VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SUHAS BP (1BM23CS345)

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 **SUHAS BP (1BM23CS345),** 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

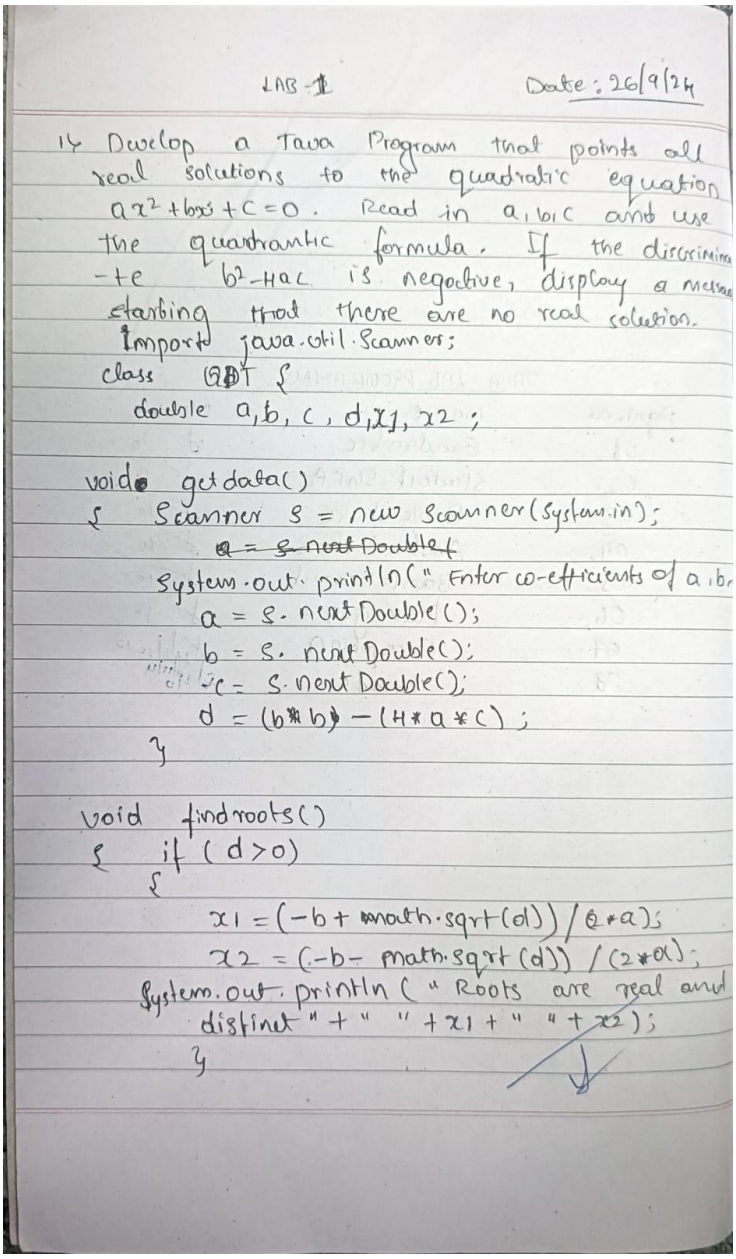
Associate Professor,
Department of CSE,
BMSCE, Bengaluru

Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

INDEX

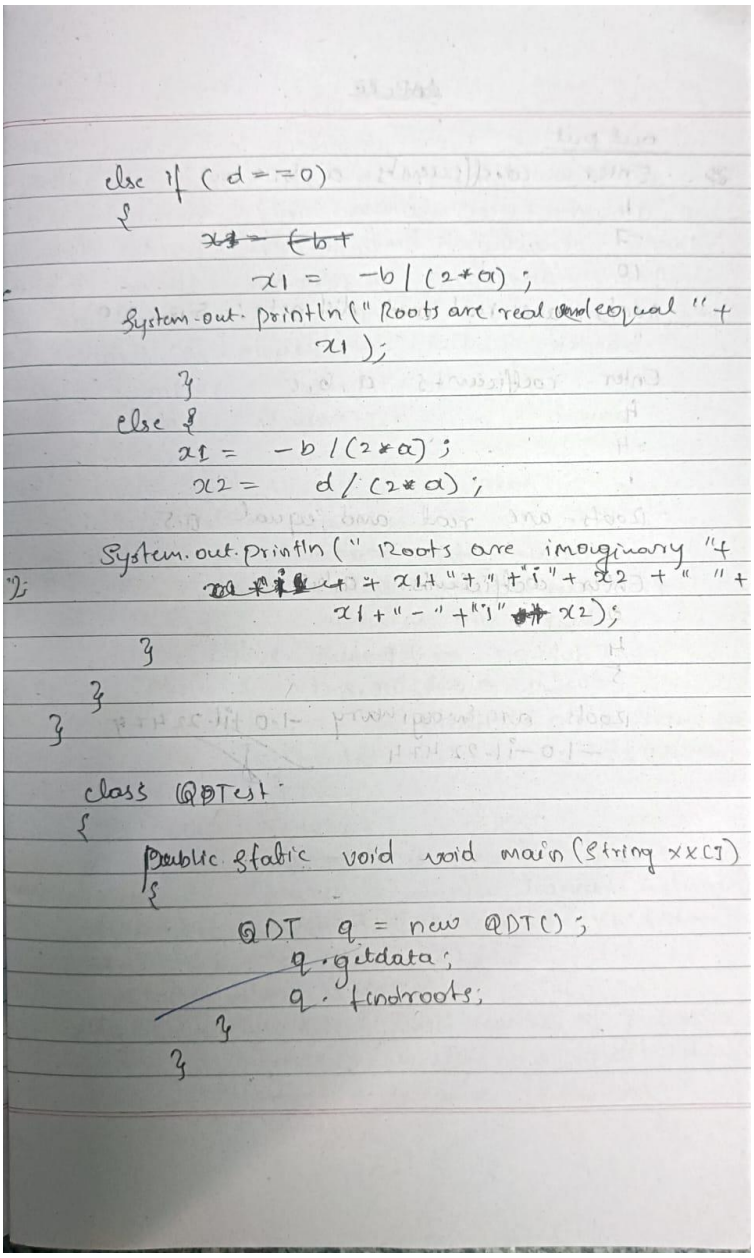
1. Develop a Java program that prints all real solutions to the quadratic equation ax2+bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminate b2-4ac is negative, display a message stating that there are no real solutions.



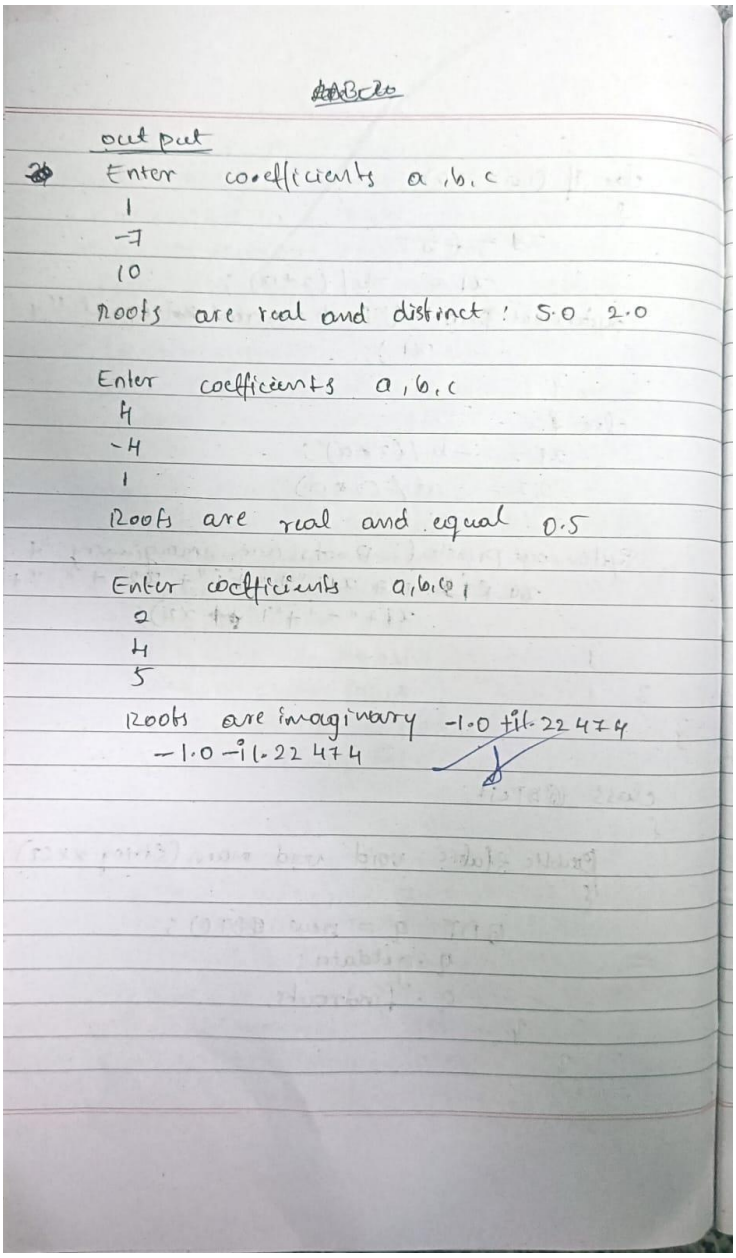
LAB - 1 Date: 26/9/24

1) Develop a Java Program that points all real solutions to the quadratic equation $ax^2+bx+c=0$. Read in a, b, c and use the quadratic formula. If the discrimina-te b^2-4ac is negative, display a messa starting that there are no real solution.

```
import java.util.Scanner;
class QDT {
  double a,b,c,d,x1,x2;

void getdata()
{  Scanner s = new Scanner(System.in);
   a = s.nextDouble();
   System.out.println("Enter co-efficients of a,b,
     a = s.nextDouble();
     b = s.nextDouble();
     c = s.nextDouble();
     d = (b*b)-(4*a*c);
  }

void findroots()
{  if (d>0)
   {
     x1 = (-b+Math.sqrt(d))/(2*a);
     x2 = (-b-Math.sqrt(d))/(2*a);
   System.out.println("Roots are real and
     distinct"+" "+x1+" "+x2);
  }
```



```
else if (d==0)
{
    x1 = -b/(2*a);
System.out.println("Roots are real and equal "+
                    x1);
}
else {
    x1 = -b/(2*a);
    x2 = d/(2*a);
System.out.println("Roots are imaginary "+
                    x1+"+"+"i"+x2+" "+
                    x1+"-"+"i"+x2);
}
}
}

class QDTest
{
public static void main(String xx[])
{
    QDT q = new QDT();
    q.getdata;
    q.findroots;
}
}
```



output

Enter coefficients a, b, c
1
-7
10
Roots are real and distinct : 5.0 2.0

Enter coefficients a, b, c
4
-4
1
Roots are real and equal 0.5

Enter coefficients a, b, c
2
4
5
Roots are imaginary -1.0 +i1.22474
-1.0 -i1.22474

```
import java.util.Scanner;

class QDT { double a;

double b;

double c;

double d;
```

```
 double x1;

double x2;

void getData() {
        Scanner s = new Scanner(System.in); System.out.println("Enter  coefficients a,
        b, and c: ");
        a = s.nextDouble();
        b = s.nextDouble();
        c = s.nextDouble();
        d = b * b - 4 * a * c;
}

void findRoots() { if (d >
        0) {
                x1 = (-b + Math.sqrt(d)) / (2 * a);
                x2 = (-b - Math.sqrt(d)) / (2 * a); System.out.println("Roots are real and distinct:
                " + x1 + " "
+ x2);
        } else if (d == 0) { x1 = -b / (2 *
                a);
                System.out.println("Roots  are real and equal: " + x1);
        } else {
                double realPart = -b / (2 * a);
                double imaginaryPart = Math.sqrt(-d) / (2 * a); System.out.println("Roots are
                imaginary: " + realPart + " + "
+ imaginaryPart + "i and "
                        + realPart + " - " + imaginaryPart + "i");
        }
}


}

class QDTest {

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

```
QDT q = new QDT();

q.getData();
```

q.findRoots(); } }

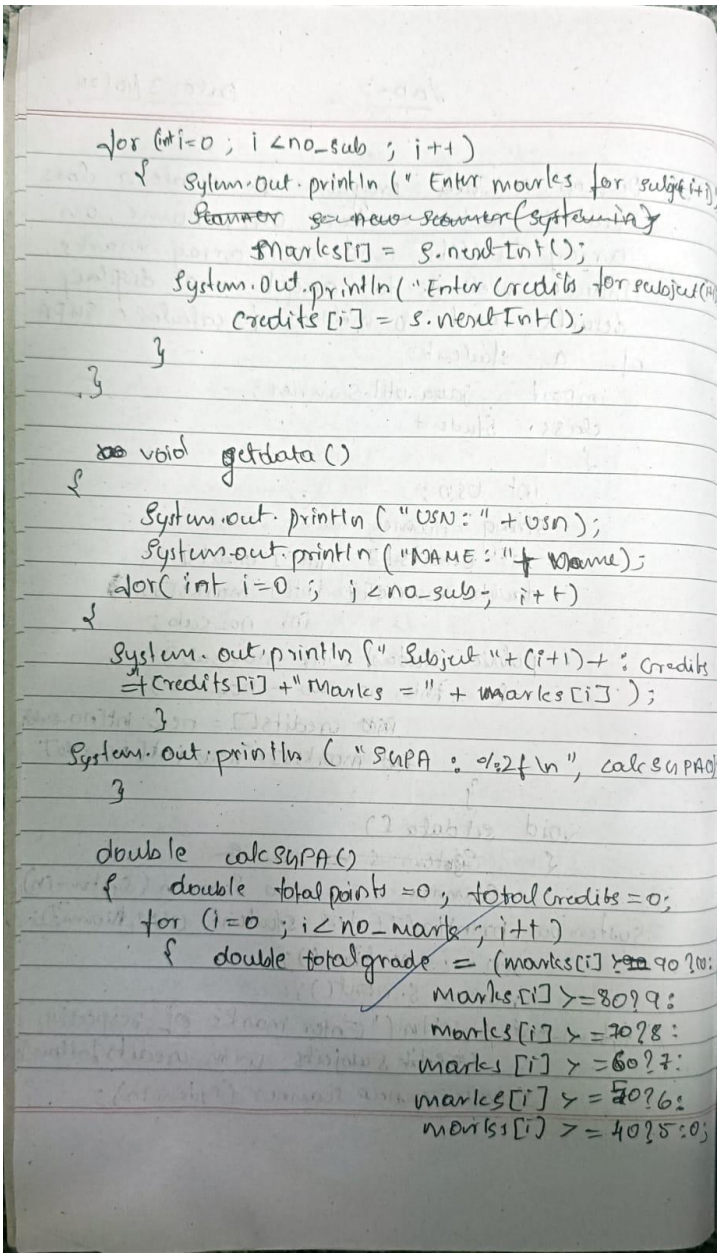
```
D:\Suhas>javac QDTest.java

D:\Suhas>java QDTest
Enter coefficients a, b, and c:
1
-7
10
Roots are real and distinct: 5.0 2.0

D:\Suhas>java QDTest
Enter coefficients a, b, and c:
4
-4
1
Roots are real and equal: 0.5

