VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming (23CS3PCOOJ)

Submitted by

Kratish Porwal(1BM23CS157)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Sep-2024 to Jan-2025

B.M.S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Object Oriented Java Programming (23CS3PCOOJ)" carried out by Kratish Porwal **(1BM23CS157),** who is Bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Object-Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

Ambuja K Assistant Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

Index

Github Link(all programs): https://github.com/KratishPorwal/JAVA-LAB-PROGRAMS

Program 1: Develop a Java program that prints all real solutions to the quadratic equation $ax^2+bx+c = 0$. Read in a, b, c and use the quadratic formula. If the discriminate b^2-4ac is negative, display a message stating that there are no real solutions.

GitHubLink:https://github.com/KratishPorwal/JAVA-LAB-PROGRAMS/blob/main/quadratic%20equation.docx

Algorithm:

30/09/24

LAB program 1

Develop a Java program that prints all real solutions to the quadratic equation $ax^2+bx+c=0$. Read in a, b, c and use the quadratic formula. If the discriminate b^2-4ac is negative, display a message stating that there are no real solutions

```
import static java.lang.Math.sqrt;
import java.util.Scanner;

class Quadratic {
    int a, b, c;
    double r1, r2, d;

void input () {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter value of a:");
    a = sc.nextInt();

while (a==0) {
        System.out.println("Enter a non zero number");
        a = sc.nextInt();
    }
    System.out.print("Enter value of b:");
    b = sc.nextInt();
    System.out.print("Enter value of c:");
    c = sc.nextInt();

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

```
Void display () {
    if (d==0) {
        r1 = -b / (2.0*a);
        System.out.println("Roots are real and equal")
        System.out.println("Root : " + r1);
    } else if (d>0) {
        r1 = (-b + sqrt(d)) / (2.0*a);
        r2 = (-b - sqrt(d)) / (2.0*a);
        System.out.println("Roots are real and different
        System.out.println("r1 = " + r1 + ", r2 = " + r2)
    } else {
        r1 = -b/(2.0*a);
        r2 = sqrt(-d) / (2.0*a);
        System.out.println("Roots are imaginary");
        System.out.println("r1 = " + r1 + " + " + r2 + "i");
        System.out.println("r2 = " + r1 + " - " + r2 + "i");
        System.out.println("Kratish Porwal : 1BM23CS375")
    }
}

Public static void main (String [] args) {
    quadratic qe = new quadratic ();
    qe.input ();
    qe.display ();
}

}
```

Output :

```
Enter value of a : 0
Enter a non-zero number for a :
1
Enter value of b : 2
Enter value of c : 4
Roots are imaginary
r1 = -1.0 + 1.732050...i
r2 = -1.0 - 1.7320...i


Enter value of a : 1
Enter value of b : -7
Enter value of c : 10
Roots are real and different
    r1 = 5.0 , r2 = 2.0
```

20/9/24

Code:

```
import java.util.Scanner;

public class Quadratic

{

public static void main(String[] args)

{

int a;

int b;

int c;

Scanner sc = new Scanner(System.in);

System.out.print("Enter 'a' value: ");

a= sc.nextInt();

System.out.print("Enter 'b' value: ");

b=sc.nextInt();

System.out.print("Enter 'c' value: ");

c=sc.nextInt();

float disc = ((b*b)-4*a*c);

System.out.println(disc);

if (a==0)

{

System.out.println("Not Quadratic");

}

else

{

if (disc<0)
```

```
{
System.out.println("No real roots ");
}
else if (disc>0)
{
double root1= (-b + Math.sqrt(disc))/(2*a);
double root2= (-b - Math.sqrt(disc))/(2*a);
System.out.println("Real roots ");
System.out.println("Root-1: "+root1);
System.out.println("Root-2: "+root2);
}
else
{
double root1=(-b)/(2*a);
System.out.println("Real and equal");

System.out.println("Root-1: "+root1);
System.out.println("Root-2: "+root1);

}
System.out.println("Karan");
System.out.println("1BM23CS139");

}
}
```

```
D:\1BM23CS157>java Quadratic.java
Enter value of a: 0
Enter a non-zero number for a:
1
Enter value of b: 2
Enter value of c: 4
Roots are imaginary
r1 = -1.0 + 1.7320508075688772i
r2 = -1.0 - 1.7320508075688772i
Kratish Porwal: 1BM23CS157

D:\1BM23CS157>java Quadratic.java
Enter value of a: 3
Enter value of b: 5
Enter value of c: 7
Roots are imaginary
r1 = -0.8333333333333334 + 1.2801909579781012i
r2 = -0.8333333333333334 - 1.2801909579781012i
Kratish Porwal: 1BM23CS157

D:\1BM23CS157>java Quadratic.java
Enter value of a: 1
Enter value of b: -7
Enter value of c: 10
Roots are real and different
r1 = 5.0, r2 = 2.0

D:\1BM23CS157>
```

}

Program 2: Develop a Java program to create a class Student with members usn, name, an array credits and an array mark. Include methods to accept and display details and a method to calculate SGPA of a student.

GithubLink: https://github.com/KratishPorwal/JAVA-LAB-PROGRAMS/blob/main/SGPA

Algorithm:

LAB 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 & display details & a method to calculate SGPA of a student

```
import java.util.Scanner;

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

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

    Scanner sc = new Scanner(System.in);

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

        gradepoints[j] = (marks[j]/10) + 1;
```

```
    if (gradepoints[j] >10){
        gradepoints[j] =10;
    }
}
}

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

void display() {
    System.out.println("name: "+name);
    System.out.println("USN: "+usn);
    System.out.println("SGPA: "+SGPA);
}
public static void main(String[] args) {
    Student s1 = new Student();
        s1.getStudentDetails();
        s1.getMarks();
        s1.computeSGPA();
        s1.display();
    }

}
```

```
output
Enter name:
Krathik
Enter USN
1BM23CS157

Enter 1 Subject Marks:
91
Enter credits for Subject 1:
1
Enter 2 Subject marks:
67
Enter credits for Subject 2:
4
Enter 3 Subject marks:
73
Enter credits for subject 3:
3
Enter 4 subject marks
81
Enter credits for subject 4:
3

Name: Krathik
USN: 1BM23CS157
SGPA: 8.02609
```

7/11/24

Code:

```
import java.util.Scanner;

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

System.out.print("Enter the number of students: ");
int numStudents = sc.nextInt();
sc.nextLine();

String[] names = new String[numStudents];
String[] usns = new String[numStudents];
int[][] creditsArray = new int[numStudents][];
int[][] marksArray = new int[numStudents][];
double[] sgpas = new double[numStudents];

for (int s = 0; s < numStudents; s++) {
System.out.println("Enter details for student " + (s + 1) + ":");

System.out.print("Enter your name: ");
names[s] = sc.nextLine();
System.out.print("Enter your USN: ");
usns[s] = sc.nextLine();

System.out.print("Enter the number of subjects: ");
int numSubjects = sc.nextInt();
```

```
int[] credits = new int[numSubjects];
System.out.println("Enter the credits for each subject:");
for (int i = 0; i < numSubjects; i++) {
credits[i] = sc.nextInt();
}
creditsArray[s] = credits;

int[] marks = new int[numSubjects];
System.out.println("Enter the marks for each subject out of 100:");
for (int i = 0; i < numSubjects; i++) {
marks[i] = sc.nextInt();
}
marksArray[s] = marks;

int[] gradePoints = new int[numSubjects];
int[] resultArray = new int[numSubjects];
for (int i = 0; i < numSubjects; i++) {
gradePoints[i] = (marks[i] / 10) + 1;
resultArray[i] = credits[i] * gradePoints[i];
}

int totalCredits = sum(credits);
int totalResult = sum(resultArray);
if (totalCredits > 0) {
sgpas[s] = (double) totalResult / totalCredits;
```

