d Result. add Action Listner (this);

add Window Listner (new Window Adapter () {

        public void windowClosing (WindowEvent we){
            System. exit(0);
        }
});

public void actionPerformed (ActionEvent ae){

    int n1, n2;
    try{
        if (ae. getSource() == d Result){
            n1 = Integer. parseInt (num1. getText());
            n2 = Integer. parseInt num2. getText());
        }

        if (n2==0){
            throw new ArithmeticException();
        }
    out = n1 + " " + n2;
    result Num = n1/n2;
```

```
    out += String. valueOf (result Num);
    repaint();
}

catch (NumberFormatException e1){

    flag=1;
    out = "Number Format Exception !";
    repaint();
}

catch (ArithmeticException e2){

    flag=1;
    out = "Divide by 0 Exception !"
    repaint();
}

public void paint (Graphics g){
    if (flag==0){

        g. drawString (out, out Result. getX()+
                        out. getWidth(), out
                        Result. getHeight-8);
    else{
        g. drawString (out, 100, 200);
        flag=0;
```

Program 10

Demonstrate Interprocess communication and deadlock

```
class Q {
int n;
boolean valueSet = false;

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

synchronized void put(int n) {
while(valueSet)
try {
System.out.println("\nProducer waiting\n");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("\nIntimate Consumer\n");
notify();
}
}

class Producer implements Runnable {
Q q;
Producer(Q q) {
this.q = q;
new Thread(this, "Producer").start();
}
public void run() {
int i = 0;
while(i<15) {
q.put(i++);
}
}
}

class Consumer implements Runnable {
Q q;
Consumer(Q q) {
this.q = q;
```

```
new Thread(this, "Consumer").start();
}
public void run() {
            int i=0;
while(i<15) {
int r=q.get();
System.out.println("consumed:"+r);
i++;
}
}
}