D:\Suhas>java QDTest
Enter coefficients a, b, and c:
2
4
5
Roots are imaginary: -1.0 + 1.2247448713915891 and -1.0 - 1.2247448713915891
```

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.

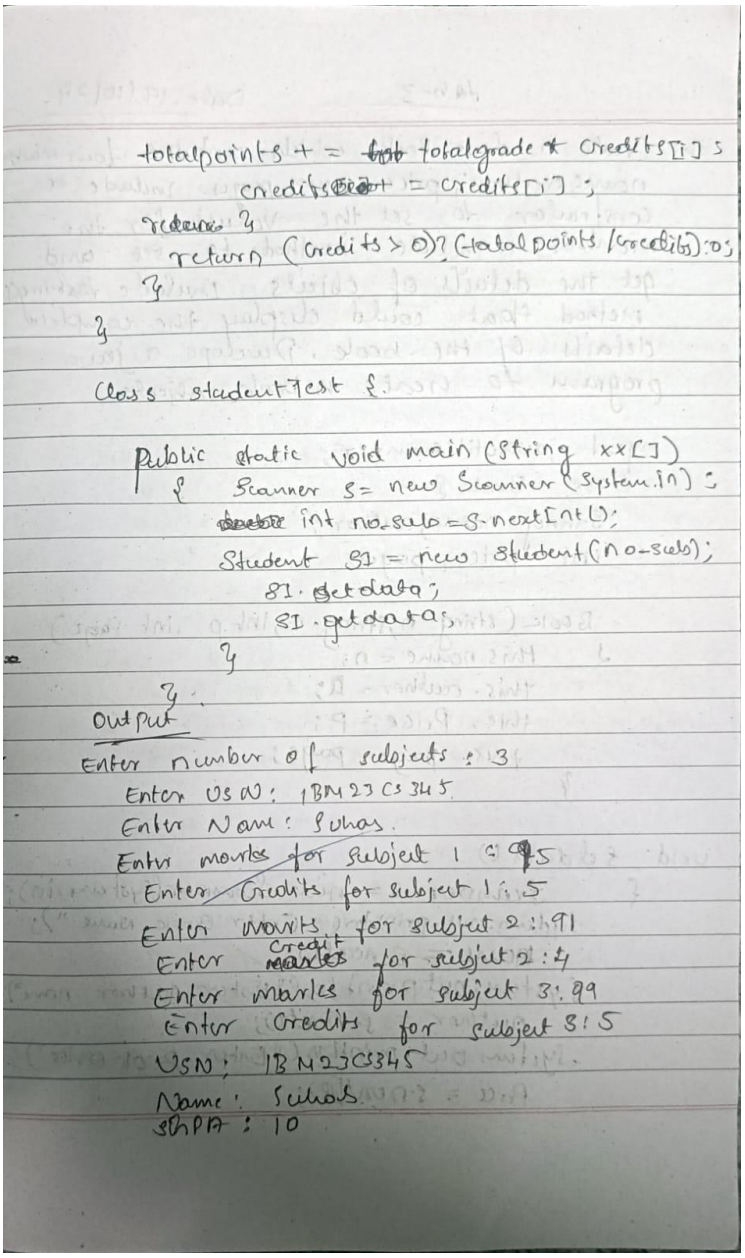
Lab-2 Date: 3/10/24

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
{
    int usn;
    String name;
    int[] credits[];
    int[] marks[];
                int no_sub;
    public Student(int no_sub)
    { this.no_sub = no_sub;
        credits[] = new int[no_sub]
        marks[] = new int[no_sub];
    }
void setdata()
{   Scanner s = new Scanner(System.in)
System.out.println("Enter student USN, Name");
    usn = s.nextInt();
    name = s.next();
// System.out.println("Enter marks of respective
        Subjects with credits follows");
    Scanner s = new Scanner(System.in);
```



```
for (int i=0 ; i<no_sub ; i++)
  {  System.out.println (" Enter marks for subject i+1)
     Scanner s= new Scanner(System.in);
        marks[i] = s.nextInt();
     System.out.println(" Enter credits for subject(i+1)
        credits[i] = s.nextInt();
  }
}

void getdata()
{
   System.out.println ("USN: " + usn);
   System.out.println ("NAME: " + name);
   for( int i=0 ; i<no_sub; i++)
{
   System.out.println (" Subject "+(i+1)+ : Credits
   =+credits[i] +" Marks = " + marks[i] );
  }
System.out.println ( "SGPA : %.2f \n", calcSGPA()
}

double calcSGPA()
{  double totalpoints =0 , totalCredits =0;
   for (i=0 ; i<no_marks ; i++)
   {  double totalgrade = (marks[i] >=90?10:
                           marks[i] >=80?9:
                           marks[i] >=70?8:
                           marks[i] >=60?7:
                           marks[i] >=50?6:
                           marks[i] >=40?5:0;
```



```
totalpoints += totalgrade * Credits[i];
credits += credits[i];
}
return (credits > 0)? (totalpoints/credits):0;
}
}

Class studentTest {

Public static void main (String xx[])
{ Scanner s= new Scanner(System.in);
int no_subs=s.next[nt();
Student S1 = new student(no_subs);
S1.getdata;
S1.getdata;
}
}
```

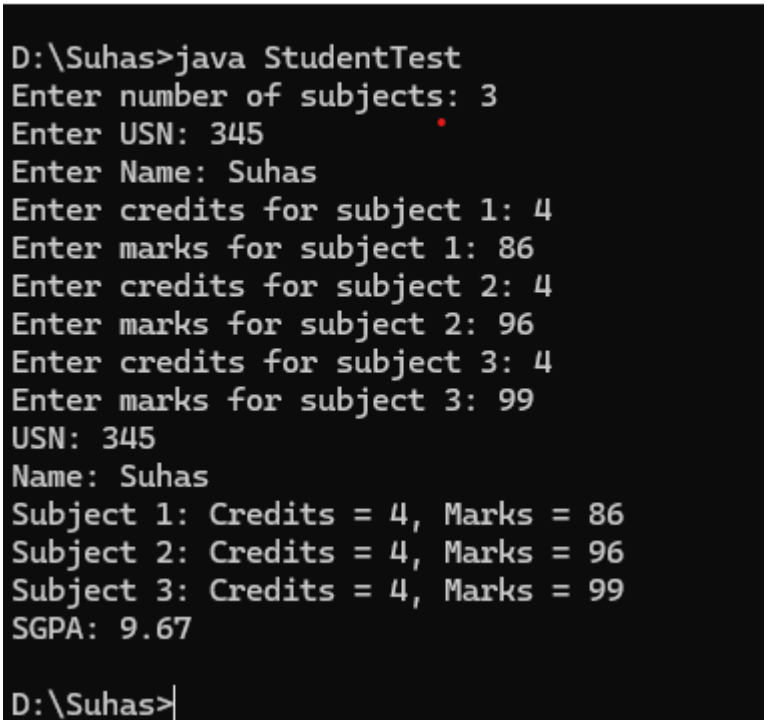
Output

```
Enter number of subjects : 3
Enter USN: 1BM23CS345
Enter Name: Suhas.
Enter marks for subject 1 : 95
Enter Credits for subject 1 : 5
Enter marks for subject 2 : 91
Enter Credit marks for subject 2 : 4
Enter marks for subject 3: 99
Enter Credits for subject 3: 5
USN: 1BM23CS345
Name: Suhas
SGPA : 10
```

```
import java.util.Scanner;

class Student { String usn;

String name;

 int[] credits;

int[] marks;

int num_Sub;
```

```
public Student(int num_Sub) { this.num_Sub
        = num_Sub; credits = new
        int[num_Sub]; marks = new
        int[num_Sub];
}

void Details() {
        Scanner s = new Scanner(System.in);

        System.out.print("Enter USN: "); usn =
        s.nextLine(); System.out.print("Enter Name:
        "); name = s.nextLine();

        for (int i = 0; i < num_Sub; i++) {
                System.out.print("Enter credits for subject " + (i + 1) + ": credits[i] = s.nextInt();
");
                System.out.print("Enter marks for subject " + (i + 1) + ": "); marks[i] = s.nextInt();
        }
}

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

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

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

```
            double gradePoint = (marks[i] >= 90) ? 10 :
                                (marks[i] >= 80) ? 9 :
                                (marks[i] >= 70) ? 8 :
                                (marks[i] >= 60) ? 7 :
                                (marks[i] >= 50) ? 6 :
                                (marks[i] >= 40) ? 5 : 0;

            totalPoints += gradePoint * credits[i]; totalCredits += credits[i];
        }

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


}

class StudentTest {

public static void main(String[] args) {

 Scanner scanner = new Scanner(System.in);

      System.out.print("Enter  number of subjects: "); int num_Sub =
        scanner.nextInt();


        Student student = new Student(num_Sub);
        student.Details(); student.displayDetails();
}


}
```



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.

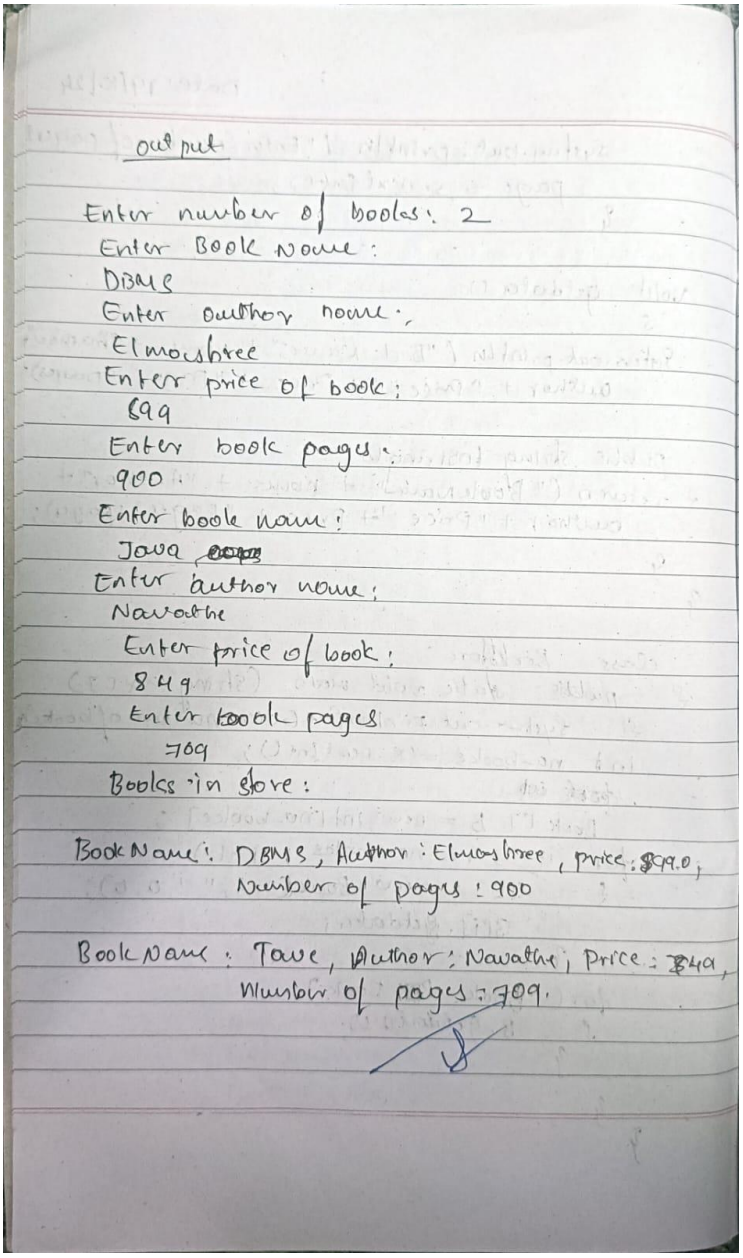
Lab-3 Date: 19/10/24

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 objects. Include toString() method that could display the complete details of the book. Develop a java program to create n book objects.

```
import java.util.Scanner;
class Book {
    String name, author;
    int price, pages;

    Book(String n, String a, int p, int Page)
    {
        this.name = n;
        this.author = a;
        this.Price = p;
        this.page = page;
    }

void setdata()
{
    Scanner s = new Scanner(System.in);
    System.out.println("Enter Book Name");
        Name = s.next();
    System.out.println("Enter Author name");
        author = s.next();
    System.out.println("Enter Book Price");
        Price = s.nextInt();
```

Date: 19/10/24

```
        System.out.println ("Enter number of pages");
            page = s.nextInt();
        }

    Void getdata()
        {
    System.out.println ("Book Name:" + name + "Author:"+
            author + "Price :" + Price + "Pages:" + pages);

        public String toString()
    { return ("BookName" + name + "Author:" +
            author + "Price :" + Price + "Pages" + page);
        }
    }

        class BookStore
    {   public static void main (String xx[])
        {   System.out.println("Enter number of books");
        int no_books = s.nextInt();
            Book[] B = new int[no_books];
        for( i=0; i< no_books; i++)
        {       B[] = new Book("","",0,0);
                B[i].getdata;
        }
        for( Book B: Book)
        {       B.getdata();
        }

    }
}
```


output

Enter number of books: 2
Enter Book Name:
DBMS
Enter author name:
Elmasbree
Enter price of book:
849
Enter book pages:
900
Enter book name:
Java
Enter author name:
Navathe
Enter price of book:
849
Enter book pages
709
Books in store:

BookName: DBMS, Author: Elmasbree, price: $849.0,
Number of pages: 900

BookName: Java, Author: Navathe, price: $849,
Number of pages: 709.