```
} else {
sgpas[s] = 0.0;
}
}

System.out.println("\n--- Results ---");
for (int s = 0; s < numStudents; s++) {
System.out.println("Student " + (s + 1) + " (" + names[s] + ", " + usns[s] + "):");
System.out.print("Credits: ");
for (int credit : creditsArray[s]) {
System.out.print(credit + " ");
}
System.out.println();
System.out.print("Marks: ");
for (int mark : marksArray[s]) {
System.out.print(mark + " ");
}
System.out.println();
System.out.println("SGPA: " + sgpas[s]);
System.out.println();
}}
static int sum(int[] array) {
int sum = 0;
for (int value : array) {
sum += value;
}
```

return sum;

```
Enter the number of students: 2
Enter details for student 1:
Enter your name: karan
Enter your USN: 139
Enter the number of subjects: 3
Enter the credits for each subject:
4
3
3
Enter the marks for each subject out of 100:
88
82
91
Enter details for student 2:
Enter your name: Enter your USN: vinod 132
Enter the number of subjects: 3
```

```
Enter details for student 2:
Enter your name: Enter your USN: vinod 132
Enter the number of subjects: 3
Enter the credits for each subject:
4
3
3
Enter the marks for each subject out of 100:
78
98
73

--- Results ---
Student 1 (karan, 139):
Credits: 4 3 3
Marks: 88 82 91
SGPA: 9.3

Student 2 (, vinod 132):
Credits: 4 3 3
Marks: 78 98 73
SGPA: 8.6
```

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.

GithubLink: https://github.com/KratishPorwal/JAVA-LAB-PROGRAMS/tree/main/Books

Algorithm:

4/10/

LAB Program 3

Q) Create a class Book which contains 4 members: name, author, price, num-pages. Include a constructor to set the values for the members. Include methods to set & get the details of the object. Include a toString() method that could display the complete details of the Book. Develop a Java program to create n book objects

```
Import java.util.Scanner;
class Books
{
    String name;
    String author;
    int price;
    int numPages;

    Books (String name, String author, int price, int numPages)
    {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }
    Public String toString()
    {
        String name, author, price, numPages;
        name = "Book name: " + this.name + "\n";
        author = "Author name: " + this.author + "\n";
        price = "Price: " + this.price + "\n");
        numPages = "Number of pages: " + this.numPages + "\n";
```

```
        return name + author + price + numPages ;
    }
    public Static Void main (String args[])
    {
        Scanner sc = new Scanner (System.in);
        int n;
        String name;
        String author;
        int price;
        int numPages;
        System.out.Print ln ("Enter the number of books")
        n = sc.nextInt ();
        Books b[];
        b = new Books [n];
        System.out.println ("Enter book details");
        for (int i=0 ; i<n ; i++)
        {
            System.out.println ("enter the book" + (i+1) + "name");
            name = sc.next ();
            System.out.print ln ("enter the book" + (i+1) + "author")
            author = sc.next ()
            System.out.print ln("enter the book" + (i+1) + "price")
            price = sc.nextInt ();
            System.out.println ("enter the book" + (i+1) + "pages")
            numPages = sc.nextInt ();
            b [i] = new Books (name, author, price, numPages);
        }
        for (int i=0 ; i<n ; i++)
        {
            b[i].toString ();
            System.out.Print ln (b[i]);
        }
    }
}
```

output

```
Enter the number of books
 2
Enter Book details
enter the Book 1 name
harry
enter the Book 1 author
Rowling
enter the Book 1 price
 699
enter the Book 1 pages
 777

Enter the Book 2 name
Inferno
Enter the Book 2 author
Brown
Enter the Book 2 price
 899
Enter the Book 2 pages
 789

Book name : harry
Author name : rowling
Price : 699
Number of pages : 777

Book name : inferno
Author name : Brown
Price : 899
Number of Pages : 789
```

10/10/24

Code:

```
import java.util.Scanner;

class Books

{

   String name;

   String author;

   int price;

   int numPages;


   Books(String name,String author,int price,int numPages)

{

   this.name=name;

   this.author=author;

   this.price=price;

   this.numPages=numPages;

}

public String toString()

{

   String name, author, price, numPages;

   name="Book name:"+this.name+"\n";

   author ="Author name:"+this.author+"\n";

   price ="Price:"+this.price+"\n";

   numPages="Number of pages:"+this.numPages+"\n";

   return name+author+price+numPages;

}

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

```
{
    Scanner sc=new Scanner(System.in);
    int n;
    String name;
    String author;
    int price;
    int numPages;
    System.out.println("enter the number of book");
    n=sc.nextInt();
    Books b[];
    b=new Books[n];
    System.out.println("Enter book details");
    for (int i=0;i<n;i++)
    {
      System.out.println("enter the book"+(i+1)+"name");
      name=sc.next();
      System.out.println("enter the book"+(i+1)+"author");
      author=sc.next();
      System.out.println("enter the book"+(i+1)+"price");
      price=sc.nextInt();
      System.out.println("enter the book"+(i+1)+"pages");
      numPages=sc.nextInt();
      b[i]=new Books(name,author,price,numPages);
    }
    for(int i=0;i<n;i++)
    {
```

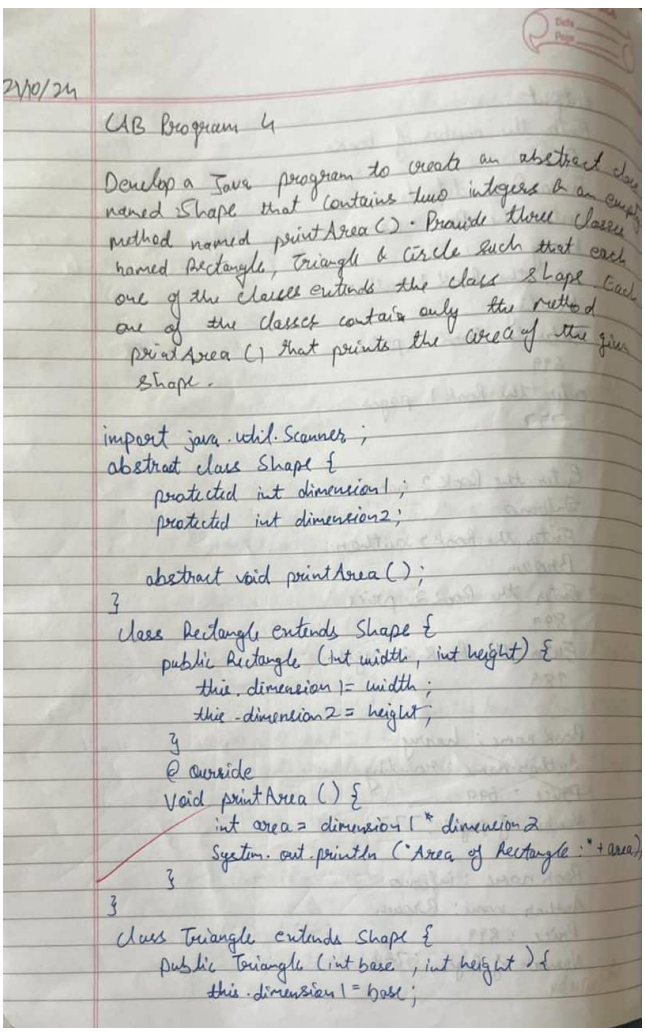
```
      b[i].toString();
      System.out.println(b[i]);
    }
}
}
```

```
D:\1BM23CS157>java Books.java
enter the number of book
2
Enter book details
enter the book1name
harry
enter the book1author
rowling
enter the book1price
699
enter the book1pages
777
enter the book2name
inferno
enter the book2author
brown
enter the book2price
899
enter the book2pages
789
Book name:harry
Author name:rowling
Price:699
Number of pages:777

Book name:inferno
Author name:brown
Price:899
Number of pages:789


D:\1BM23CS157>
```

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.