class PCFixed {
public static void main(String args[]) {
Q q = new Q();
new Producer(q);
new Consumer(q);
System.out.println("Press Control-C to stop.");
}
}
```

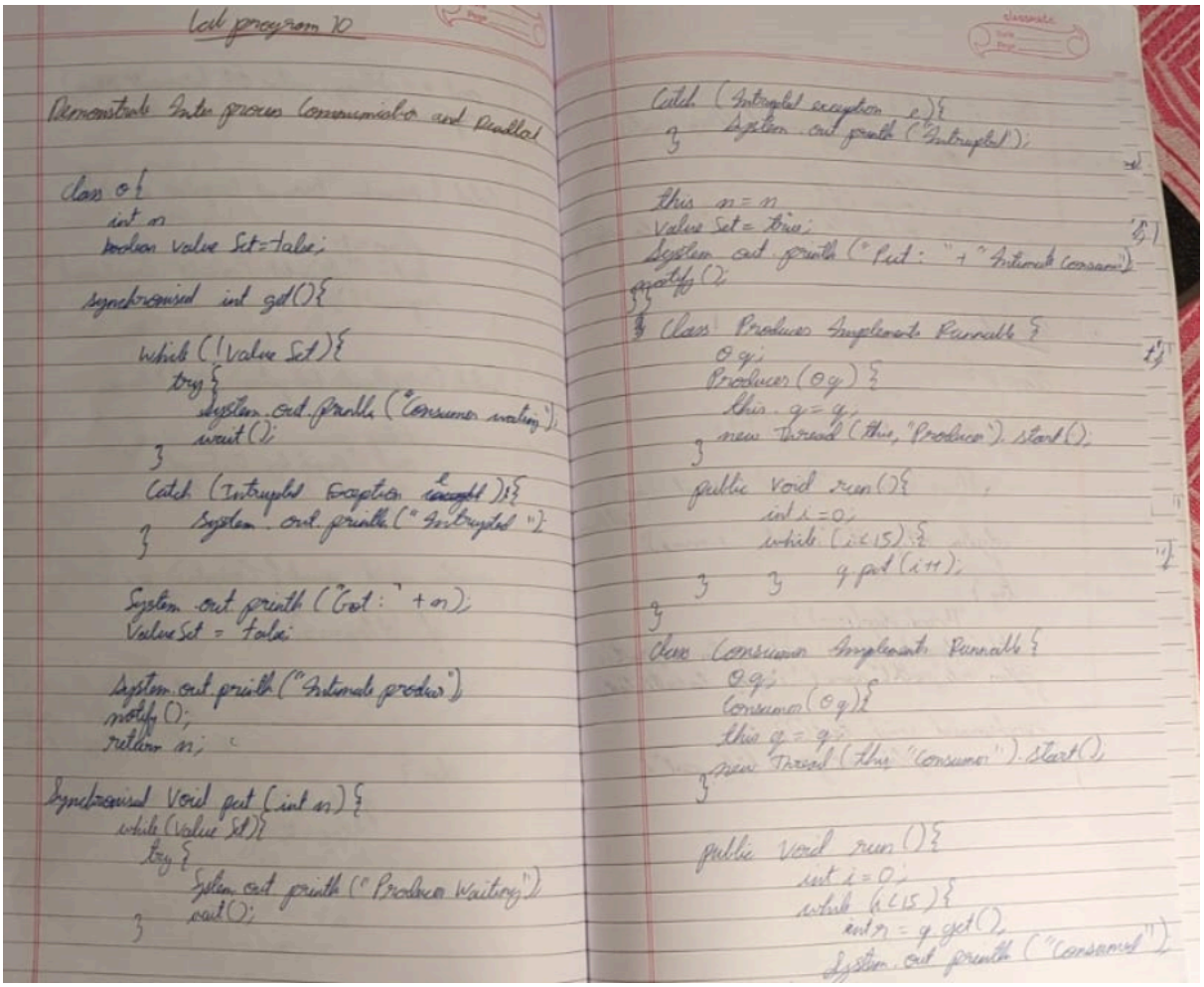
OUTPUT

ii. Demonstration of deadlock

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

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

}


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

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

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



Lab program 10

Demonstrate Inter process Communication and Deadlock

```
class Q{
    int n;
    boolean ValueSet=false;

    synchronized int get(){
        while(!ValueSet){
            try{
                System.out.println("Consumer waiting");
                wait();
            }
            Catch (InterruptedException e){
                System.out.println("Interrupted");
            }

        System.out.println("Got: " + n);
        ValueSet = false;

        System.out.println("Intimate producer");
        notify();
        return n;

    Synchronized Void put(int n){
        while(ValueSet){
            try{
                System.out.println("Producer Waiting");
                wait();
            }
            Catch (InterruptedException e){
                System.out.println("Interrupted");
            }

        this.n=n;
        ValueSet=true;
        System.out.println("Put: " + 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");
        }
    }
}

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");
    }
}
```

ii) Deadlock

```
Class A{

    Synchronized void foo(B b){

        String name = Thread.currentThread().getName();

        System.out.println("Entered " + name);

        try{
            Thread.sleep(1000);}
        Catch (InterruptedException e){System.out.println(e);}
        System.out.println(name + "Trying to call B.last()");

        Synchronized void last(){
            System.out.println("A.last");
        }
    }

Class B{
    synchronized void bar(A a){
        String name = Thread.currentThread().getName();

        try{Thread.sleep(1000);}
        catch(IE e){System.out.println("B interrupted");}

        System.out.println(name + "Trying to call A");
        A.last();

        synchronized void last(){System.out.println("A.last");}
    }

class Deadlock{

    A a=new A(); B b=new B();
    Deadlock(){
        Thread.currentThread().setName("Main");
        Thread t=new Thread(this,"Racing");
        t.start();
        a.foo(b);
    }

    public void run(){b.bar(a);
        System.out.println("Back");
    }
}
```