```
import java.util.Scanner;


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

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

void setdetails() {
    Scanner s = new Scanner(System.in);
    System.out.println("Enter  book name:");
    name = s.next();
    System.out.println("Enter  author name:");
    author = s.next();
    System.out.println("Enter  price of book:");
    price = s.nextDouble();
    System.out.println("Enter  book total pages:");
    numPages = s.nextInt();
}

void getdetails() {
    System.out.println("Book  Name: " + name + ", Author: " + author + ", Price: $" + price +
                ", Number of Pages: " + numPages);
}

public String toString() {
```

```
        return "Book Name: " + name + ", Author: " + author + ", Price: $" + price +
               ", Number of Pages: " + numPages;
    }
}

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

        System.out.print("Enter  the number of books: ");
        int n = scanner.nextInt();

        Book[] books = new Book[n];

        for (int i = 0; i < n; i++) {
            books[i] = new Book("", "", 0.0, 0);
            books[i].setdetails();
        }

        System.out.println("\nBooks  in the store:");
        for (Book book : books) {
            book.getdetails();
        }

        scanner.close();
    }
```

}

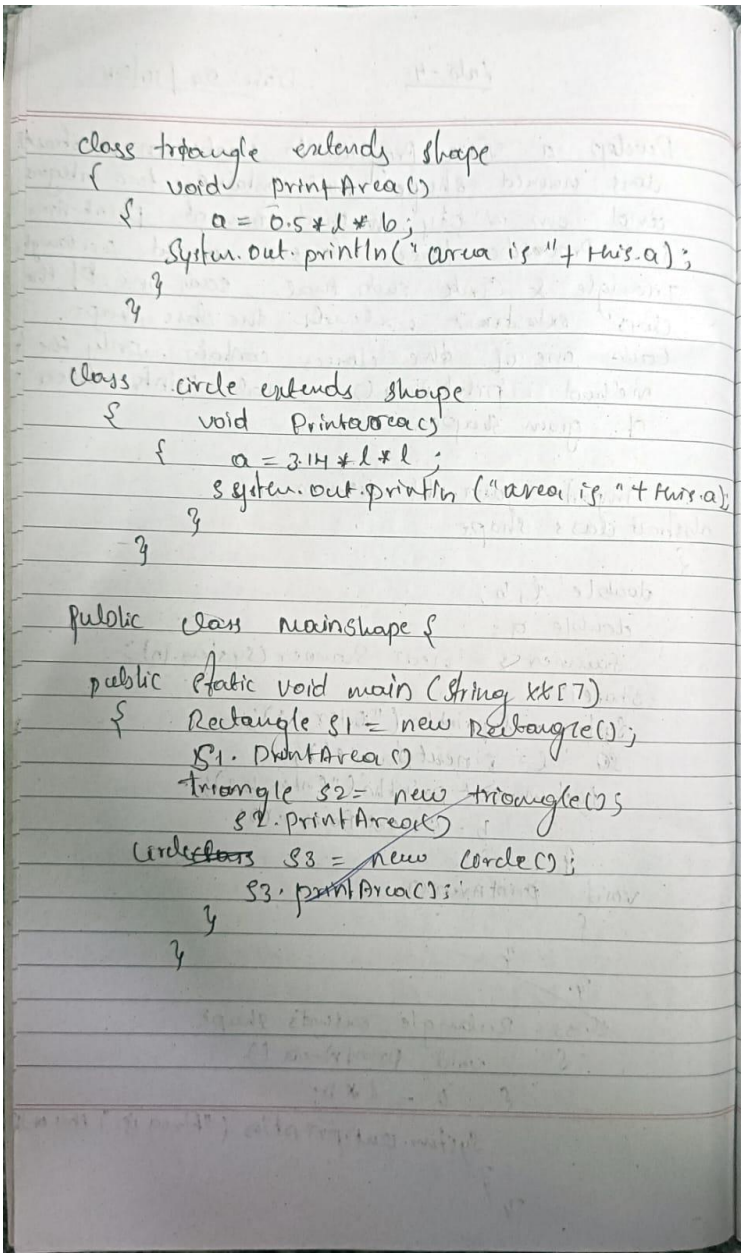
```
D:\Suhas>java BookStore
Enter the number of books: 3
Enter book name:
DBMS
Enter author name:
Elmashree
Enter price of book:
899
Enter book total pages:
900
Enter book name:
Java oops
Enter author name:
Enter price of book:
849
Enter book total pages:
709
Enter book name:
Vtfluria
Enter author name:
navathe
Enter price of book:
900
Enter book total pages:
1039

Books in the store:
Book Name: DBMS, Author: Elmashree, Price: $899.0, Number of Pages: 900
Book Name: Java, Author: oops, Price: $849.0, Number of Pages: 709
Book Name: Vtfluria, Author: navathe, Price: $900.0, Number of Pages: 1039
```

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.

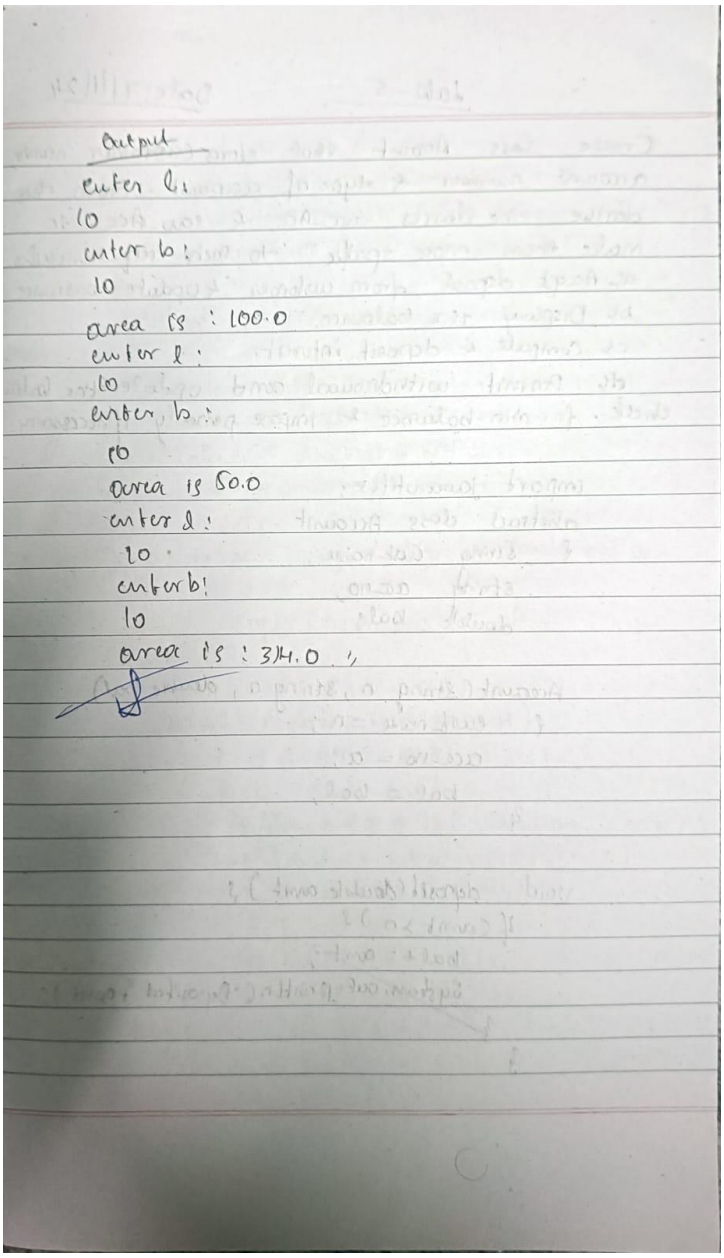
Lab-4 Date: 24/10/24

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 & Circle such that each one of the class extends the class shape. Each one of the classes contain only the method printArea() that prints area of given shape.

```
import java.util.Scanner;
abstract class shape
{
  double l, b;
  double a;
  Scanner s = new Scanner(System.in);
  shape()
  System.out.println("Enter l");
  l = s.nextDouble();
  System.out.println("Enter b");
  b = s.nextDouble();
  }
  void printArea()
  {
  }
class Rectangle extends shape
  {  void printArea()
     {  a = l*b;
        System.out.println("Area is "+this.a);
     }
}
```



```
class triangle extends shape
{   void printArea()
{     a = 0.5 * l * b;
   System.out.println("area is" + this.a);
  }
}

class circle extends shape
{   void printArea()
{     a = 3.14 * l * l;
   System.out.println("area is" + this.a);
  }
}

public class mainshape {

public static void main (String xx[])
{   Rectangle s1 = new Rectangle();
    s1.printArea()
   triangle s2 = new triangle();
     s2.printArea()
  Circle s3 = new circle();
     s3.printArea();
  }
}
```




```
import java.util.Scanner;

abstract class Shape {

int l, b;

 double a;

 Scanner ss = new Scanner(System.in);
```

```
Shape() {
        System.out.println("Enter  l:"); l =
        ss.nextInt(); System.out.println("Enter
        b:"); b = ss.nextInt();
}

void printArea() {
        // Abstract method to be overridden
}


}

class Rectangle extends Shape {

 void printArea()

 {

 a = l * b;

System.out.println("Area  of Rectangle is: " + this.a); } }

class Triangle extends Shape {

 void printArea() {

a = 0.5 * l * b;

 System.out.println("Area  of Triangle is: " + this.a); } }

class Circle extends Shape {

 void printArea() {

 a = 3.14 * l * l;

System.out.println("Area  of Circle is: " + this.a); } }

public class MainShape {

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

```
Rectangle s2 = new Rectangle();

Triangle s3 = new Triangle();

Circle s4 = new Circle();

      s2.printArea(); s3.printArea();
        s4.printArea();
}


}
```

```
D:\3 sem\java\java prgms>java MainShape
enter l:
10
enter b:
10
enter l:
10
enter b:
10
enter l:
10
enter b:
10
area is :100.0
area is :50.0
area is :314.0
```

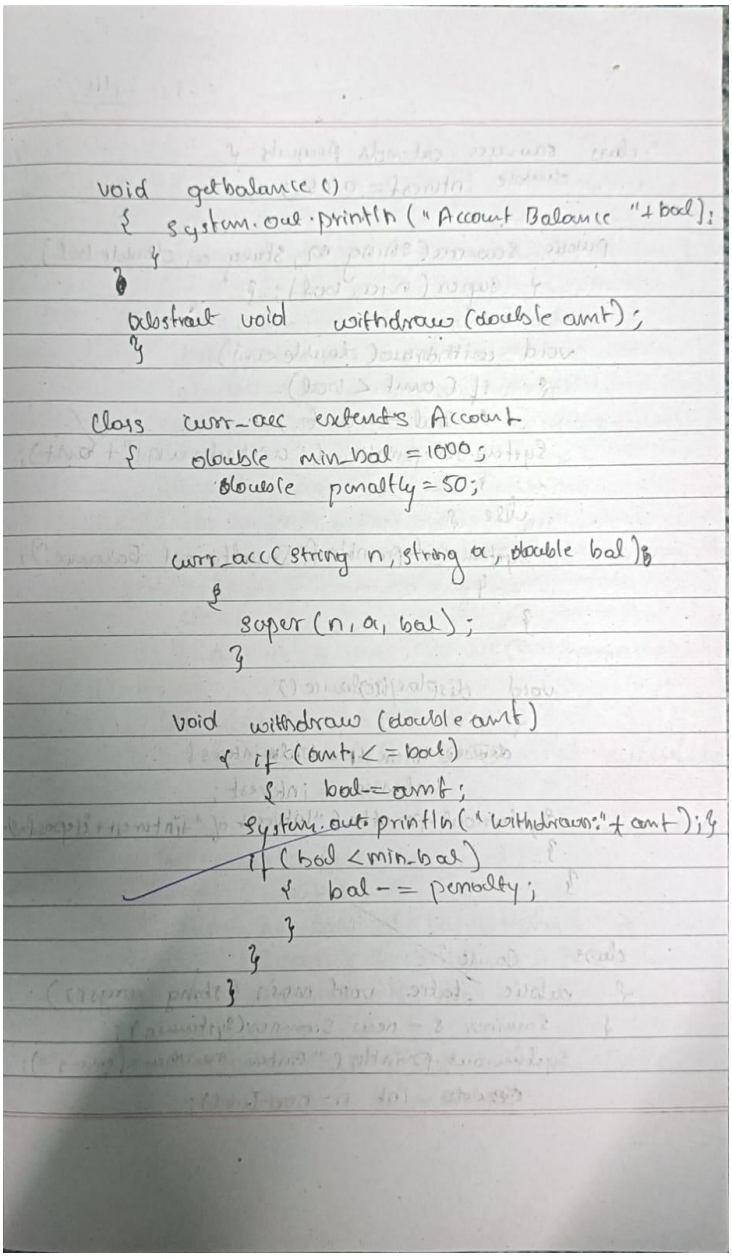
5. Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book 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.

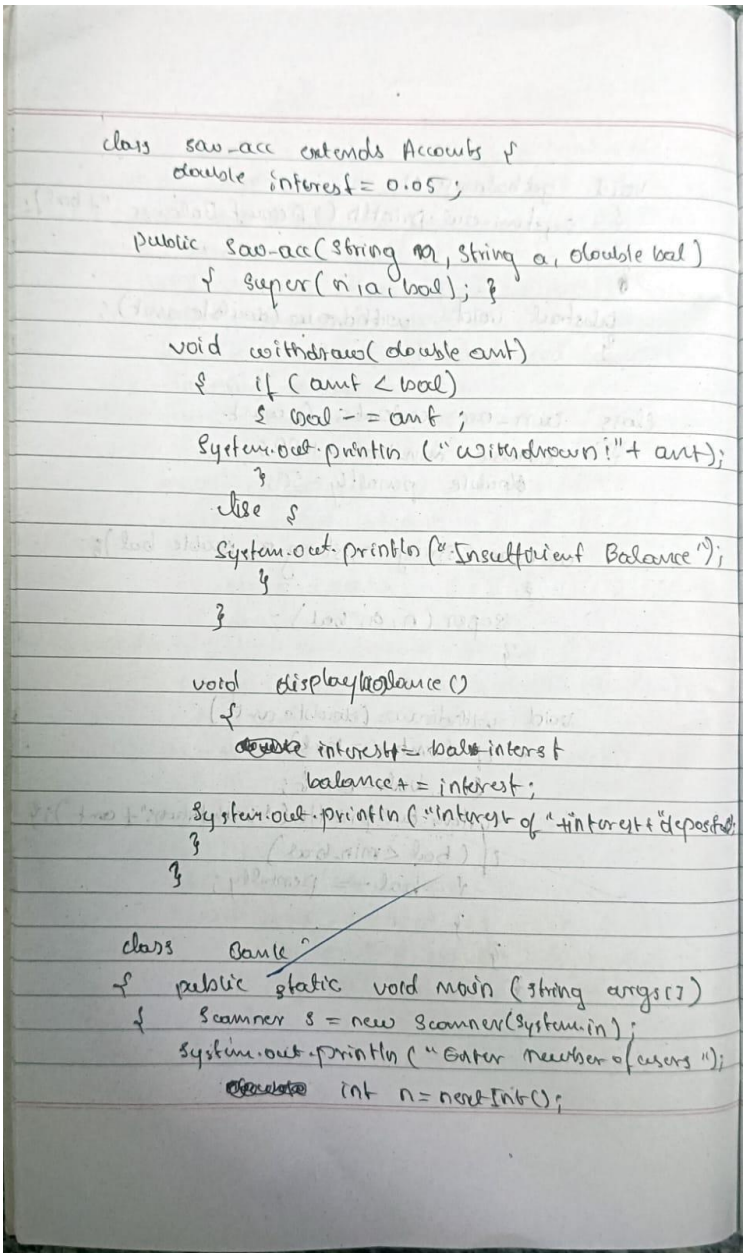
Lab-5 Date: 7/11/24

Create class Account that stores customers name, account number & type of account. From this derive the classes cur_Acc & sav_Acc to make them more specific to their requirements as a) Accept deposit from customers & update balance
b) Display the balance.
c) Compute & deposit interest.
d) Permit withdrawal and update the balance check for min balance & impose penalty if necessary.