GithubLink:https://github.com/KratishPorwal/JAVA-LAB-PROGRAMS/tree/main/Shapes

Algorithm:



21/10/24

LAB Program 4

Develop a Java program to create an abstract class named Shape that contains two integers & an empty method named printArea( ). Provide three classes named Rectangle, Triangle & 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 {
    protected int dimension1;
    protected int dimension2;

    abstract void printArea();
}
class Rectangle extends Shape {
    public Rectangle (int width, int height) {
        this.dimension1 = width;
        this.dimension2 = height;
    }
    @ Override
    void printArea () {
        int area = dimension1 * dimension2
        System.out.println ("Area of Rectangle :" + area)
    }
}
class Triangle extends Shape {
    public Triangle (int base, int height ) {
        this.dimension1 = base;
```

```
        this.dimension2 = height;
    }
    @Override
    void printArea(){
        double area = 0.5 * dimension1 * dimension2;
        System.out.println("Area of Triangle: " + area);
    }
}
class circle extends Shape {
    public circle (int radius) {
        this.dimension1 = radius;
    }
    @Override
    void printArea() {
        double area = Math.PI * dimension1 * dimension1;
        System.out.println("Area of circle " + area);
    }
}

Public class Shapes {
    public static void main (String [] args) {
        System.out.println("enter the dimensions of the
                                rectangle: (length & breadth):");
        Scanner sc = new Scanner (System.in);
        int l1 = sc.nextInt();
        int b1 = sc.nextInt();
        Shape rectangle = new Rectangle (l1, b1);
        rectangle.printArea();
        System.out.println("enter dimensions of the
                                triangle (base & height):");
        int b2 = sc.nextInt();
        int h1 = sc.nextInt();
```

```
        Shape triangle = new triangle (b2, h1);
        triangle.printArea()
        System.out.println("enter the dimensions of
        ~~int r1 = sc.nextInt~~ the circle (radius):");
        int r1 = sc.nextInt();
        Shape circle = new circle (r1);
        circle.printArea();
        }
}
```

Output

```
Enter the dimensions of the rectangle: (length & breadth)
 6
 8
Area of Rectangle: 48
Enter the dimensions of triangle: (base & height):
 8
 12
Area of Triangle: 48.0
Enter the dimensions of the circle (radius):
 7
Area of circle: 153.93804
```

Dy 20/10

Code:

```
import java.util.Scanner;
abstract class Shape {
   protected int dimension1;
   protected int dimension2;

   abstract void printArea();
}

class Rectangle extends Shape {
   public Rectangle(int width, int height) {
      this.dimension1 = width;
      this.dimension2 = height;
   }

   @Override
   void printArea() {
      int area = dimension1 * dimension2; // Area = width * height
      System.out.println("Area of Rectangle: " + area);
   }
}
class Triangle extends Shape {
   public Triangle(int base, int height) {
      this.dimension1 = base;
      this.dimension2 = height;
   }
```

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

class Circle extends Shape {
    public Circle(int radius) {
        this.dimension1 = radius; // Radius
    }

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

public class Shapes {
    public static void main(String[] args) {
        System.out.println("enter the dimensions of the rectangle:(length and breadth):");
        Scanner sc = new Scanner(System.in);
        int l1=sc.nextInt();
        int b1=sc.nextInt();
```

```
        Shape rectangle = new Rectangle(l1, b1);

        rectangle.printArea();

        System.out.println("enter the dimensions of the traingle:(base and height):");

        int b2 = sc.nextInt();

        int h1=sc.nextInt();

        Shape triangle = new Triangle(b2, h1);

        triangle.printArea();

        System.out.println("enter the dimensions of the circle:(radius):");

        int r1=sc.nextInt();

        Shape circle = new Circle(r1);

        circle.printArea();

    }

}
```

```
D:\1BM23CS157>javac Shapes.java

D:\1BM23CS157>java Shapes
enter the dimensions of the rectangle:(length and breadth):
6
8
Area of Rectangle: 48
enter the dimensions of the traingle:(base and height):
8
12
Area of Triangle: 48.0
enter the dimensions of the circle:(radius):
7
Area of Circle: 153.93804002589985

D:\1BM23CS157>
```

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

GithubLink:https://github.com/KratishPorwal/JAVALABPROGRAMS/tree/main/Bank%20program

Algorithm:

LAB Program 5

Develop a Java program to create a class Bank that maintains two kinds of accounts for its customers, one called savings account & the other current account. The savings account provides compound interest & 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 & update the balance
b) Display the balance
c) compute & deposit interest
d) Permit withdrawal and update the balance
e) Check for the minimum balance, impose penalty if necessary and update the balance

```
import java.util.Scanner;

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

    Account (String name, int accNumber, String accType) {
        customerName = name;
        accountNumber = accNumber;
        accountType = accType;
        balance = 0;
    }
    ~~public void deposit (double amount)~~
    Public void deposit (double amount) {
        balance += amount;
        System.out.println("Deposited: " + " updated balance
        balance);
    }
    Public void displayBalance () {
        System.out.println("Account Balance : " + balance)
    }
    Public void withdraw (double amount) {
        System.out.println("This operation is specific to
        account type.");
    }
}
class SavAccount extends Account {
    double interestRate = 0.04;
    SavAccount (String name, int accNumber) {
        Super (name, accNumber, "Savings");
    }
    Public void computeInterest () {
        double interest = balance * interestRate;
        balance += interest;
        System.out.println("Interest added : " + interest
        . updated balance : " + balance);
    }
```

```
@Override
    Public void withdraw (double Amount) {
        if (balance >= amount){
            balance -= amount ;
            System.out.println ("withdrawn: " + amount + "
              updated balance: " + balance );
        }
        else {
            System.out.println ("Insufficient balance") ;
        }
    }
}
class CurAccount extends Account {
    double minBalance = 500.0;
    double serviceCharge = 50.0;

    CurAccount (String name, int accNumber) {
        super (name, accNumber, "current");
    }
    Public void checkMinBalance () {
        if (balance < minBalance) {
            balance -= Service charge ;
            System.out.println ("Balance below minimum. Service
              charge imposed: " + Service Charge + ", updated balance
              + balance);
        }
    }
    @ override
    Public void withdraw (double amount) {
        if (balance >= amount ) {
            balance -= amount;
            System.out.println ("withdrawn: " + amount +
              updated balance: " + balance);
```

```
        check Min Balance();
    }
    else {
        System.out.Println ("Insufficient balance.");
    }
  }
}

Public class Bank {
    public static void main (String [] args ) {
        Scanner sc = new Scanner (System.in);
        System.out.println ("Enter customer name:") ;
        String name = sc.next();
        System.out.print ln ("Enter account number:");
        int accountNumber = sc.nextInt();
        SavAccount SavingsAccount = new SavAccount (name,
          account number);
        System.out.Print ln ("Enter customer Name:");
        String name1 = sc.next ();
        System.out.Print ln ("Enter account number:") ;
        int accountNumber1 = sc.nextInt() ;
        CurAccount currentAccount = new CurAccount (name1,
          accountnumber1);

        While (true) {
            System.out.print ln ("\n ---Menu---" );
            System.out.print ln (" 1 Deposit \n2 Withdraw \n3.
            compute Interest for Savings Account \n4 Display
            Account Details \n5. Exit ");
            System.out.print ("Enter your choice:");
            int choice = sc.nextInt ();
```

```
System.out.print("Enter the type of account (Saving/current)
String accType = Sc.next
If (accType.equals("Saving")){
  Switch (choice){
    Case 1:
      System.out.print("Enter the deposit amount:");
      double depositAmount = Sc.nextDouble();
      SavingsAccount.deposit(depositAmount);
      break;
    Case 2:
      System.out.print("Enter the withdrawal amount:");
      double withdrawalAmount = Sc.nextDouble();
      SavingsAccount.withdraw(withdrawalAmount);
      break;
    Case 3:
      SavingsAccount.computeInterest
      break;
    Case 4:
      System.out.println("customer name:"+ SavingsAccount.customerName
      System.out.println("Account number:"+ SavingsAccount.accountNumber
      System.out.println("Type of Account:"+ SavingsAccount.accountType
      SavingsAccount.displayBalance();
      break;
    default:
      System.out.println("Invalid choice");
  }
else if (accType.equals("current")){
  Switch (choice){
    case 1:
      System.out.print("Enter the deposit amount:");
      double depositAmount = sc.nextDouble();
      currentAccount.deposit(depositAmount);
```

```
Enter account number:
157
---MENU---
1 Deposit
2. withdraw
3. compute Interest for Savings Account
4. Display Account Details
5) Exit
Enter your choice: 1
Enter the type of account (Saving/current): Saving
Enter the deposit amount: 8000
Deposited 8000.0 updated balance: 8000.0

---- MENU ---
1 Deposit
2 withdraw
3 Compute Interest for Savings account
4 Display Account Details
5 Exit
```

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

Output

```
Enter customer Name:
Keerthi
Enter Account number:
157
Enter customer name:
Keerthi
```

Code:

```
import java.util.Scanner;

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

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

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

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

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

class SavAccount extends Account {
    double interestRate = 0.04;  // 4% annual interest rate

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

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

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

```
    }
}
class CurAccount extends Account {
    double minBalance = 500.0;
    double serviceCharge = 50.0;

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

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

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

```
    }
  }
}

public class Bank {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter customer name:");
    String name=sc.next();
    System.out.println("Enter account number:");
    int accountnumber=sc.nextInt();
    SavAccount savingsAccount = new SavAccount(name, accountnumber);
System.out.println("Enter customer name:");
    String name1=sc.next();
    System.out.println("Enter account number:");
    int accountnumber1=sc.nextInt();
    CurAccount currentAccount = new CurAccount(name1, accountnumber1);

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

      System.out.print("Enter the type of account (saving/current): ");
```

```
String accType = sc.next();


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

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

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

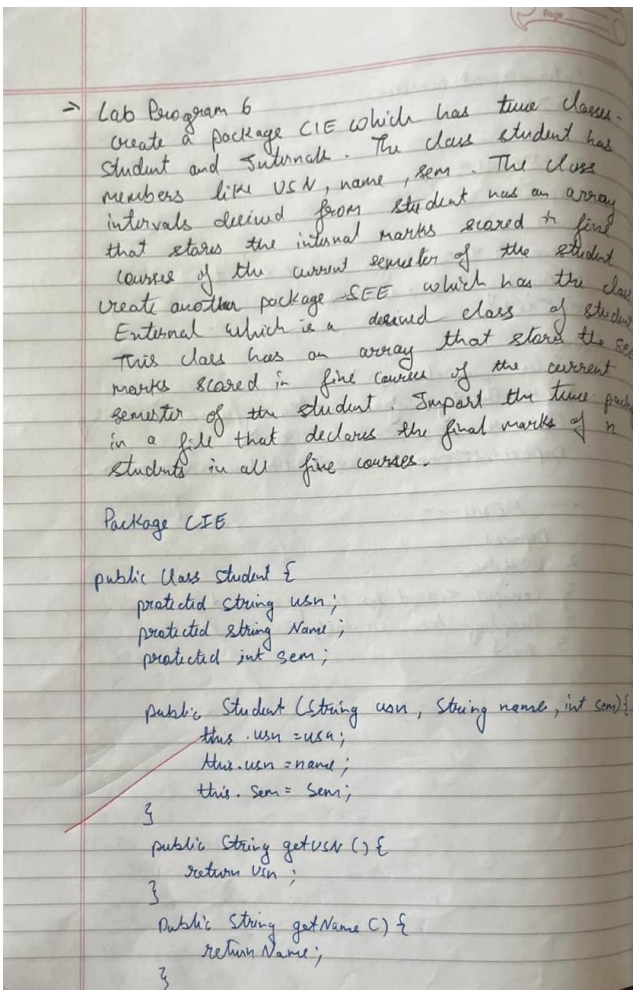
```
D:\1BM23CS157>java Bank
Enter customer name:
kratish
Enter account number:
157
Enter customer name:
kratish
Enter account number:
1571

-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 1
Enter the type of account (saving/current): saving
Enter the deposit amount: 8000
Deposited: 8000.0. Updated balance: 8000.0

-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 1
Enter the type of account (saving/current): current
Enter the deposit amount: 8000
Deposited: 8000.0. Updated balance: 8000.0

-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 3
Enter the type of account (saving/current): saving
Interest added: 320.0. Updated balance: 8320.0
```

```
-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 3
Enter the type of account (saving/current): current
Current accounts do not earn interest.

-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 4
Enter the type of account (saving/current): saving
Customer name: kratish
Account number: 157
Type of Account: Savings
Account Balance: 8320.0

-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 2
Enter the type of account (saving/current): current
Enter the withdrawal amount: 7800
Withdrawn: 7800.0. Updated balance: 200.0
Balance below minimum. Service charge imposed: 50.0. Updated balance: 150.0
```

Program 6: Create a package CIE which has two classes- Student and Internals. The class Personal has members like usn, name, sem. The class internals have 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.

GithubLink: https://github.com/KratishPorwal/JAVA-LAB-PROGRAMS/tree/main/bmsce

Algorithm:



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

```
Package CIE

public class Student {
    protected string usn;
    protected string Name;
    protected int sem;

    public Student (String usn, String name, int sem){
        this.usn = usn;
        this.usn = name;
        this.sem = sem;
    }
    public String getUsn () {
        return usn;
    }
    Public String getName () {
        return Name;
    }
```

```
    public int get Sem () {
        return sem ;
    }
}

Package CIE

Public class Internals extends Student {
    private int [] internal marks ;

    public Internals (String USN , String Name , int Sem ,
      int [] Internal Marks ) {
        Super (USN , name , Sem ) ;
        this . Internal Marks = internal marks ;
    }
    public int [] get Internal Marks () {
        return Internal Marks ;
    }
}

Package SEE ;

import CIE. Student ;

Public class External extends Students {
    private int [] External Marks ;