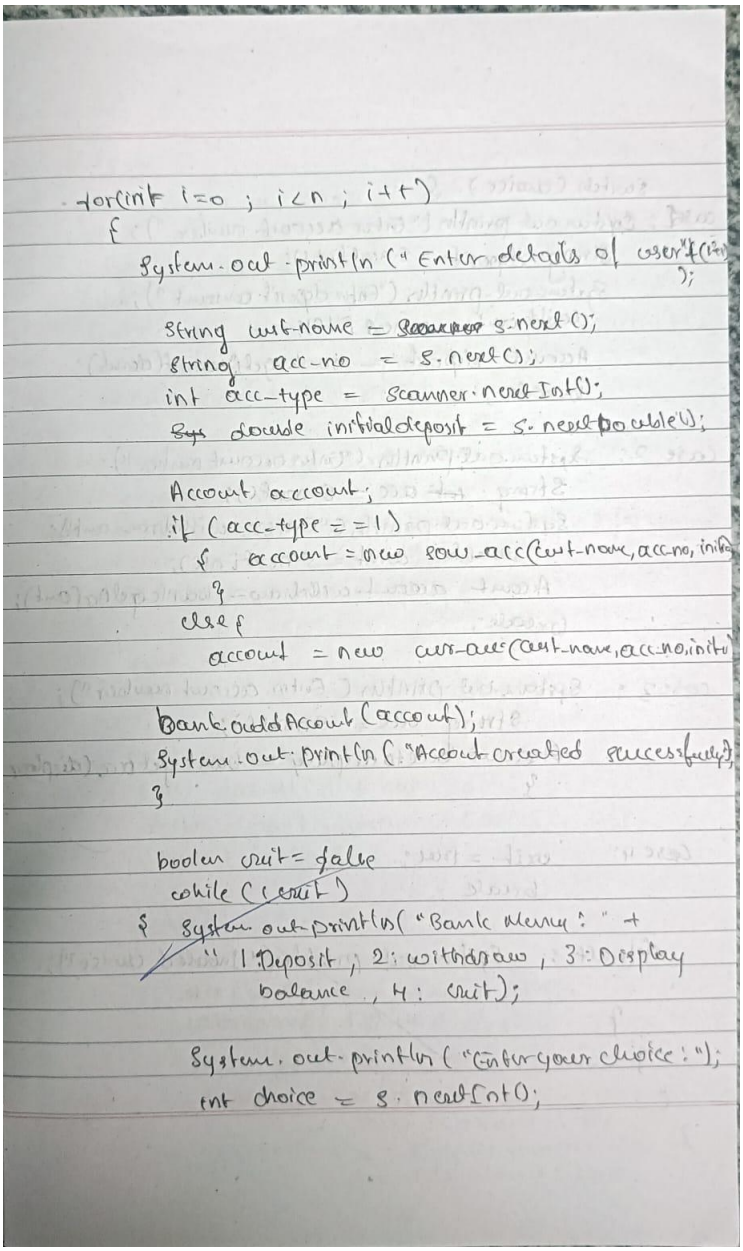
```
import java.util.*;
abstract class Account
{   String cust_name;
    string acc_no;
    double bal;

  Account(String n, string a, double bal)
   {   cust_name = n;
       acc_no = a;
       bal = bal;
   }

void deposit(double amt){
    if(amt>0){
        bal += amt;
        System.out.println("Deposited "+amt);
    }
}
```



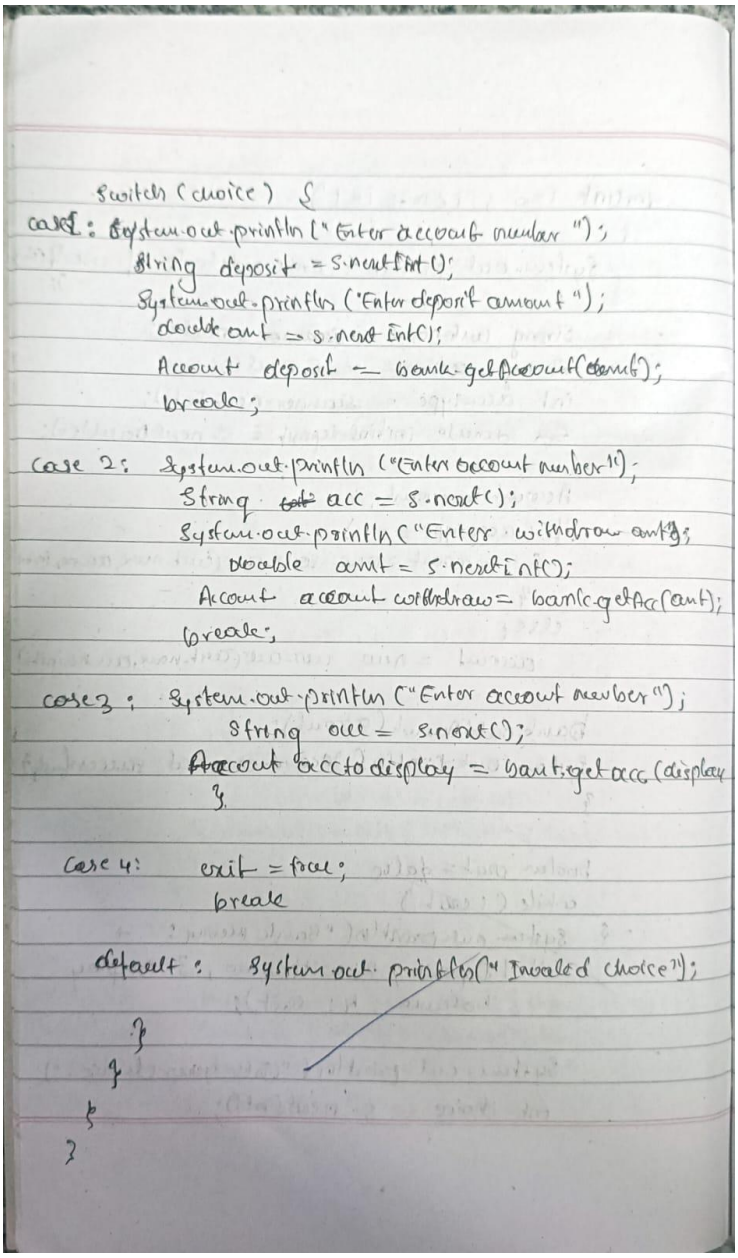
```
void getbalance()
{ System.out.println("Account Balance "+bal);
}
abstract void withdraw(double amt);
}

class curr_acc extends Account
{ double min_bal = 1000;
  double penalty = 50;

curr_acc(String n, String a, double bal){
{
  super(n, a, bal);
}

void withdraw(double amt)
{ if(amt<=bal)
  { bal-=amt;
  System.out.println("Withdrawn:"+amt);}
    if(bal<min_bal)
    { bal-= penalty;
    }
  }
}
```



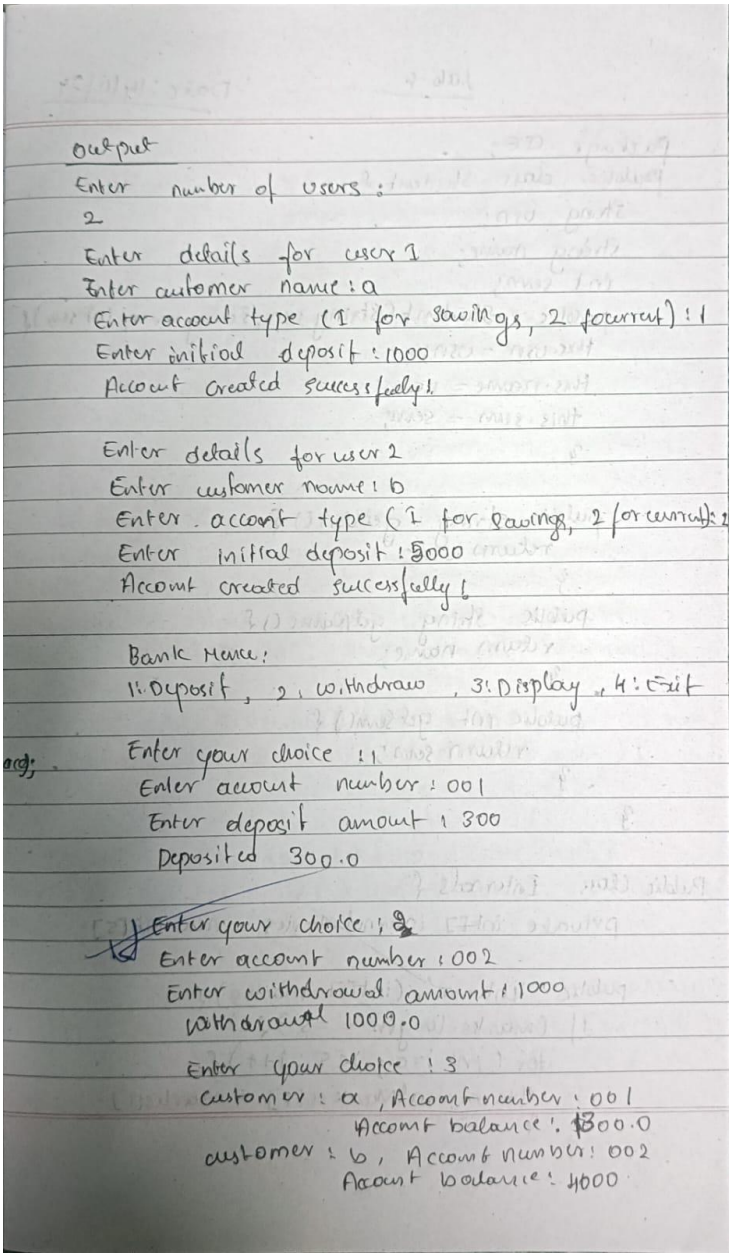
```
class sav_acc extends Accounts {
    double interest = 0.05;

    public sav_acc(String n, String a, double bal)
        { super(n,a,bal); }

    void withdraw(double amt)
    {   if (amt < bal)
        { bal -= amt;
    System.out.println ("withdrawn:"+ amt);
        }
        else {
        System.out.println("Insufficient Balance");
        }
    }

    void displayBalance()
    {
        double interest = bal*interest
            balance += interest;
    System.out.println("Interest of "+interest+"deposited");
    }
}

class Bank
{   public static void main (String args[])
    {   Scanner s = new Scanner(System.in);
        System.out.println("Enter Number of users");
        int n = next.Int();
```



```
for(int i=0 ; i<n ; i++)
{
System.out.println("Enter details of user"+(i+1));

String cust_name = s.next();
String acc_no = s.next();
int acc_type = scanner.nextInt();
double initialdeposit = s.nextDouble();

Account account;
if (acc_type == 1)
{ account = new sav_acc(cust_name, acc_no, init
}
else{
account = new cur_acc(cust_name, acc_no, init
Bank.addAccount(account);
System.out.println("Account created succesfully");
}

boolean exit = false
while (!exit)
{ System.out.println("Bank Menu:" +
"1:Deposit, 2: withdraw, 3: Display
balance, 4: exit);

System.out.println("Enter your choice:");
int choice = s.nextInt();
```



```
switch (choice) {
case 1: System.out.println("Enter account number");
        String deposit = s.nextInt();
        System.out.println("Enter deposit amount");
        double amt = s.nextInt();
        Account deposit = bank.getAccount(amt);
        break;

case 2: System.out.println("Enter account number");
        String acc = s.next();
        System.out.println("Enter withdraw amt");
        double amt = s.nextInt();
        Account account withdraw = bank.getAcc(amt);
        break;

case 3: System.out.println("Enter account number");
        String acc = s.next();
        Account acctodisplay = bank.getacc(display
        }

case 4: exit = true;
        break

default: System.out.println("Invalid choice");

    }
  }
}
```


Output

Enter number of users :
2
Enter details for user 1
Enter customer name : a
Enter account type (1 for savings, 2 for current) : 1
Enter initial deposit : 1000
Account created successfully!

Enter details for user 2
Enter customer name : b
Enter account type (1 for savings, 2 for current) : 2
Enter initial deposit : 5000
Account created successfully!

Bank Menu :
1 : Deposit, 2 : Withdraw, 3 : Display, 4 : Exit

Enter your choice : 1
Enter account number : 001
Enter deposit amount : 300
Deposited 300.0

Enter your choice : 2
Enter account number : 002
Enter withdrawal amount : 1000
Withdrawal 1000.0

Enter your choice : 3
Customer : a, Account number : 001
Account balance : 1300.0
customer : b, Account number : 002
Account balance : 4000

```
import java.util.*;


abstract class Account {
    protected String customerName;
    protected String accountNumber;
    protected double balance;
```

```
    public Account(String customerName, String accountNumber, double initialBalance) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.balance = initialBalance;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: " + amount);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }

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

    public abstract void withdraw(double amount);
}

class CurrentAccount extends Account {
    private static final double MIN_BALANCE = 1000;
    private static final double SERVICE_CHARGE = 50;
```

```
    public CurrentAccount(String customerName, String accountNumber, double initialBalance) {
        super(customerName, accountNumber, initialBalance);
    }

    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
            if (balance < MIN_BALANCE) {
                balance -= SERVICE_CHARGE;
                System.out.println("Balance below minimum. Service charge of " + SERVICE_CHARGE +
" imposed.");
            }
        } else {
            System.out.println("Insufficient funds!");
        }
    }
}

class SavingsAccount extends Account {
    private static final double INTEREST_RATE = 0.05;

    public SavingsAccount(String customerName, String accountNumber, double initialBalance) {
        super(customerName, accountNumber, initialBalance);
    }
```

```
    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient funds!");
        }
    }

    public void computeAndDepositInterest() {
        double interest = balance * INTEREST_RATE;
        balance += interest;
        System.out.println("Interest of " + interest + " deposited.");
    }
}

class Bank {
    private List<Account> accounts = new ArrayList<>();

    public void addAccount(Account account) {
        accounts.add(account);
    }

    public void displayAllAccounts() {
        for (Account account : accounts) {
```

```
            System.out.println("Customer:  " + account.customerName + ", Account Number: " +
account.accountNumber);
            account.displayBalance();
        }
    }

    public Account getAccount(String accountNumber) {
        for (Account account : accounts) {
            if (account.accountNumber.equals(accountNumber)) {
                return account;
            }
        }
        return null;
    }
}

public class BankApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Bank bank = new Bank();
        int n;

        System.out.println("Enter  the number of users:");
        n = scanner.nextInt();
        scanner.nextLine();
```

```
for (int i = 0; i < n; i++) {
    System.out.println("Enter  details for User " + (i + 1));

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

    System.out.print("Enter  account number: ");
    String accountNumber = scanner.nextLine();

    System.out.print("Enter  account type (1 for Savings, 2 for Current): ");
    int accountType = scanner.nextInt();
    scanner.nextLine();

    System.out.print("Enter  initial deposit: ");
    double initialDeposit = scanner.nextDouble();
    scanner.nextLine();

    Account account;
    if (accountType == 1) {
        account = new SavingsAccount(customerName, accountNumber, initialDeposit);
    } else {
        account = new CurrentAccount(customerName, accountNumber, initialDeposit);
    }

    bank.addAccount(account);
    System.out.println("Account  created successfully!\n");
```