    Public External (String USN , String Name , int Sem
    , int [] external Marks ) {
        Super (usn , name , sem ) ;
        this . external Marks = external Marks ;
    }
```

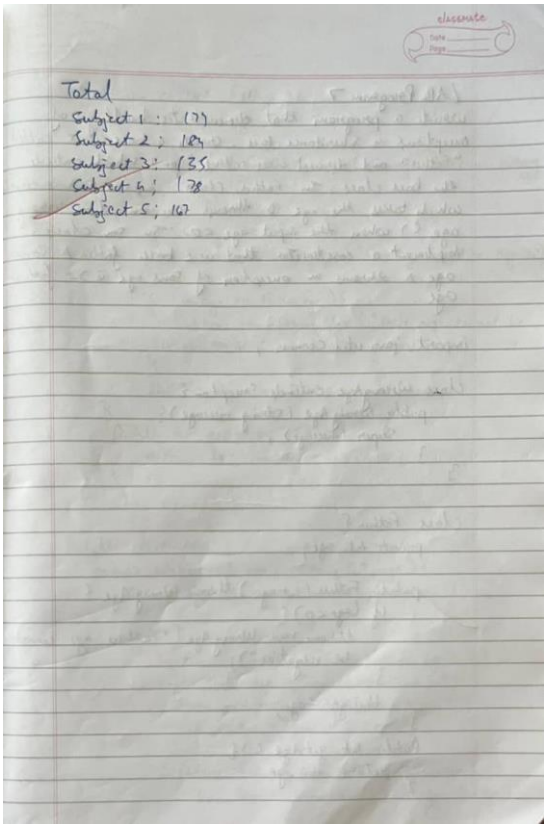
```
    Public int [] get External Marks () {
        return external Marks ;
    }
}

import CIE. Internals
import SEE. Externals
import Java. util. Scanner

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

        Internals [] internals = new Internals [n] ;
        External [] externals = new External [n] ;

        for (int i = 0 ; i < n ; i++ ) {
            System. out. print ln ("Enter details for students "+
              (i+1) + ":" ) ;
            System. out. print ("USN : ") ;
            String usn = Scanner . next () ;
            System. out. print ("Name : ") ;
            String name = Scanner . next () ;
            System. out. print ("semester : ") ;
            int sem = Scanner . next Int () ;

            int [] Internal Marks = new int [5] ;
            System. out. print ln ("Enter internal marks for
              5 courses ") ;
```

```
for (int j = 0; j < s; j++){
        internalMarks[j] = scanner.nextInt();
}

int[] externalMarks = new int[s];
System.out.println("Enter external marks for s courses");
for (int j = 0; j < s; j++){
        externalMarks[j] = scanner.nextInt();
}

internals[i] = new Internals(usn, name, sem, internalMarks);
externals[i] = new External(usn, name, sem, externalMarks);
}

System.out.println("Final Marks of students:");
for (int i = 0; i < n; i++){
    System.out.println("Students " + internals[i].getName() +
    " (" + internals[i].getUsn() + ")");
    System.out.Println("Internal Marks: ");
    int[] internalMarks = internals[i].getInternalMarks();
    for (int mark : internalMarks){
        System.out.print(mark + " ")
    }
    System.out.println();

    int total = 0;
    for (int mark : internalMarks){
        total += mark;
    }
    for (int mark : externalMarks){
        total += mark;
    }
```

```
    System.out.println("Total Marks: " + total);
    System.out.println();
}
Scanner.close();
}
}
```

Output:

```
Enter Number of students: 2
Name: Kratz
USN: ABC123
Semester: 3
~~Enter details for~~ Enter CIE marks for 5 subjects
Subject 1: 79
Subject 2: 83
Subject 3: 76
Subject 4: 81
Subject 5: 71

Enter SEE marks for 5 subjects:
Subject 1: 81
Subject 2: 73
Subject 3: 45
Subject 4: 77
Subject 5: 83

(similar for student 2)
```



Code:

```
package CIE;


public class Student {

   protected String usn;

   protected String name;

   protected 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;
    }
}
package CIE;

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

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

    public int[] getInternalMarks() {
        return internalMarks;
    }
```

```
}
package SEE;

import CIE.Student;

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

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

   public int[] getExternalMarks() {
      return externalMarks;
   }
}
import CIE.Internals;
import SEE.External;

import java.util.Scanner;

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

```
int n = scanner.nextInt();

Internals[] internals = new Internals[n];
External[] externals = new External[n];

for (int i = 0; i < n; i++) {
    System.out.println("Enter details for Student " + (i + 1) + ":");
    System.out.print("USN: ");
    String usn = scanner.next();
    System.out.print("Name: ");
    String name = scanner.next();
    System.out.print("Semester: ");
    int sem = scanner.nextInt();

    int[] internalMarks = new int[5];
    System.out.println("Enter internal marks for 5 courses:");
    for (int j = 0; j < 5; j++) {
        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++) {
        externalMarks[j] = scanner.nextInt();
    }
    internals[i] = new Internals(usn, name, sem, internalMarks);
    externals[i] = new External(usn, name, sem, externalMarks);
```

```
    }
    System.out.println("\nFinal Marks of Students:");
    for (int i = 0; i < n; i++) {
        System.out.println("Student: " + internals[i].getName() + " (" + internals[i].getUsn() +
")");
        System.out.println("Internal Marks: ");
        int[] internalMarks = internals[i].getInternalMarks();
        for (int mark : internalMarks) {
            System.out.print(mark + " ");
        }
        System.out.println();

        System.out.println("External Marks: ");
        int[] externalMarks = externals[i].getExternalMarks();
        for (int mark : externalMarks) {
            System.out.print(mark + " ");
        }
        System.out.println();
        int total = 0;
        for (int mark : internalMarks) {
            total += mark;
        }
        for (int mark : externalMarks) {
            total += mark;
        }
        System.out.println("Total Marks: " + total);
```

```
        System.out.println();
    }

    scanner.close();
  }
}
```

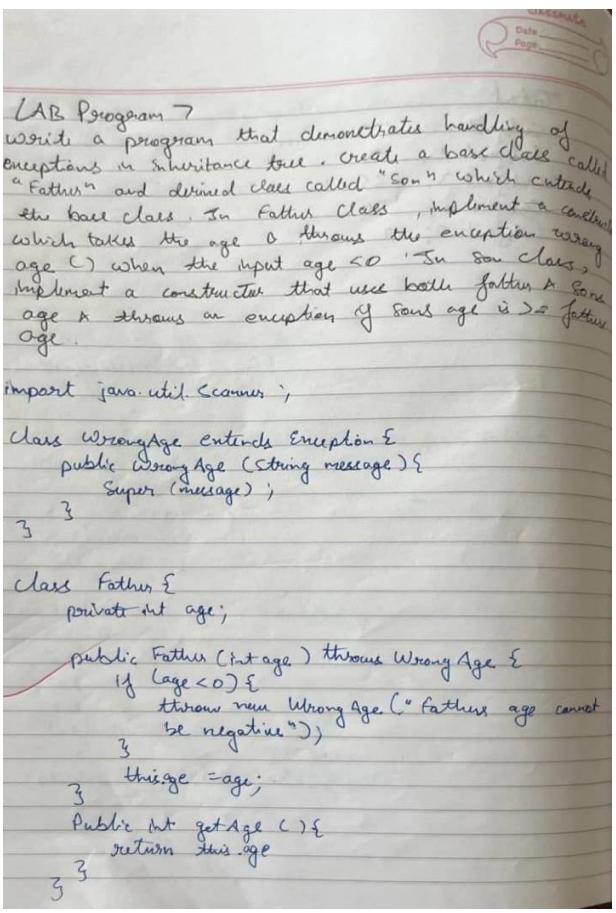
```
PS D:\1BM23CS139> cd "d:\1BM23CS139\" ; if ($?) { javac main.java } ; if ($?) { java main }

Enter the number of students:
2

Enter details for Students here1:
Enter your usn here:
139
Enter your name here:
Karna
Enter your semester here:
3
Enter your CIE marks for the 5 subjects here:
Subject1:98
Subject2:99
Subject3:100
Subject4:100
Subject5:97
CIE marks are as follows:

Subject1:98
Subject2:99
Subject3:100
Subject4:100
Subject5:97
Enter the 5 SEE Marks here:

Subject1:100

Subject2:100

Subject3:99

Subject4:98

Subject5:95
```

```
Enter details for Students here2:
Enter your usn here:
140
Enter your name here:
Karthik
Enter your semester here:
3
Enter your CIE marks for the 5 subjects here:
Subject1:99
Subject2:98
Subject3:99
Subject4:95
Subject5:91
CIE marks are as follows:

Subject1:99
Subject2:98
Subject3:99
Subject4:95
Subject5:91
Enter the 5 SEE Marks here:

Subject1:100

Subject2:100

Subject3:92

Subject4:91

Subject5:98

Finalmarks of students:

Student1:
Name:Karna
USN:139
Semester:3

The final marks of the 5 subjects are:

Subject1:99
Subject2:99
Subject3:99
Subject4:99
Subject5:96
```

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.

GithubLink:https://github.com/KratishPorwal/JAVALABPROGRAMS/tree/main/labprog%207

Algorithm:



LAB 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 & throws the exception wrongAge() when the input age <0. In Son class, implement a constructor that use both father & son age & throws an exception if sons age is >= fathers age.

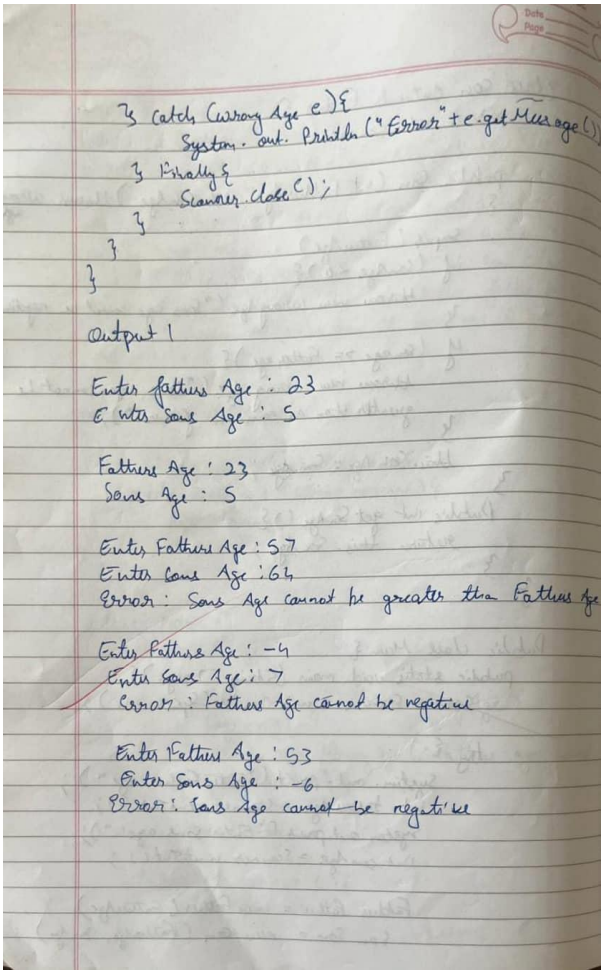
```
import java.util.Scanner;

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

class Father {
    private int age;

    public Father (int age) throws WrongAge {
        if (age < 0) {
            throw new WrongAge ("Fathers age cannot
            be negative");
        }
        this.age = age;
    }
    Public int getAge () {
        return this.age
    }
}
```

```
class Son extends Father {
    private int SonAge

    public Son (int FatherAge, int SonAge) throws WrongAge
    {
        super (FatherAge)
        if (SonAge < 0) {
            throw new WrongAge ("Sons age cannot be negative")
        }
        if (Son age >= Father age) {
            throw new WrongAge ("Sons age cannot be
            greater than or equal to father ");
        }
        this. Son Age = Son Age ;
    }
    Public int get SonAge () {
        return this. SonAge ;
    }
}

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

        try {
            System. out. print ("Enter Father age ") ;
            int father Age = Scanner nextInt ();
            System. out. print ("Enter Sons age: ")
            int son Age = Scanner. nextInt () ;

            Father Father = new Father ( FatherAge) ;
            Son Son = new Son (FatherAge, Son Age) ;
```

```
    } catch (WrongAge e) {
        System. out. Println ("Error" + e. get Message ())
    } Finally {
        Scanner close ();
    }
  }
}
```

Output 1

```
Enter father Age : 23
Enter Sons Age : 5

Father Age : 23
Sons Age : 5

Enter Father Age : 57
Enter Sons Age : 64
Error : Sons Age cannot be greater than Father Age

Enter Father Age : -4
Enter Sons Age : 7
Error : Father Age cannot be negative

Enter Father Age : 53
Enter Sons Age : -6
Error : Sons Age cannot be negative
```



```
} catch (WrongAge e){
        System.out.Println("Error"+e.getMessage());
    } Finally {
        Scanner.close();
    }
  }
}
```

Output 1

Enter fathers Age : 23
Enter Sons Age : 5

Fathers Age : 23
Sons Age : 5

Enter Fathers Age : 57
Enter Sons Age : 64
Error : Sons Age cannot be greater than Fathers Age

Enter Fathers Age : -4
Enter Sons Age : 7
Error : Fathers Age cannot be negative

Enter Fathers Age : 53
Enter Sons Age : -6
Error : Sons Age cannot be negative

Code:

```
import java.util.Scanner;


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


class Father {
   private int age;
```

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

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

class Son extends Father {
    private int sonAge;

    public Son(int fatherAge, int sonAge) throws WrongAge {
        super(fatherAge); // Calls the Father's constructor

        if (sonAge < 0) {
            throw new WrongAge("Son's age cannot be negative.");
        }

        if (sonAge >= fatherAge) {
            throw new WrongAge("Son's age cannot be greater than or equal to Father's age.");
        }
```

```
        this.sonAge = sonAge;
    }

    public int getSonAge() {
        return this.sonAge;
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try (
            System.out.print("Enter Father's age: ");
            int fatherAge = scanner.nextInt();

            System.out.print("Enter Son's age: ");
            int sonAge = scanner.nextInt();

            Father father = new Father(fatherAge);
            Son son = new Son(fatherAge, sonAge);

            System.out.println("Father's age: " + father.getAge());
            System.out.println("Son's age: " + son.getSonAge());

        } catch (WrongAge e) {
```

```
        System.out.println("Error: " + e.getMessage());
    } finally {
        scanner.close(); // Close the scanner to prevent resource leak
    }
  }
}
```

```
Microsoft Windows [Version 10.0.22631.4541]
(c) Microsoft Corporation. All rights reserved.

C:\1BM23CS157>javac Main.java

C:\1BM23CS157>java Main
Enter Father's age: 12
Enter Son's age: 5
Father's age: 12
Son's age: 5

C:\1BM23CS157>java Main
Enter Father's age: 56
Enter Son's age: 67
Error: Son's age cannot be greater than or equal to Father's age.

C:\1BM23CS157>java Main
Enter Father's age: -4
Enter Son's age: 5
Error: Father's age cannot be negative.

C:\1BM23CS157>java Main
Enter Father's age: 56
Enter Son's age: -6
Error: Son's age cannot be negative.