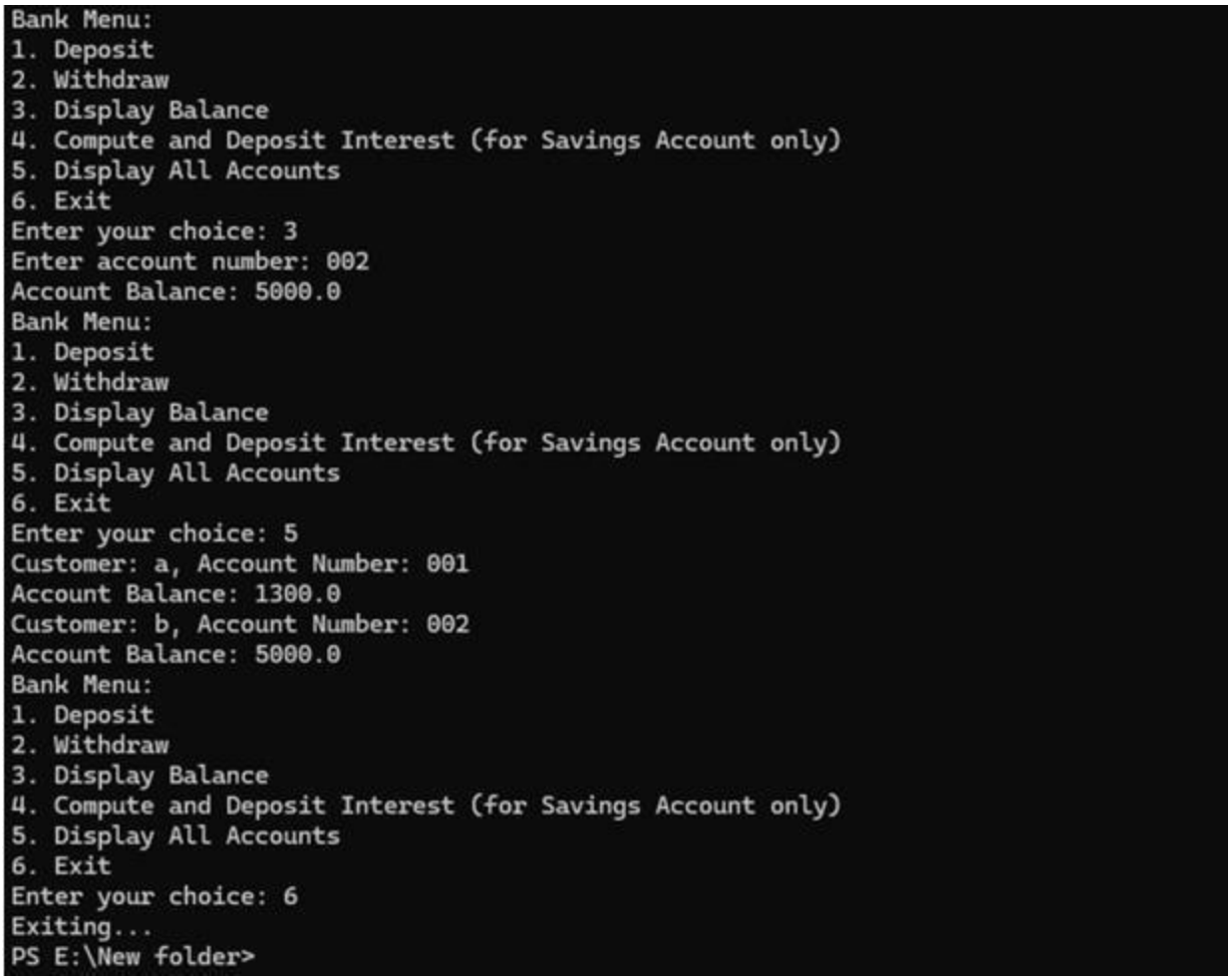
```
}

boolean exit = false;
while (!exit) {
   System.out.println("Bank  Menu:");
   System.out.println("1.  Deposit");
   System.out.println("2.  Withdraw");
   System.out.println("3.  Display Balance");
   System.out.println("4.  Compute and Deposit Interest (for Savings Account only)");
   System.out.println("5.  Display All Accounts");
   System.out.println("6.  Exit");

   System.out.print("Enter  your choice: ");
   int choice = scanner.nextInt();
   scanner.nextLine();

   switch (choice) {
      case 1:
         System.out.print("Enter  account number: ");
         String depositAccount = scanner.nextLine();
         System.out.print("Enter  deposit amount: ");
         double depositAmount = scanner.nextDouble();
         scanner.nextLine();
         Account accountToDeposit = bank.getAccount(depositAccount);
         if (accountToDeposit != null) {
            accountToDeposit.deposit(depositAmount);
```

```
    } else {
        System.out.println("Account  not found.");
    }
    break;

case 2:
    System.out.print("Enter  account number: ");
    String withdrawAccount = scanner.nextLine();
    System.out.print("Enter  withdrawal amount: ");
    double withdrawAmount = scanner.nextDouble();
    scanner.nextLine();
    Account accountToWithdraw = bank.getAccount(withdrawAccount);
    if (accountToWithdraw != null) {
        accountToWithdraw.withdraw(withdrawAmount);
    } else {
        System.out.println("Account  not found.");
    }
    break;

case 3:
    System.out.print("Enter  account number: ");
    String displayAccount = scanner.nextLine();
    Account accountToDisplay = bank.getAccount(displayAccount);
    if (accountToDisplay != null) {
        accountToDisplay.displayBalance();
    } else {
```

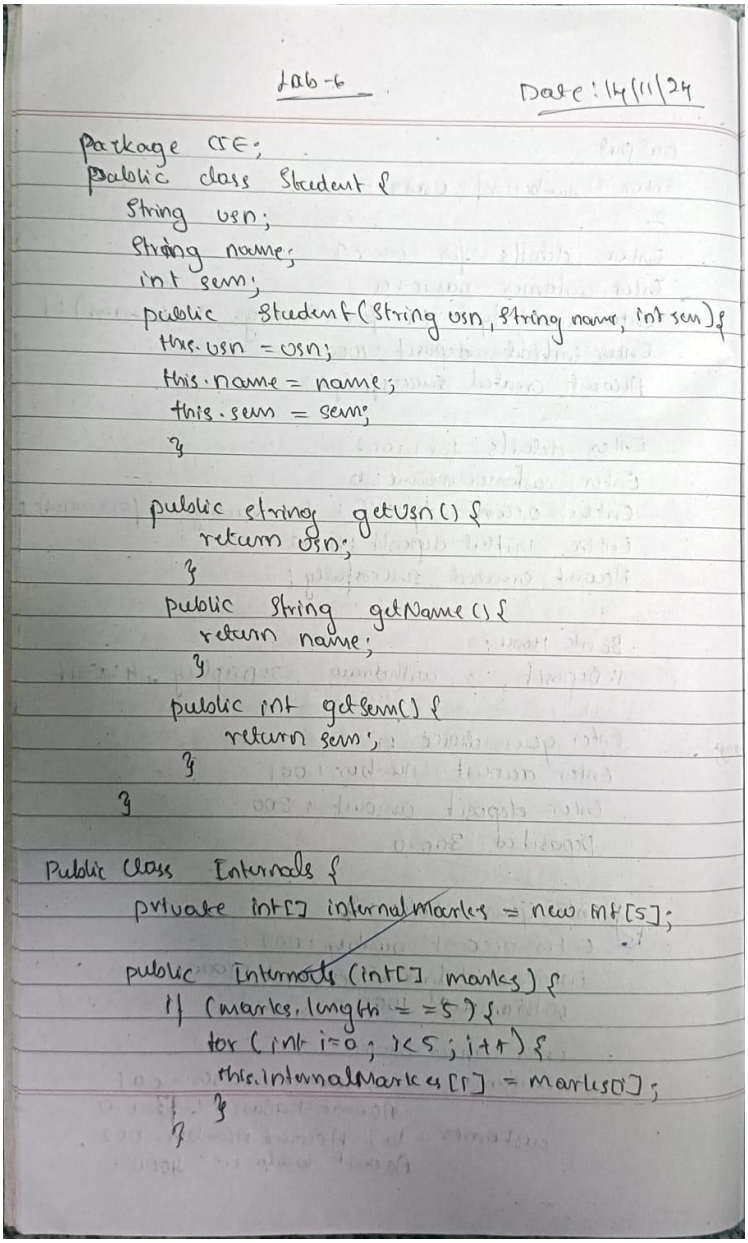
```
            System.out.println("Account  not found.");
        }
        break;

    case 4:
        System.out.print("Enter  account number: ");
        String interestAccount = scanner.nextLine();
        Account accountForInterest = bank.getAccount(interestAccount);
        if (accountForInterest != null && accountForInterest instanceof SavingsAccount) {
            ((SavingsAccount) accountForInterest).computeAndDepositInterest();
        } else {
            System.out.println("Invalid  account or not a Savings Account.");
        }
        break;

    case 5:
        bank.displayAllAccounts();
        break;

    case 6:
        exit = true;
        System.out.println("Exiting...");
        break;

    default:
        System.out.println("Invalid  choice! Please try again.");
```

```
        }
    }
  }
}
```

```
PS E:\New folder> java BankApp;
Enter the number of users:
2
Enter details for User 1
Enter customer name: a
Enter account number: 001
Enter account type (1 for Savings, 2 for Current): 1
Enter initial deposit: 1000
Account created successfully!

Enter details for User 2
Enter customer name: b
Enter account number: 002
Enter account type (1 for Savings, 2 for Current): 2
Enter initial deposit: 5000
Account created successfully!

Bank Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute and Deposit Interest (for Savings Account only)
5. Display All Accounts
6. Exit
Enter your choice: 1
Enter account number: 001
Enter deposit amount: 300
Deposited: 300.0
Bank Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute and Deposit Interest (for Savings Account only)
5. Display All Accounts
6. Exit
Enter your choice: 2
Enter account number: b
Enter withdrawal amount: 100
Account not found.
```

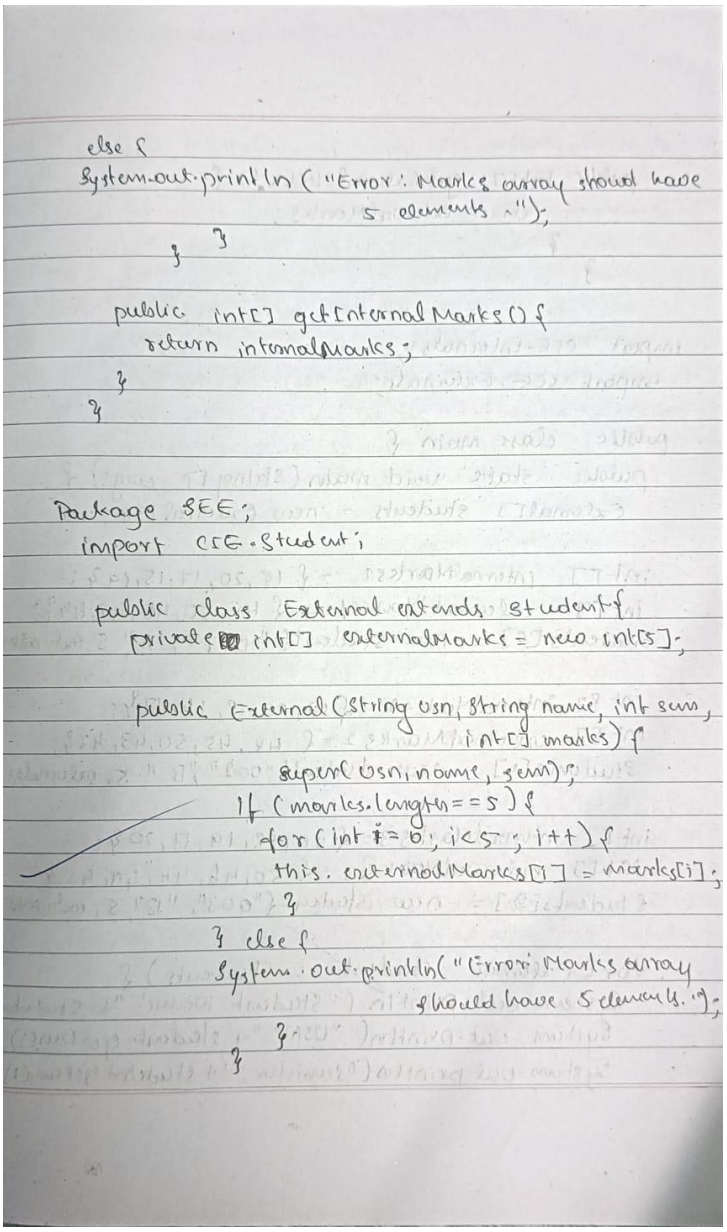


6. Create a package CIE which has two classes- Student and Internals. The class Personal has members like usn, name, sem. The class internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

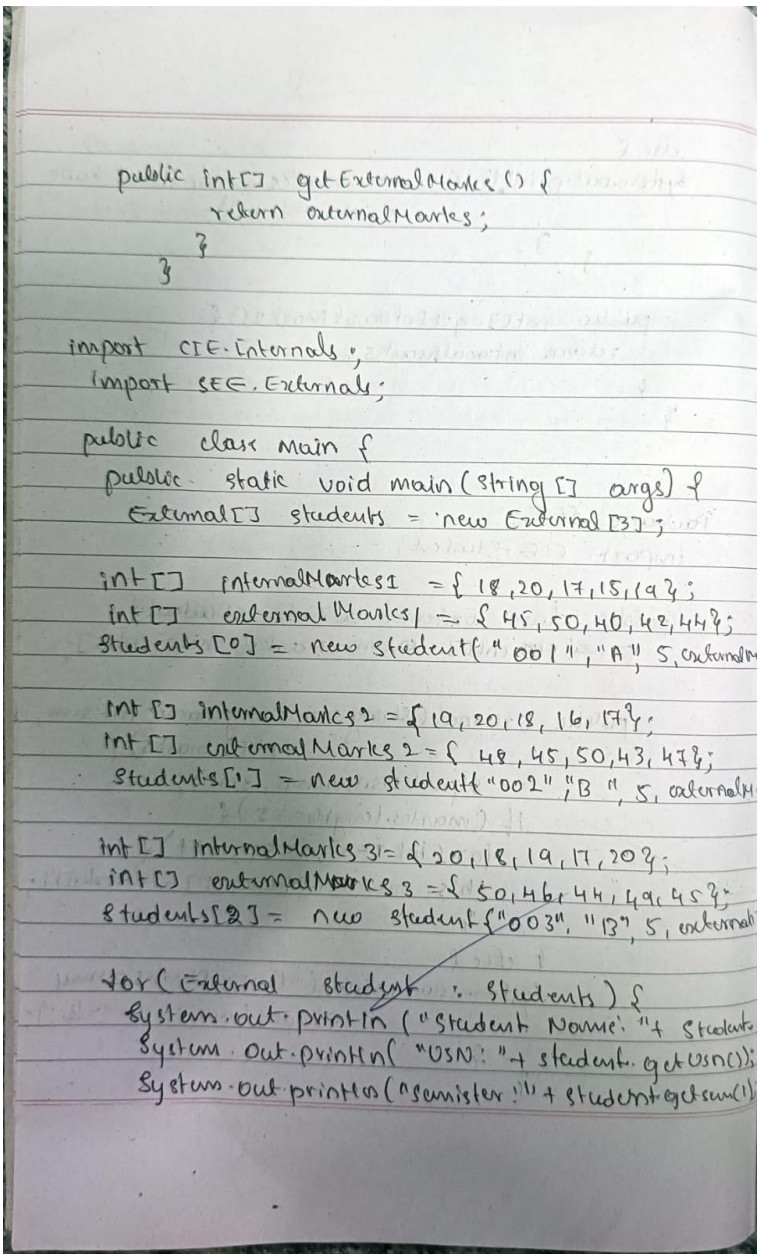


Lab-6 Date: 14/11/24

```
package CIE;
public class Student {
    String usn;
    String name;
    int sem;
    public Student(String usn, String name, int sem){
      this.usn = usn;
      this.name = name;
      this.sem = sem;
    }

    public string getUsn() {
        return usn;
    }
    public String getName() {
        return name;
    }
    public int getSem() {
        return sem;
    }
}

Public class Internals {
    private int[] internalMarks = new int[5];

    public Internals(int[] marks) {
      if (marks.length == 5) {
        for (int i=0; i<5; i++) {
        this.internalMarks[i] = marks[i];
        }
      }
```



```
else {
System.out.println("Error: Marks array should have
                    5 elements.");
    }  }

  public int[] getExternalMarks() {
    return internalMarks;
  }
}

Package SEE;
import CIE.Student;

public class External extends Student {
  private int[] externalMarks = new int[5];

  public External(String usn, String name, int sem,
                  int[] marks) {
    super(usn, name, sem);
    if (marks.length == 5) {
      for (int i = 0; i < 5; i++) {
        this.externalMarks[i] = marks[i];
      }
    } else {
      System.out.println("Error: Marks array
        should have 5 elements.");
    }
  }
```

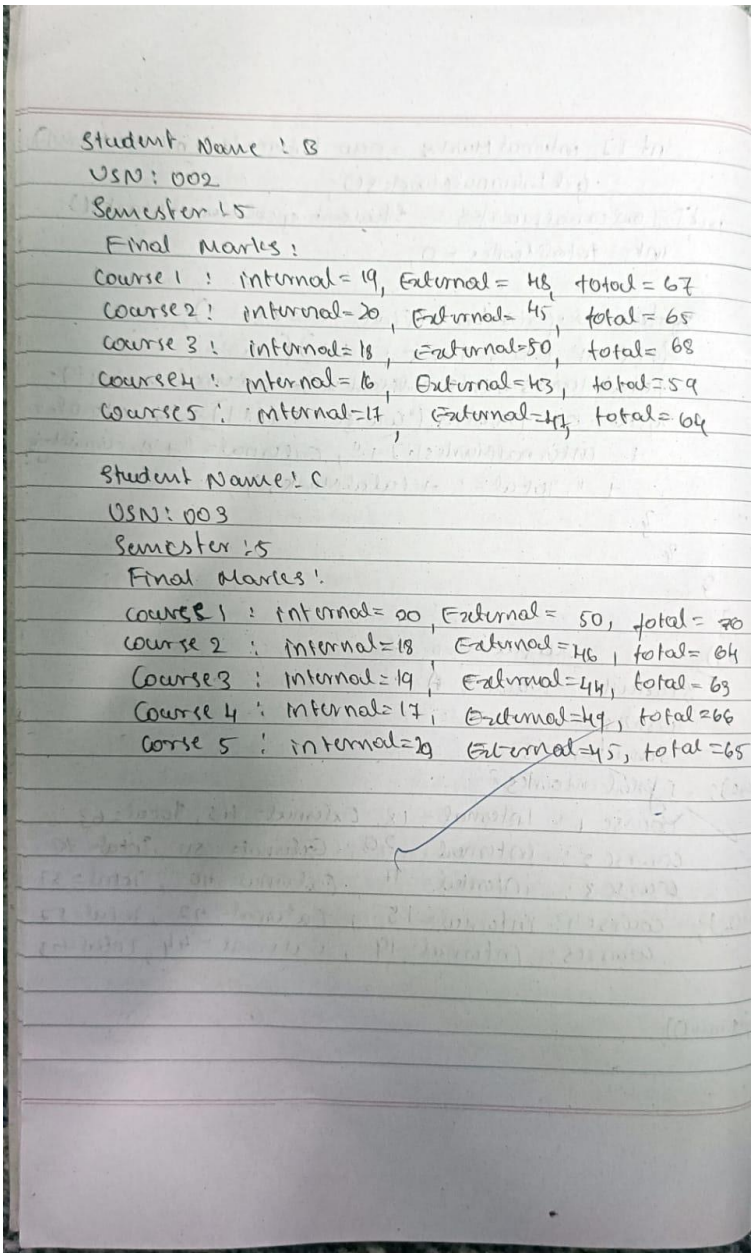


```
    public int[] getExternalMarks() {
        return externalMarks;
    }
}

import CIE.Internals;
import SEE.Externals;

public class Main {
    public static void main(String[] args) {
        External[] students = new External[3];

        int[] internalMarks1 = {18, 20, 17, 15, 19};
        int[] externalMarks1 = {45, 50, 40, 42, 44};
        students[0] = new student("001", "A", 5, externalM

        int[] internalMarks2 = {19, 20, 18, 16, 17};
        int[] externalMarks2 = {48, 45, 50, 43, 47};
        students[1] = new student("002", "B", 5, externalM

        int[] internalMarks3 = {20, 18, 19, 17, 20};
        int[] externalMarks3 = {50, 46, 44, 49, 45};
        students[2] = new student("003", "13", 5, externali

        for(External student : students) {
            System.out.println("Student Name: " + student
            System.out.println("USN: " + student.getUSN());
            System.out.println("Semister: " + student.getSem());
```

```
int [] internalMarks = new Internals (internalMarks);
        getInternalMarks();
int[] externalMarks = Student.getExternalMarks();
    int totalMarks = 0;

System.out.println ("Final Marks: ");
for( int i=0 ; i< 5 ; i++){
totalMarks = internalMarks[i]+ externalMarks[i];
System.out.println ("Course " + (i+1) + ": interal=" +
  + internalMarks[i] +", External= " + externalMarks[i]
    + ", Total= " + totalMarks );
  }
 }
}
}
```

Output

```
Student Name : A
USN : 001
Semister : 5
Final Marks:
Course 1 : Internal = 18, External = 45, Total = 63
Course 2 : Internal = 20, External = 50, Total = 70
Course 3 : Internal = 17, External = 40, Total = 57
Course 4 : Internal = 15, External = 42, Total = 57
Course 5 : Internal = 19, External = 44, Total = 63
```



Student Name : B
USN : 002
Semester : 5
Final Marks :
Course 1 : internal = 19, External = 48, total = 67
Course 2 : internal = 20, External = 45, total = 65
Course 3 : internal = 18, External = 50, total = 68
Course 4 : internal = 16, External = 43, total = 59
Course 5 : internal = 17, External = 47, total = 64

Student Name : C
USN : 003
Semester : 5
Final Marks :
Course 1 : internal = 20, External = 50, total = 70
Course 2 : internal = 18, External = 46, total = 64
Course 3 : internal = 19, External = 44, total = 63
Course 4 : internal = 17, External = 49, total = 66
Course 5 : internal = 20, External = 45, total = 65

```
public class Student {

    protected String usn;
    protected String name;
    protected String sem;

    public Student(String usn, String name, String sem) {
        this.usn = usn;
        this.name = name;
        this.sem = sem;
    }

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

public class Internals extends Student {

    protected int[] internalMarks = new int[5];

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

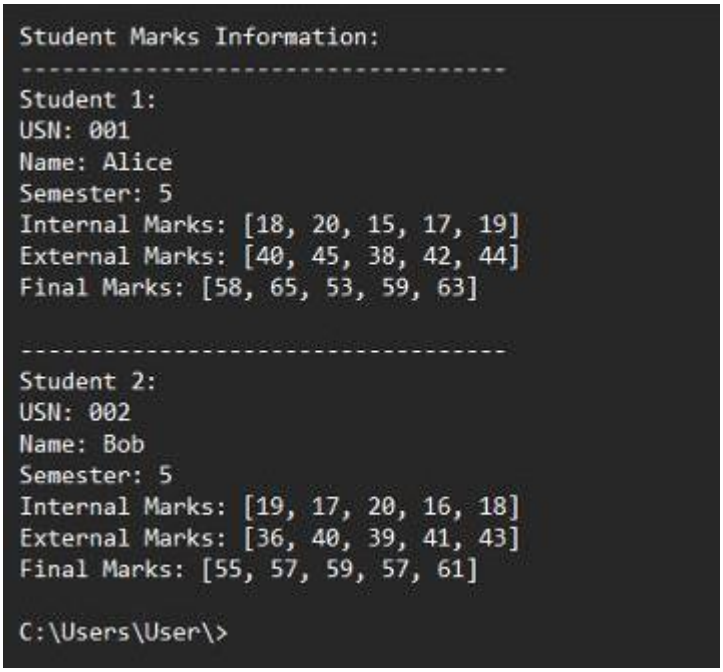
```
    }

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

package SEE;

import CIE.Internals;

public class External extends Internals {
    int[] externalMarks = new int[5];

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

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

```
        }
    }

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

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

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

        System.out.print("Enter  number of students: ");
        int n = scanner.nextInt();
        scanner.nextLine();

        External[] students = new External[n];
```

```
for (int i = 0; i < n; i++) {
    System.out.println("\nEnter  details for student " + (i + 1));

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

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

    System.out.print("Enter  Semester: ");
    String sem = scanner.nextLine();

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

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

```
            students[i] = new External(usn, name, sem, internalMarks, externalMarks);
        }

        System.out.println("\nStudent  Marks Information:");
        for (int i = 0; i < n; i++) {
            students[i].displayDetails();
            students[i].displayInternalMarks();
            students[i].displayExternalMarks();
            students[i].displayFinalMarks();
            System.out.println();
        }
        scanner.close();
    }
}
```

```
Enter number of students: 2

Enter details for student 1
Enter USN: 001
Enter Name: Alice
Enter Semester: 5
Enter internal marks for 5 courses:
Course 1: 18
Course 2: 20
Course 3: 15
Course 4: 17
Course 5: 19
Enter external marks for 5 courses:
Course 1: 40
Course 2: 45
Course 3: 38
Course 4: 42
Course 5: 44

Enter details for student 2
Enter USN: 002
Enter Name: Bob
Enter Semester: 5
Enter internal marks for 5 courses:
Course 1: 19
Course 2: 17
Course 3: 20
Course 4: 16
Course 5: 18
Enter external marks for 5 courses:
Course 1: 36
Course 2: 40
Course 3: 39
Course 4: 41
Course 5: 43
```

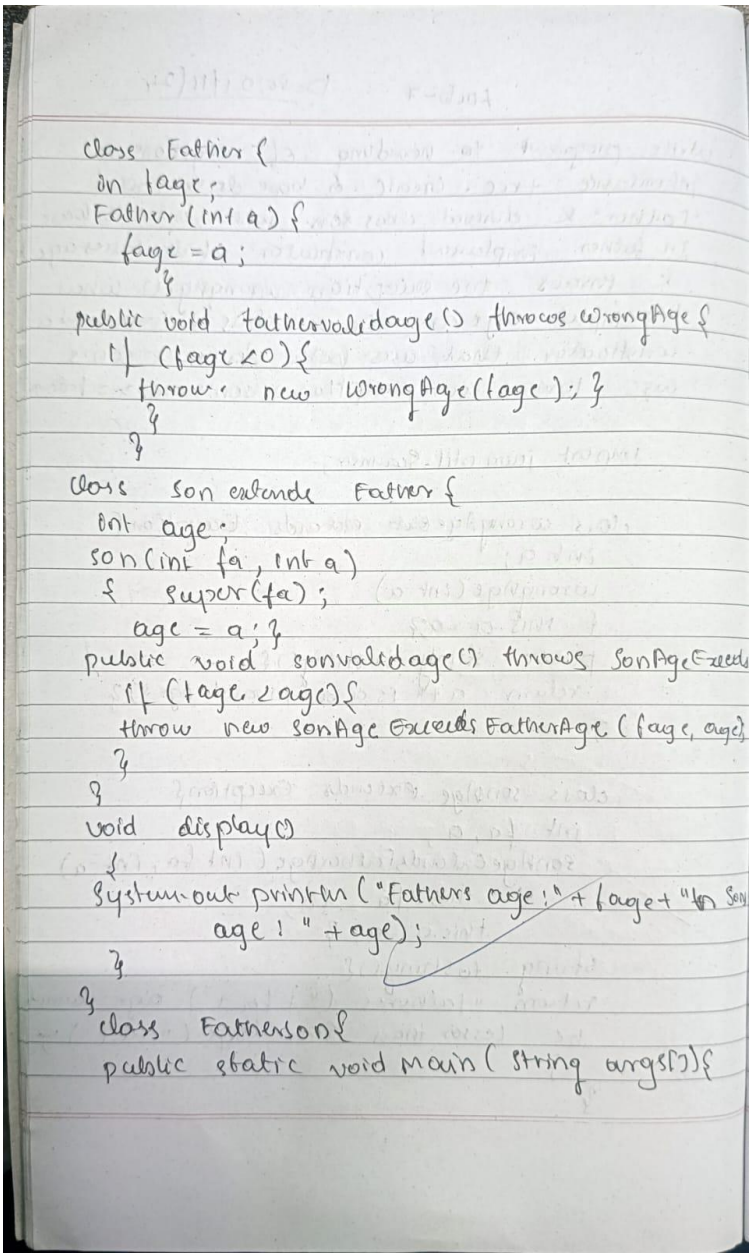


Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son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

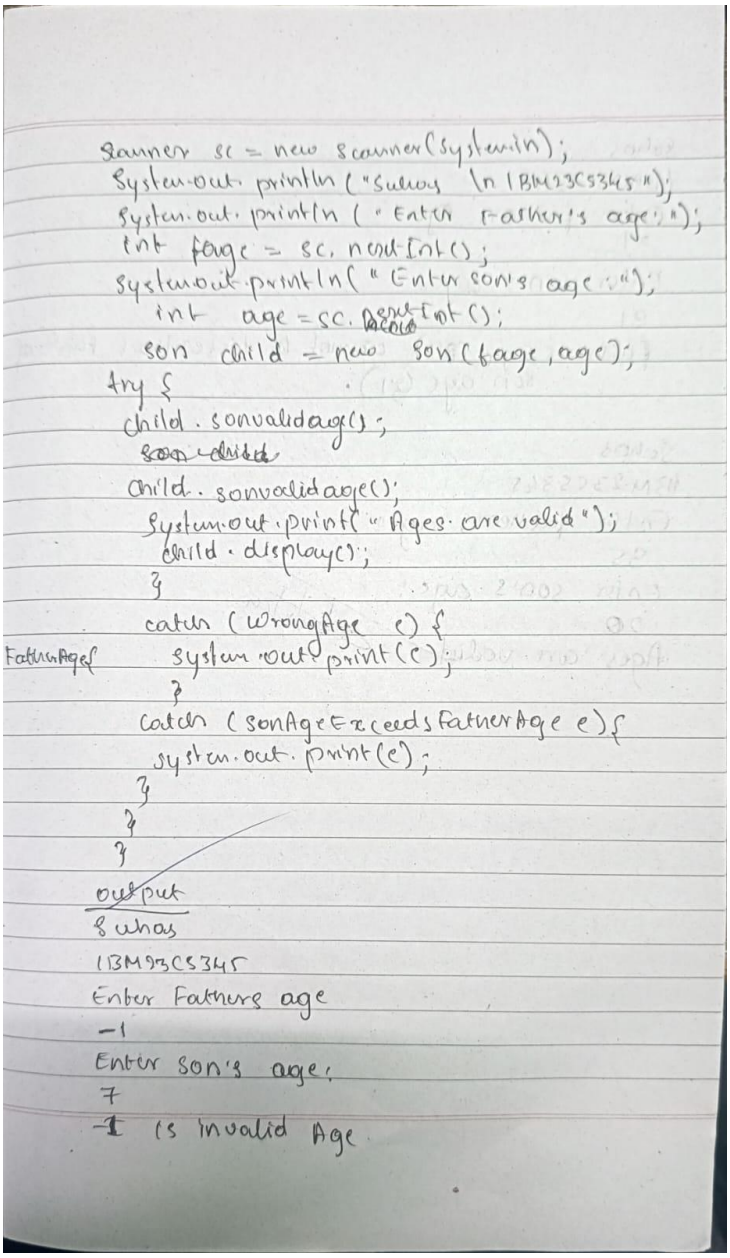
Lab-7 Date: 21/11/24

Write program to handling of exceptions in inheritance tree. Create a base class called "Father" & derived class son which extends base. In father implement constructor which takes age & throws the exception wrongAge() when the input age<0. In son class, implement a constructor that uses both father & son's age throws an exception son's age >= father