C:\1BM23CS157>
```

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.

GithubLink:https://github.com/KratishPorwal/JAVALABPROGRAMS/tree/main/threads%20prg%208

Algorithm:

LAB 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 DisplayBMS extends thread {
    @ Override
    public void run (){
        while (true ){
            try {
                System. out. println ("BMS college of Engineering")
                thread .sleep(10000);
            }
            catch (InterruptedException e ){
                System. out. Println (e) ;
            }
        }
    }
}
class DisplayCSE extends Thread
    @ Override
    public void run ( ){
        while (true ){
            try {
                System. out. println ("CSE");
                Thread .sleep (2000) ;
            } catch (InterruptedException e){
                System. out. println (e) ;
            }
        }
    }
}
```

```
Public class Main {
    public static void Main (String [] Args) {
        Display BMS thread BMS = new Display BMS();
        Display CSE thread CSE = new Display CSE();

        thread BMS start();
        thread CSE. start();
    }
}
```

output

```
BMS college of Engineering
CSE
CSE
CSE
CSE
CSE
BMS college of Engineering
CSE
CSE
CSE
CSE
CSE
BMS college of Engineering
CSE
CSE
CSE
CSE
CSE
BMS college of Engineering
```

Code:

```
class DisplayBMS extends Thread {

   @Override

   public void run() {

      while (true) {
```

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

class DisplayCSE extends Thread {
  @Override
  public void run() {
    while (true) {
        try {
            System.out.println("CSE");
            Thread.sleep(2000);  // Sleep for 2 seconds
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
  }
}
public class threads {
  public static void main(String[] args) {
    DisplayBMS threadBMS = new DisplayBMS();
```

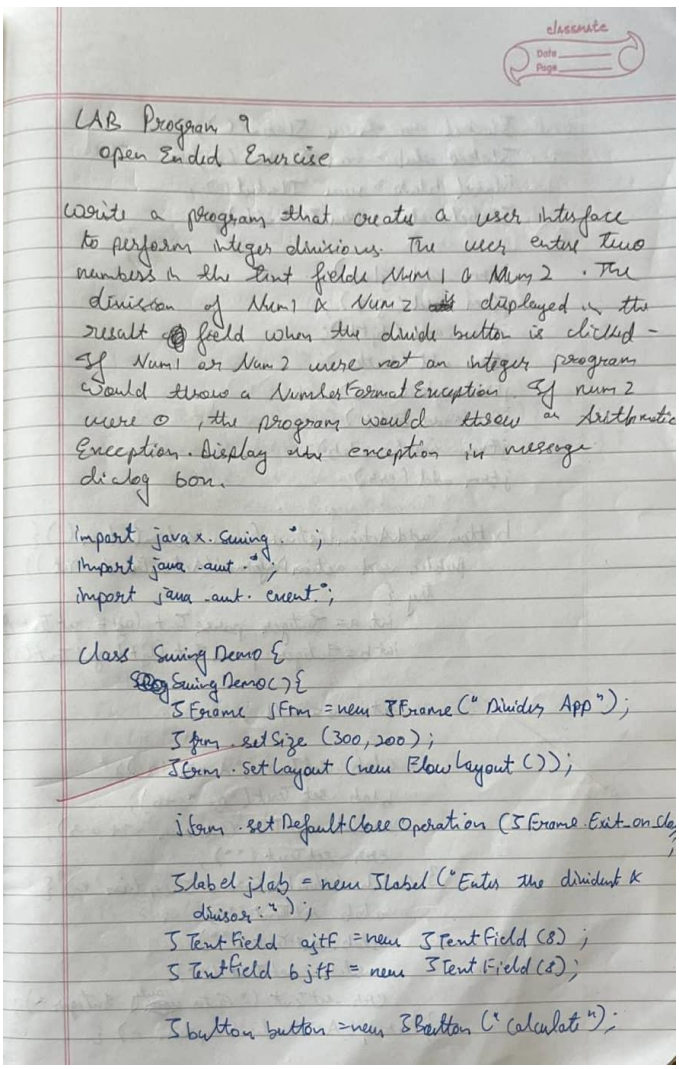
```
        DisplayCSE threadCSE = new DisplayCSE();

        threadBMS.start();

        threadCSE.start();

    }

}
```

```
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
```

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.

GithubLink:https://github.com/KratishPorwal/JAVALABPROGRAMS/tree/main/lab%20prog%209

Algorithm:



LAB Program 9
open Ended Exercise

Write a program that create a user interface to perform integer divisions. The user enter two numbers in the text fields Num1 & Num2. The division of Num1 & Num2 displayed in the result field when the divide button is clicked. If Num1 or Num2 were not an integer program would throw a NumberFormatException. If num2 were 0, the program would throw an Arithmetic Exception. Display the exception in message dialog box.

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

class SwingDemo {
    SwingDemo() {
        JFrame jFrm = new JFrame("Divider App");
        jFrm.setSize(300, 200);
        jFrm.setLayout(new FlowLayout());

        jFrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLO

        JLabel jlab = new JLabel("Enter the dividend &
          divisor:");
        JTextField ajtf = new JTextField(8);
        JTextField bjtf = new JTextField(8);

        JButton button = new JButton("Calculate");
```

```
JLabel err = new JLabel();
JLabel alab = new JLabel();
JLabel blab = new JLabel();
JLabel anslab = new JLabel();

jform.add(jlab);
jform.add(ajtf);
jform.add(bjtf);
jform.add(button);
jform.add(alab);
jform.add(blab);
jform.add(anslab);
jform.add(err);

button.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent evt){
        try {
            int a = Integer.parseInt(ajtf.getText());
            int b = Integer.parseInt(bjtf.getText());

            int ans = a/b;

            alab.setText("A =" +a);
            blab.setText("B =" +b);
            anslab.setText("Answer =" + ans);
            err.setText("");
        } Catch (NumberFormatException e){
            alab.setText("")
            blab.setText("")
            anslab.setText("");
            err.setText("Enter only Integer");
        } catch (ArithmeticException c) {
```

```
            alab.setText("");
            blab.setText("");
            ans lab.setText("");
            err.setText("B should be non zero");
        }
    }
});

JFrame.setVisible(true);
}
Public static void main(String args[]){
    SwingUtilities.invokeLater(new Runnable(){
        public void run(){
            new SwingDemo();
        }
    });

}
}
```

Output:

Enter the dividend & divisor 10 2
calculate
A= 10 B=2 Answer = 5

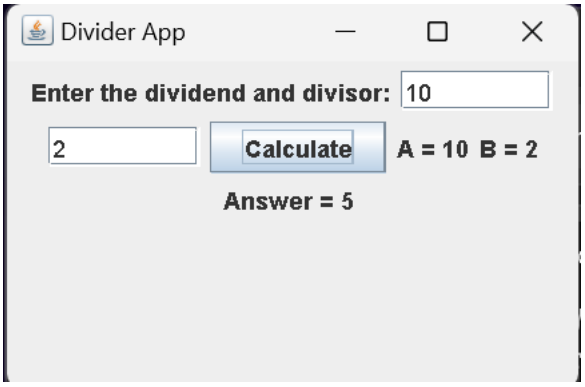
Enter the dividend and divisor 10 0
calculate
B should be non zero.