```
import java.util.Scanner;

class wrongAge extends Exception {
  int a;
  wrongAge(int a)
  { this.a = a;}
  public string toString(){
   return a+" is a invalid Age";}
}

class sonAge extends Exception{
  int fa, a;
  sonAge ExtendsFatherAge (int fa, int a)
     { this.fa = fa;
       this.a = a; }
  String toString(){
  return "father's ("+fa+") age cannot
  be lesser than son age ("+a+")";
  }
}
```



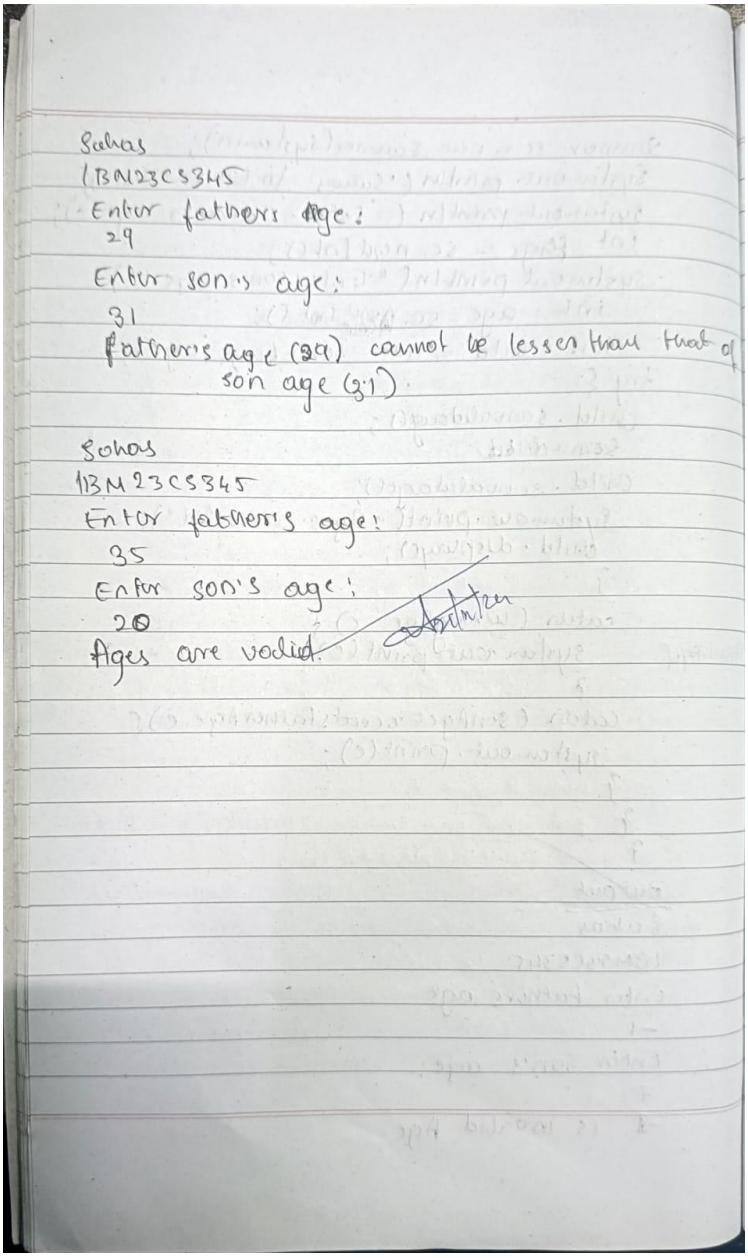
```
class Father {
 in fage;
 Father (int a) {
   fage = a;
    }
public void fathervalidage() throws WrongAge {
  if (fage<0){
    throw new WrongAge(fage); }
   }
  }
class son extends Father {
 int age;
 son(int fa, int a)
  { super(fa);
    age = a; }
public void sonvalidage() throws SonAgeExceeds
   if (fage<age){
   throw new SonAgeExceedsFatherAge (fage, age);
  }
}
void display()
 {
System.out.println("Fathers age:" + fage + "\n Son
        age : " + age);
  }
}
class FatherSon{
public static void main( String args[]){
```



```
Scanner sc = new Scanner(System.in);
System.out.println("Sulaiman \n 1BM23CS345");
System.out.println("Enter Father's age:");
int fage = sc.nextInt();
System.out.println("Enter son's age:");
  int age = sc.nextInt();
  Son child = new Son(fage, age);
try {
child.sonvalidage();
child.sonvalidage();
 System.out.print("Ages are valid");
  child.display();
 }
 catch (WrongAge e) {
FatherAge    System.out.print(e);
  }
 catch (SonAgeExceedsFatherAge e){
  System.out.print(e);
 }
}
}
```

Output

```
Sulaiman
1BM23CS345
Enter Fathers age
-1
Enter son's age:
7
-1 is invalid Age
```


Sahas
1BM23CS345
Enter father's age:
29
Enter son's age:
31
Father's age (29) cannot be lesser than that of son age (31).

Sahas
1BM23CS345
Enter father's age:
35
Enter son's age:
20
Ages are vodid.

```
import java.util.Scanner;

class WrongAge extends Exception {
    int a;

    WrongAge(int a) {
```

```
        this.a = a;
    }

    public String toString() {
        return a + " is an invalid Age";
    }
}

class SonAgeExceedsFatherAge extends Exception {
    int fa, a;

    SonAgeExceedsFatherAge(int fa, int a) {
        this.fa = fa;
        this.a = a;
    }

    public String toString() {
        return "Father's (" + fa + ") age cannot be lesser than that of son (" + a + ")";
    }
}

class Father {
    int fage;

    Father(int a) {
        fage = a;
```

```
    }

    public void fathervalidage() throws WrongAge {
        if (fage < 0) {
            throw new WrongAge(fage);
        }
    }
}

class Son extends Father {
    int age;

    Son(int fa, int a) {
        super(fa);
        age = a;
    }

    public void sonvalidage() throws SonAgeExceedsFatherAge {
        if (fage < age) {
            throw new SonAgeExceedsFatherAge(fage, age);
        }
    }

    void display() {
        System.out.println("Father's age: " + fage + "\nSon's age: " + age);
    }
```

```
}

class FatherSon {
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Suhas  \n1BM23CS345");
      System.out.println("Enter  Father's age:");
      int fage = sc.nextInt();
      System.out.println("Enter  Son's age:");
      int age = sc.nextInt();

      Son child = new Son(fage, age);

      try {
         child.fathervalidage();
         child.sonvalidage();
         System.out.println("Ages  are valid");
         child.display();
      } catch (WrongAge e) {
         System.out.println(e);
      } catch (SonAgeExceedsFatherAge e) {
         System.out.println(e);
      }
   }
}
```

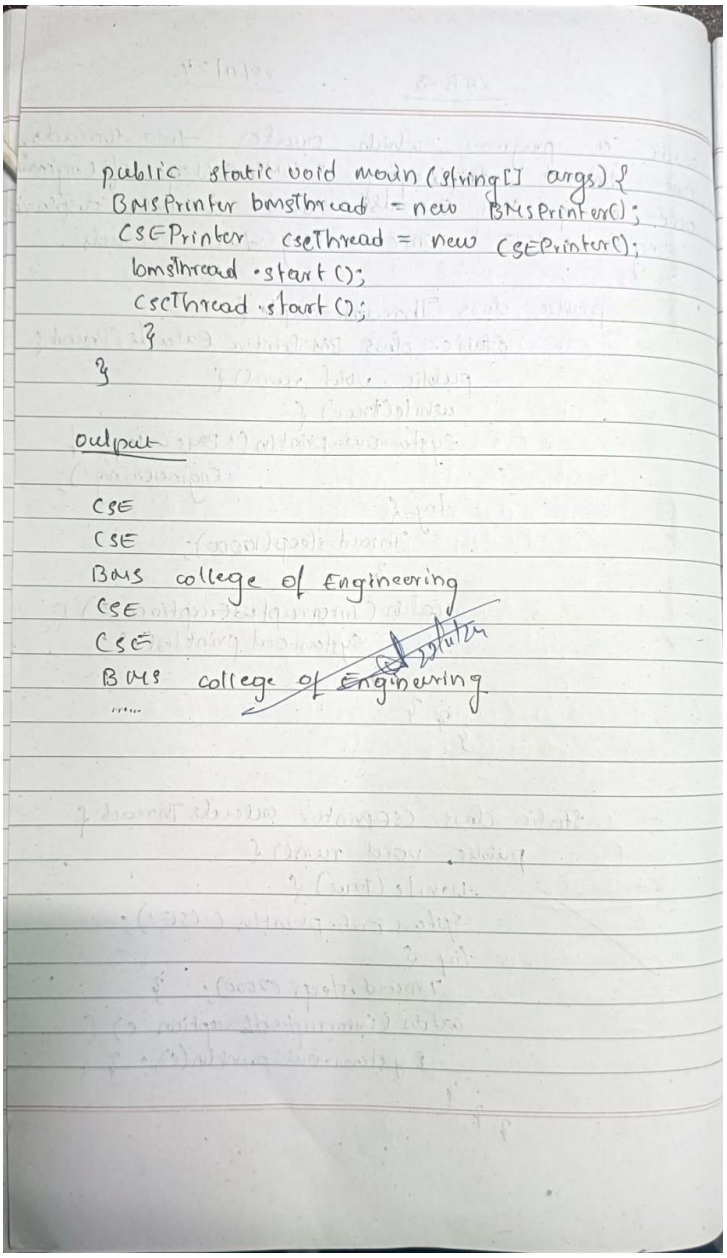
8. Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.

LAB-8 28/11/24

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

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

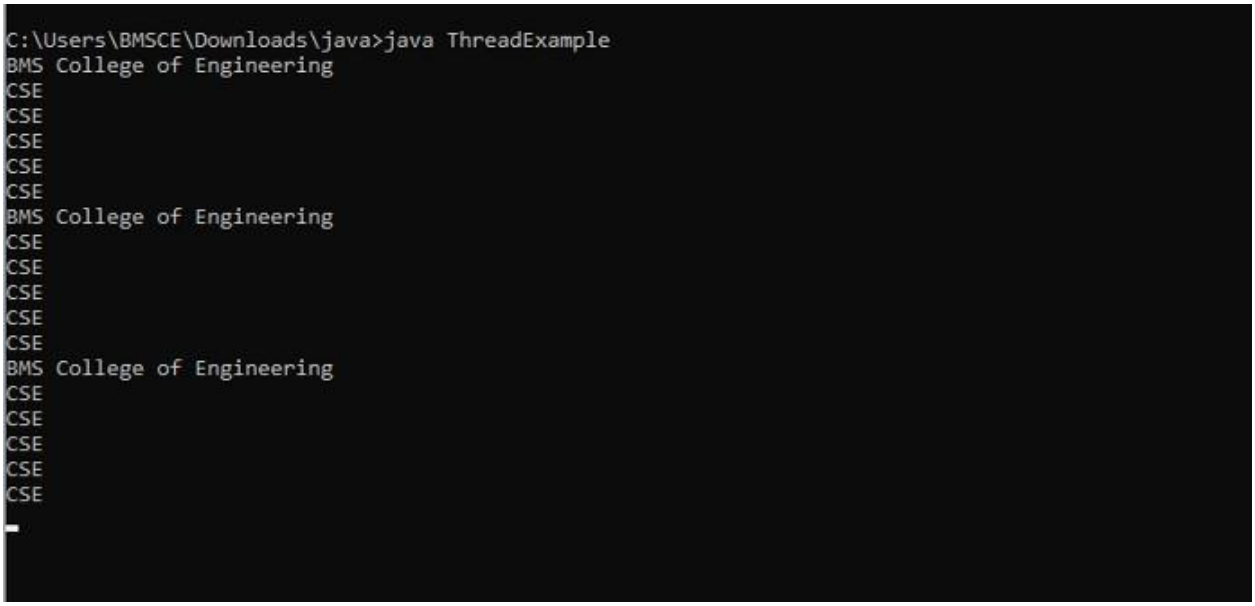
```
    static class CSEPrinter extends Thread {
        public void run() {
            while (true) {
            System.out.println ("CSE");
            try {
                Thread.sleep (2000); }
            catch (InterruptedException e) {
                System.out.println(e); }
            }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



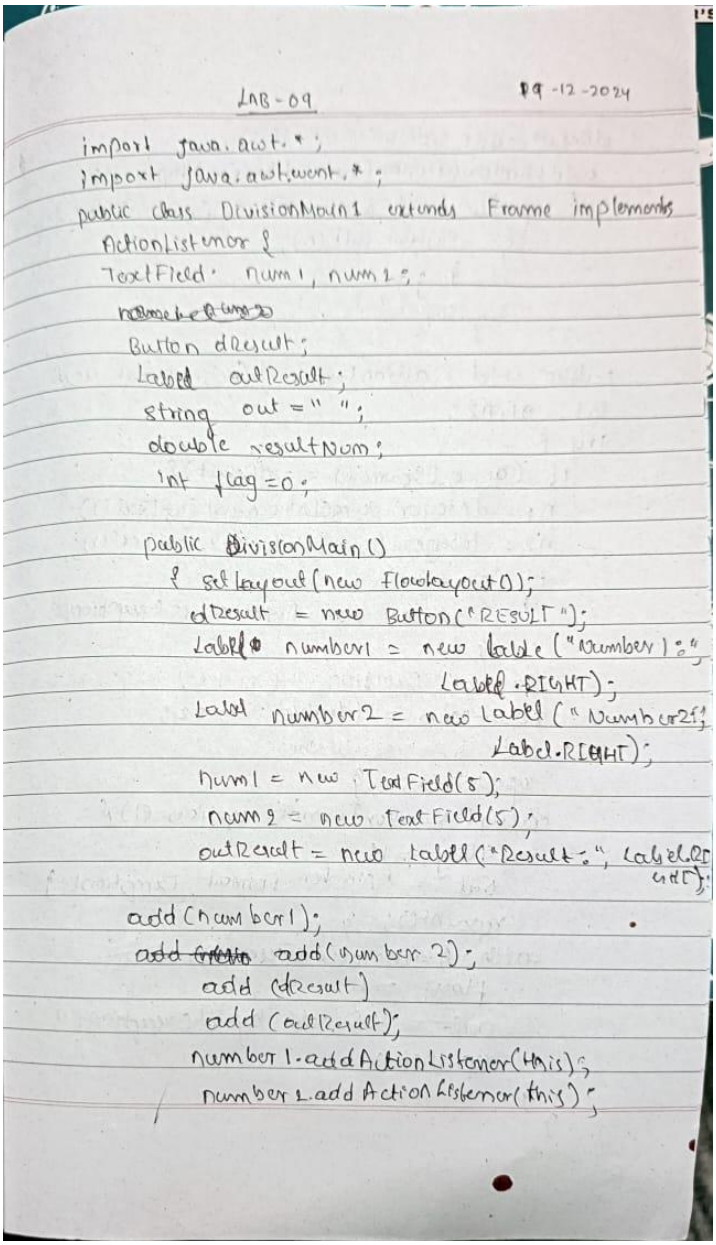

```
public class ThreadExample {


   static class BMSPrinter extends Thread {
      public void run() {
         while (true) {
            System.out.println("BMS College of Engineering");
            try {
```

```
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
  }
}

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

public static void main(String[] args) {
  BMSPrinter bmsThread = new BMSPrinter();
  CSEPrinter cseThread = new CSEPrinter();

  bmsThread.start();
```

```
        cseThread.start();
    }
}
```