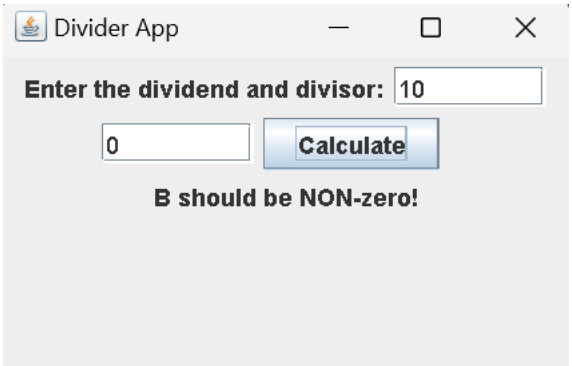
Code:

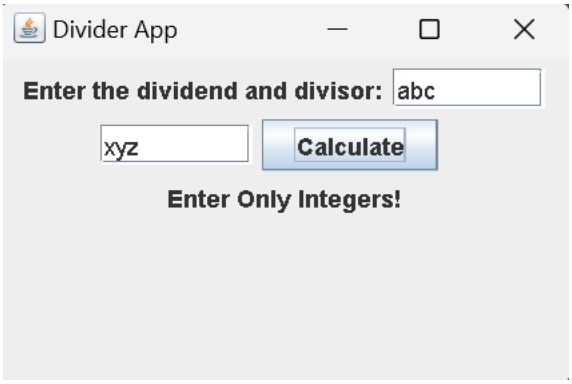
```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;


class SwingDemo {

  SwingDemo() {

    JFrame jfrm = new JFrame("Divider App");

    jfrm.setSize(300, 200);

    jfrm.setLayout(new FlowLayout());


    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


    JLabel jlab = new JLabel("Enter the dividend and divisor:");


    JTextField ajtf = new JTextField(8);

    JTextField bjtf = new JTextField(8);


    JButton button = new JButton("Calculate");


    JLabel err = new JLabel();

    JLabel alab = new JLabel();

    JLabel blab = new JLabel();

    JLabel anslab = new JLabel();


    jfrm.add(jlab);
```

```
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);
jfrm.add(err);

button.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
      try {
         int a = Integer.parseInt(ajtf.getText());
         int b = Integer.parseInt(bjtf.getText());

         int ans = a / b;

         alab.setText("A = " + a);
         blab.setText("B = " + b);
         anslab.setText("Answer = " + ans);
         err.setText("");
      } catch (NumberFormatException e) { alab.setText("");
         blab.setText("");
         anslab.setText("");
         err.setText("Enter Only Integers!");
      } catch (ArithmeticException e) {
         alab.setText("");
```

```
            blab.setText("");
            anslab.setText("");
            err.setText("B should be NON-zero!");
          }
        }
      });
      jfrm.setVisible(true);
    }

    public static void main(String args[]) {
      SwingUtilities.invokeLater(new Runnable() {
        public void run() {
          new SwingDemo();
        }
      });
    }
}
```




Divider App
Enter the dividend and divisor:
10
0
Calculate
B should be NON-zero!


Divider App
Enter the dividend and divisor:
abc
xyz
Calculate
Enter Only Integers!

Program 10: Demonstrate Inter process Communication and deadlock

GithubLink:https://github.com/KratishPorwal/JAVALABPROGRAMS/tree/main/lab%20prog%2010

Algorithm:

LAB Program 10
Demonstrate Inter process Communication and Deadlock

A) Deadlock

```
package lab;

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

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

    Deadlock() {
        Thread.currentThread().setName("Main Thread");
        Thread t = new Thread(this, "Racing Thread");
        t.start();
        a.foo(b);
        System.out.println("Back in main thread");
    }
    @Override
    public void run() {
        b.bar(a);
    }
    public static void main(String[] args) {
        new Deadlock();
    }
}
```

OutPut:

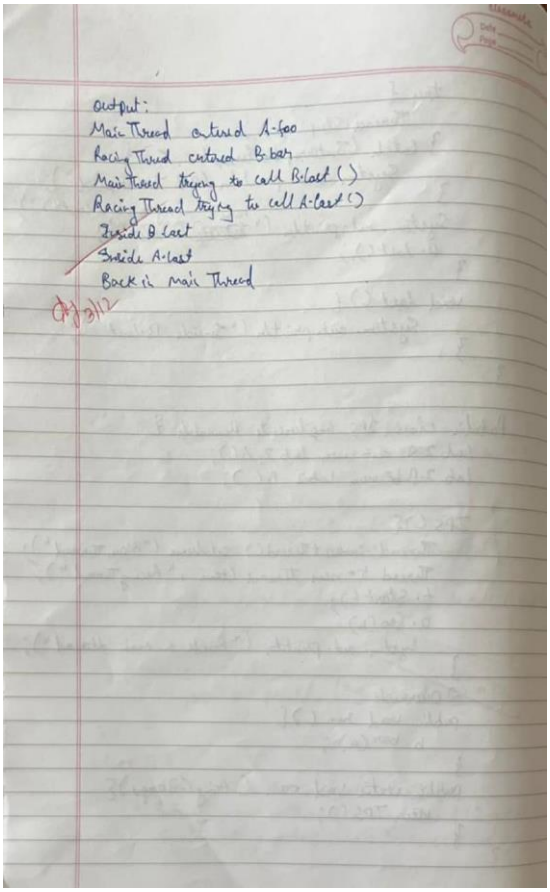
```
Main Thread entered A.foo
Racing Thread entered B.bar
```

Main Thread trying to call B.last()
Racing Thread trying to call A.last()
Inside B.last
Inside A.last
Back in Main Thread

Program 10 B IPC

```
Package lab2;

Class A {
    Synchronized void foo (lab 2.B b) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered A.foo");
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("A Interrupted")
        }
        System.out.println(name + "trying to call B.last()
        b.last();
    }
    void last() {
        System.out.println("Inside A.last");
    }
}
Class B {
    Synchronized void bar (lab 2.A a) {
        String name = Thread.currentThread().getName();
        System.out.println(name + "entered B.bar");
```

```
        try {
            Thread.sleep(1000);
        } Catch (InterruptedException e) {
            System.out.println("B Interrupted");
        }
        System.out.println(name + "trying to call A.last()");
        a.last();
    }

    void last() {
        System.out.println("Inside B.last");
    }
}

Public Class IPS implements Runnable {
    lab 2.A a = new lab 2.A();
    lab 2.B b = new lab2.B();

    IPS() {
        Thread.currentThread().setName("Main Thread");
        Thread t = new thread (this, "Racing Thread");
        t.Start();
        a.foo(b)
        System.out.println("Back in main thread");
    }
    @Override
    public void run() {
        b.bar(a);
    }
    public static void main (String[] args) {
        new IPS();
    }
}
```



output:
Main Thread entered A.foo
Racing Thread entered B.bar
Main Thread trying to call B.last()
Racing Thread trying to call A.last()
Inside B.last
Inside A.last
Back in main Thread

10A Deadlock

Algorithm: 10A Deadlock

Code:

```
package Lab;


class A {

   synchronized void foo(B b) {

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

      System.out.println(name + " entered A.foo");
```

```
        try {
            Thread.sleep(1000);  // This may throw InterruptedException
        } catch (InterruptedException e) {
            System.out.println("A Interrupted");
        }

        System.out.println(name + " trying to call B.last()");
        b.last();
    }

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

class B {
    synchronized void bar(A a) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered B.bar");

        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("B Interrupted");
        }
```

```
        System.out.println(name + " trying to call A.last()");
        a.last();
    }

    void last() {
        System.out.println("Inside B.last");
    }
}

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

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

        a.foo(b);

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

    @Override
    public void run() {
```

```
        b.bar(a);
    }

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

```
MainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call B.last()
RacingThread trying to call A.last()
Inside B.last
Inside A.last
Back in main thread

Process finished with exit code 0
```

Program 10 B: IPS

Algorithm:

Code:

```
package Lab2;

class A {
    synchronized void foo(Lab2.B b) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered A.foo");

        try {
```

```
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("A Interrupted");
        }

        System.out.println(name + " trying to call B.last()");
        b.last();
    }

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

class B {
    synchronized void bar(Lab2.A a) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered B.bar");

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

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

```
        a.last();
    }

    void last() {
        System.out.println("Inside B.last");
    }
}

public class IPS implements Runnable {
    Lab2.A a = new Lab2.A();
    Lab2.B b = new Lab2.B();

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

        a.foo(b);

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

    @Override
    public void run() {
        b.bar(a);
    }
```

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


        new IPS();

    }

}
```

```
MainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call B.last()
RacingThread trying to call A.last()
Inside A.last
Inside B.last
Back in main thread

Process finished with exit code 0
```