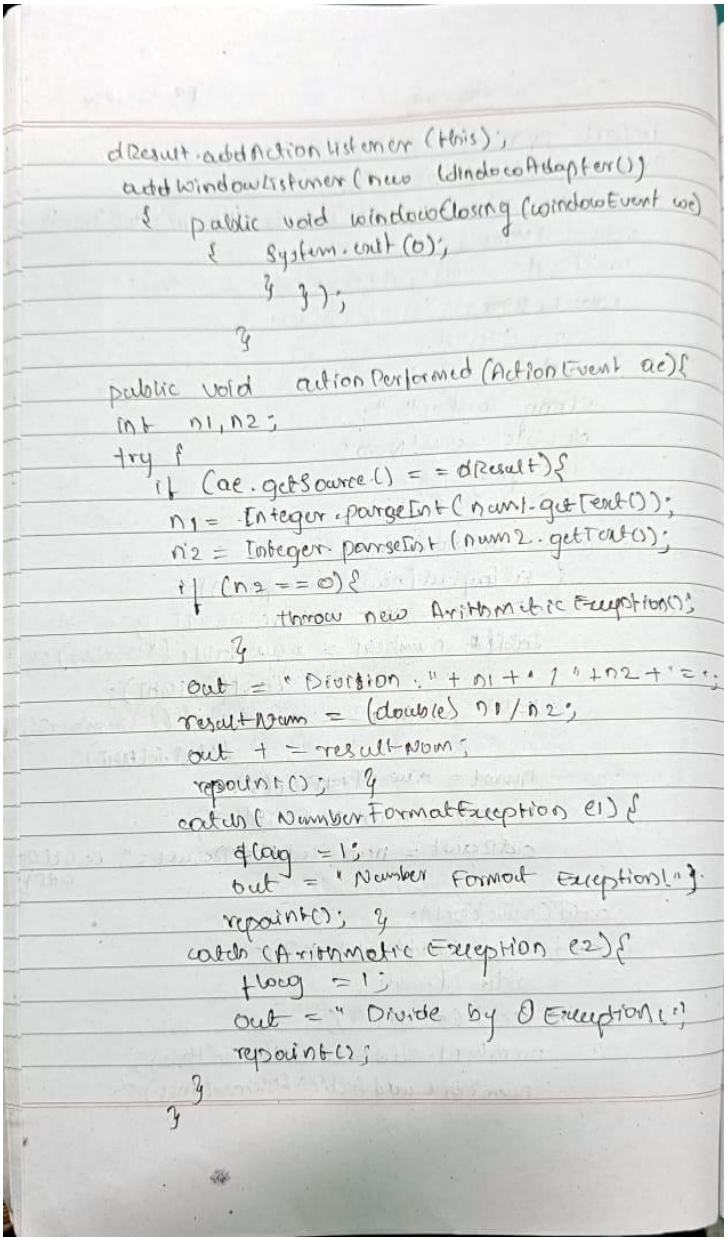


9.Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a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

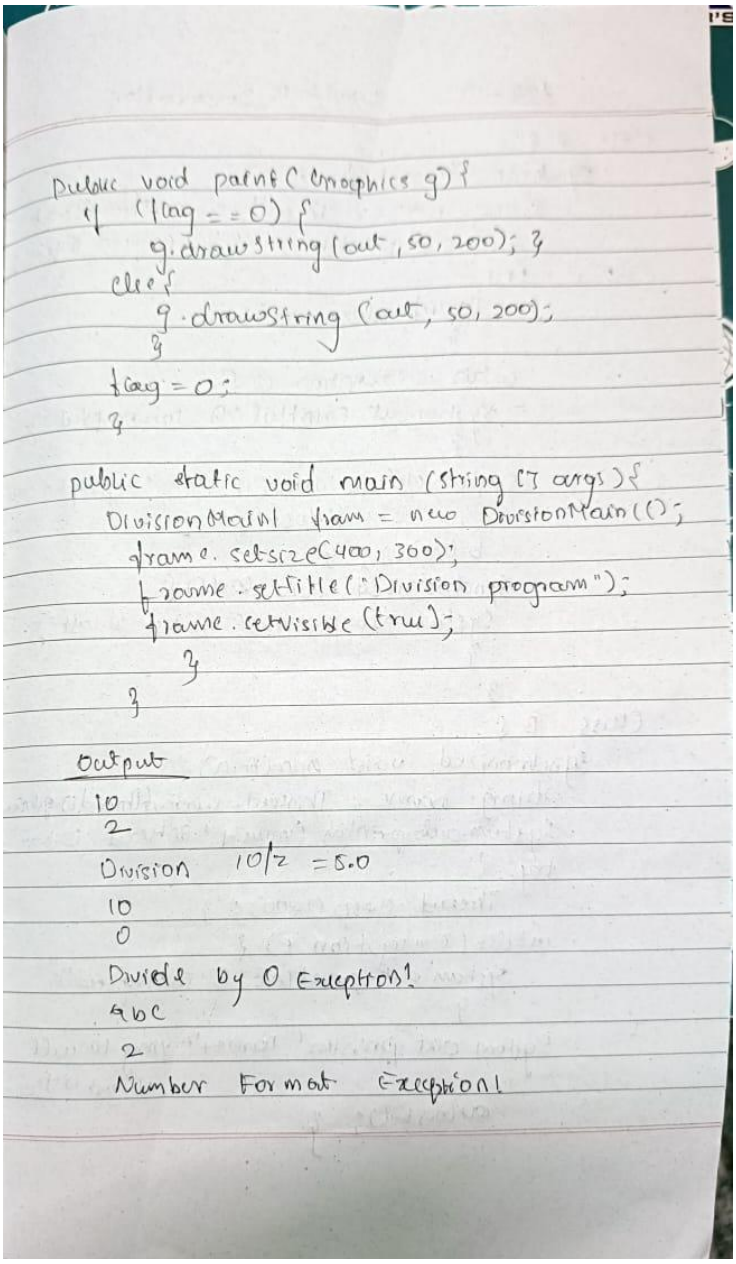


LAB - 09 09-12-2024

```
import java.awt.*;
import java.awt.event.*;
public class DivisionMain1 extends Frame implements
  ActionListener {
  TextField num1, num2;
  Button dResult;
  Label outResult;
  String out = " ";
  double resultNum;
  int flag = 0;

  public DivisionMain()
    { setLayout(new FlowLayout());
      dResult = new Button("RESULT");
      Label number1 = new Label("Number1 :",
                                Label.RIGHT);
      Label number2 = new Label("Number2:",
                                Label.RIGHT);
      num1 = new TextField(5);
      num2 = new TextField(5);
      outResult = new Label("Result :", Label.CE
                                        NTER);
  add(number1);
      add(number2);
      add(dResult)
      add(outResult);
      number1.addActionListener(this);
      number2.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 ArithmiticException();
  }
  out = "Division :" + n1 + "/" +n2 + "=";
  resultNum = (double) n1/n2;
  out += resultNum;
  repaint(); }
catch( NumberFormatException e1){
    flag =1;
    out = "Number Format Exception!";
  repaint(); }
catch (ArithmaticException e2){
    flag =1;
    out = "Divide by 0 Exception!";
  repaint();
}
}
```



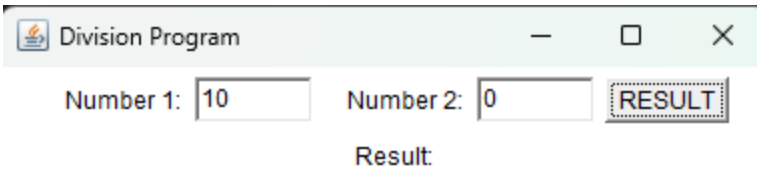
```
Dubuc void paint( Graphics g){
if (flag == 0) {
    g.drawString (out, 50, 200); }
else{
    g.drawString (out, 50, 200);
    }
flag = 0;
}

public static void main (String [] args){
  DivisionMain1 fram = new DivisionMain1();
   frame.setSize(400, 300);
   frame.setTitle("Division program");
   frame.setVisible (true);
     }
}

Output
10
2
Division  10/2 = 5.0
10
0
Divide by 0 Exception!
abc
2
Number Format Exception!
```

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

public class DivisionMain1 extends Frame implements ActionListener {
   TextField num1, num2;
   Button dResult;
```

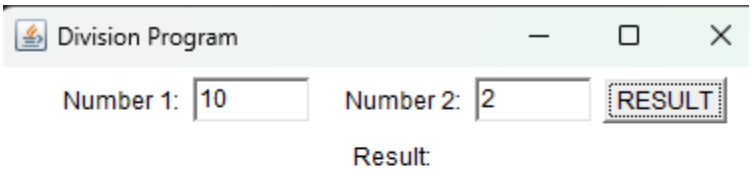
```
Label outResult;
String out = "";
double resultNum;
int flag = 0;

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 = "Division: " + n1 + " / " + n2 + " = ";
        resultNum = (double) n1 / n2;
        out += resultNum;
        repaint();
      }
    } catch (NumberFormatException e1) {
      flag = 1;
      out = "Number Format Exception!";
      repaint();
    } catch (ArithmeticException e2) {
```

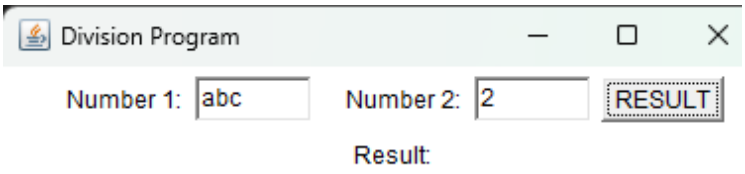
```
            flag = 1;
            out = "Divide by 0 Exception!";
            repaint();
        }
    }

    public void paint(Graphics g) {
        if (flag == 0) {
            g.drawString(out, 50, 200);
        } else {
            g.drawString(out, 50, 200);
        }
        flag = 0;
    }

    public static void main(String[] args) {
        DivisionMain1 frame = new DivisionMain1();
        frame.setSize(400, 300);
        frame.setTitle("Division Program");
        frame.setVisible(true);
    }
}
```



Divide by 0 Exception!

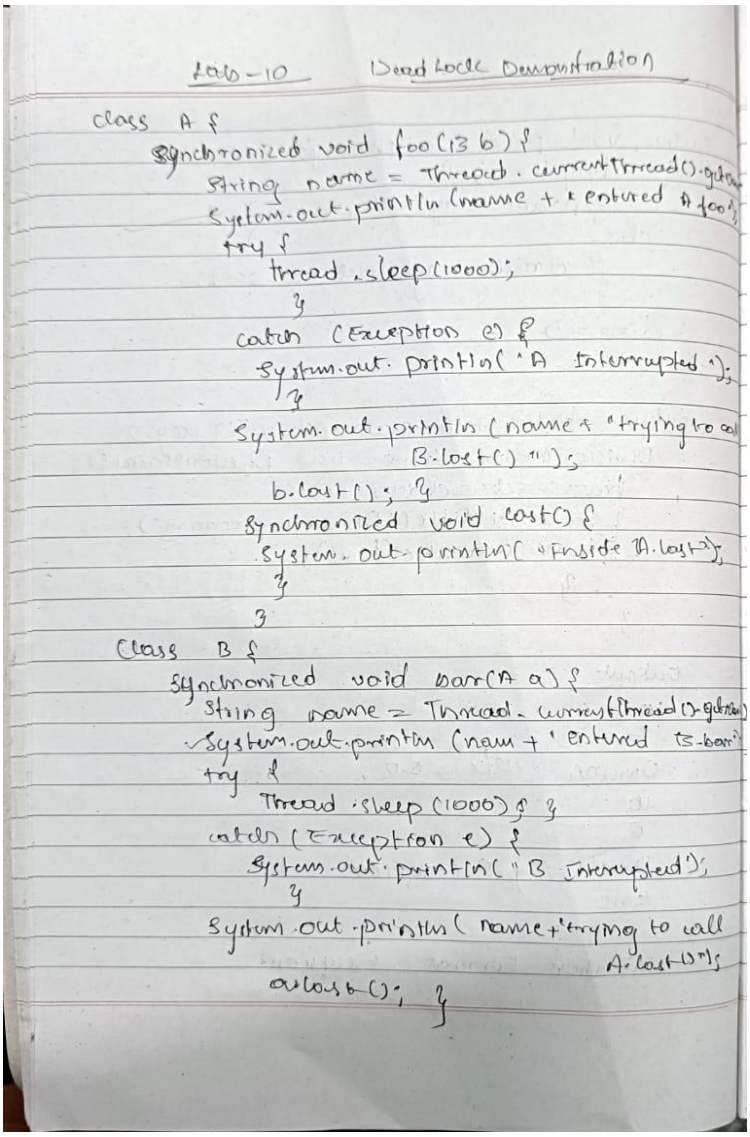


Division: 10 / 2 = 5.0

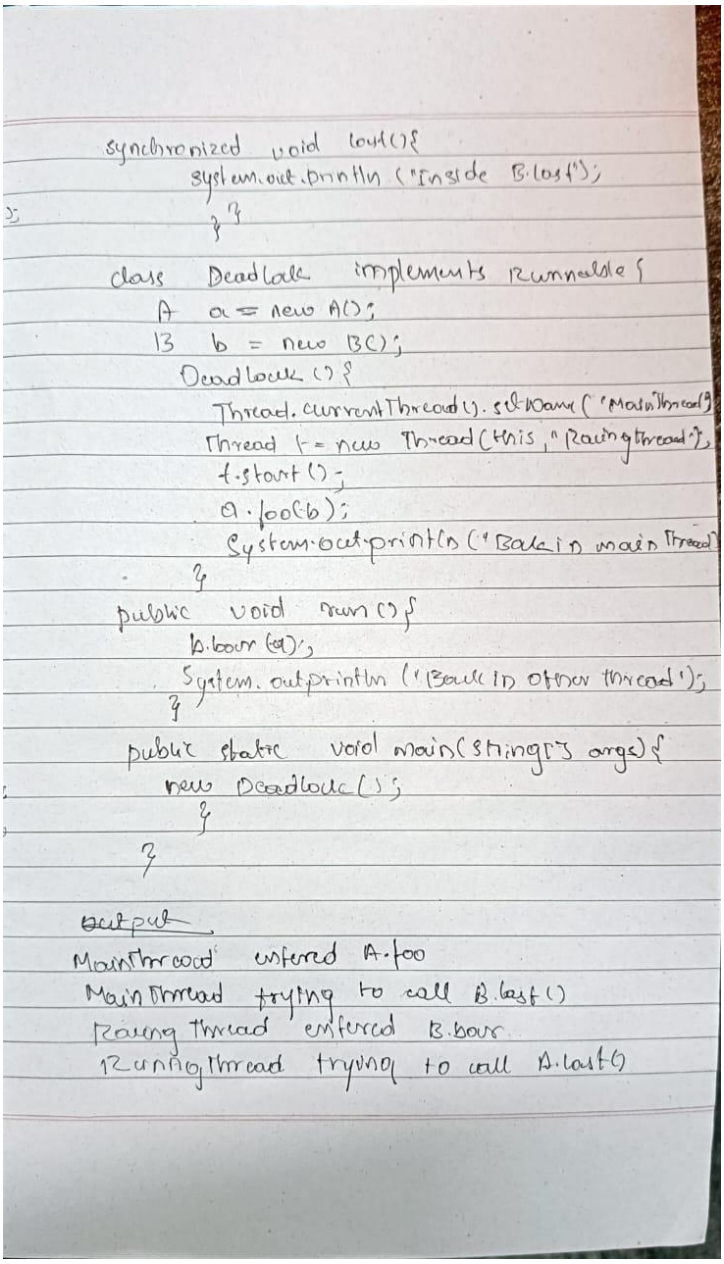


Number Format Exception!

10. Demonstrate Inter process Communication and deadlock



```
Lab-10        Dead Lock Demonstration

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();
  }
}
```




```
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  B.last");
    }
}

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

    Deadlock() {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();

        a.foo(b); // Get lock on a in this thread.
        System.out.println("Back in main thread");
    }

    public void run() {
        b.bar(a); // Get lock on b in other thread.
        System.out.println("Back in other thread");
    }
```

```
    public static void main(String[] args) {
        new Deadlock();
    }
}
```

```
C:\Users\bpsuh\Downloads\ds lab>java Deadlock
RacingThread entered B.bar
MainThread entered A.foo
RacingThread trying to call A.last()
MainThread trying to call B.last()
```