B. M. S. COLLEGE OF ENGINEERING

(AUTONOMOUS COLLEGE UNDER VTU, BELGAUM)
BANGALORE – 560019

2022-23

LAB RECORD

OBJECT ORIENTED JAVA PROGRAMMING (23CS3PCOOJ)

Submitted by :

NAME : SHRUTI KHANDELIA

USN : 1BM22CS274

SEMESTER: III

SECTION : E

Submitted to
Dr. Seema Patil
Assistant Professor
Dept. of CSE, BMSCE

my self

Name: Shruti Khandelia

Class: 3rd sem

School / College: BMSCE

Email:

Index

	Date	Page No.	Subject:	Sign	Remark
1	12/12/23		Lab 1		
2	19/12/23		Lab 2		
3	26/12/23		Lab 3		
4	02/01/24		Lab 4		
5	16/01/24		Lab 5		
6	23/01/24		Lab 6		
7	30/01/24		Lab 7		
8	06/01/24		Lab 8		
9	13/01/24		Lab 9		
10	20/01/24		Lab 10		

```
}
else if(d>0)
{
        r1 = ((-b)+(Math.sqrt(d)))/(double)(2*a);
        r2 = ((-b)-(Math.sqrt(d)))/(double)(2*a);
        System.out.println("Roots are real and distinct");
        System.out.println("Root1 = "+r1+"Root2 = "+r2);
}
else if(d<0)
{
        System.out.println("Roots are imaginary");
        r1 = (-b)/(2*a);
        r2 = Math.sqrt(-d)/(2*a);
        System.out.println("Root 1 = "+r1+" +i "+r2);
        System.out.println("Root 2 = "+r1+" -i "+r2);
}
    }
}
class QuadraticMain
{
    public static void main(String args[])
    {
        Quadratic q = new Quadratic();
        q.getd();
        q.compute();
        System.out.println("Shruti Khandelia, 1BM22CS274");
    }
}
```

Page No.
Date

Output 1

Enter the coefficients of a, b, c

1

-2

1

Roots are real and equal

Root 1 = Root 2 = 1.0

Shruti Khandelia : 1BM22CS274

Output 2

Enter the coefficients of a, b, c

3

5

6

Roots are imaginary

Root 1 = 0.0 + i 1.1426091000668406

Root 2 = 0.0 - i 1.1426091000668406

Shruti Khandelia : 1BM22CS274

Output 3

Enter the coefficients of a, b, c

1

6

3

Roots are real and distinct

Root 1 = -0.5505102572168211 Root 2 = -5.449489742783178

Shruti Khandelia : 1BM22CS274

10
8w
12/12/23

Lab-2

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

$$SGPA = \frac{\sum(\text{course credits} \times \text{Gradepoint})}{\sum \text{course credit}}$$

⇒
```
import java.util.Scanner;
class Subject
{
    int subjectMarks;
    int Credits;
    int grade;
};
class Student
{
    Subject subject[];
    String name;
    String USN;
    double SGPA;
    Scanner s;
    Student()
    {
        int i;
        subject = new Subject[9];
        for(i=0; i<8; i++)
        {
            subject[i] = new Subject();
            s = new Scanner(System.in);
        }
    }
```

```
void getStudentDetails()
{
    System.out.print("Enter your name");
    name = s.next();
    System.out.println("Enter USN:");
    USN = s.next();
}

void getMarks()
{
    for(int i=0; i<8; i++)
    {
        System.out.print("Enter marks for subject"+(i+1)+":");
        subject[i].subjectMarks = s.nextInt();
        System.out.print("Enter your credits for subject "+(i+1)+":");
        subject[i].credits = s.nextInt();
        subject[i].grade = (subject[i].subjectMarks/10)+1;
        if(subject[i].grade==11)
        {
            subject[i].grade=10;
        }
        if(subject[i].grade<=4)
        {
            subject[i].grade=0;
        }
    }
}

void computeSGPA()
{
    int effectivescore=0;
    int totalcredits=0;
    for(int i=0; i<8; i++)
    {
```

```
            effectiveScore += (subject[i].grade * subject[i].credits);
            totalCredits += subject[i].credits;
        }
        SGPA = (double) effectiveScore / (double) totalCredits;
    }
}
class Main
{
    public static void main(String args[])
    {
        Student s1 = new Student();
        s1.getStudentDetails();
        s1.getMarks();
        s1.computeSGPA();
        System.out.println("Name:" + s1.name + "USN:" + s1.USN +
                           "SGPA:" + s1.SGPA);
    }
}
```

Output

```
Enter your name Shruti
Enter your USN:
1BM22CS274
Enter marks for subject1: 88
enter your credit for subject1: 4
Enter marks for subject2: 89
enter your credits for subject2: 4
Enter marks for subject3: 92
enter your credits for subject3: 3
Enter marks for subject4: 90
enter your credits for subject4: 3
```

Enter marks for subject 5: 87
enter your credits for subject5: 3
Enter marks for subject 6: 86
Enter your credits subject6: 2
Enter marks for subject 7: 89
Enter your Credits subject7: 2
Enter marks for subject8: 95
enter your credits subject8: 1
Name: Shruti USN: 1BM22CS274 SGPA: 9.318181818181818

10

Viva 12/23

Lab-3

Page No.
Date

Q. Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the value of the member. Include methods to set and get the details of the objects. Include a toString() method that could display the complete details of the book. Develop a Java program to create n book objects

```
import java.util.Scanner;

class 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;
}
```

```
}
class Main
{
    public static void main(String args[])
    {
        Scanner s = new Scanner(System.in);
        int n;
        String name;
        String author;
        int price;
        int numpages;

        n = s.nextInt();
        Books b[];
        b = new Books[n];

        for(int i=0; i<n; i++)
        {
            name = s.next();
            author = s.next();
            price = s.nextInt();
            numPages = s.nextInt();
            b[i] = new Books(name, author, price, numPages);
        }
        for(int i=0; i<n; i++)
        {
            System.out.println(b[i].toString());
        }
        System.out.println("Shruti Khandelia: 1BM22CS274");
    }
}
```

Output:

2
One8: An emotion
Shruti Khandelia
1000
300
King Kohli
Shreyash Sinha
500
250
Book name: One8: An emotion
Author name: Shruti khandelia
Price: 1000
Number of Pages: 300
Bookname: King Kohli
Authorname: Shreyash Sinha
Price: 500
Number of pages: 250
Shrutikhandelia: 1BM22CS274

26/12/23

10

Lab-4

Page No.
Date

Q Develop a Java program to create ~~ab~~ 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 class extends the class Shape. Each one of the class contain methods given above.

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

abstract class Shape extends inputScanner
{
        double a,b;
        abstract void printArea();
        abstract void getInput();
}

class Rectangle extends Shape
{
        void getInput()
        {
                System.out.println("Enter dimensions");
                a=sc.nextDouble();
                b=sc.nextDouble();
        }
        void printArea()
        {
                System.out.println("Area=" +(a*b));
        }
}
```

```
class Triangle extends Shape
{
    void getInput()
    {
        System.out.println("Enter dimension:");
        a = sc.nextDouble();
        b = sc.nextDouble();
    }
    void printArea()
    {
        System.out.println("Area=" +((a*b)/2));
    }
class Circle extends Shape
{
    void getInput()
    {
        System.out.println("Enter dimensions for Circle");
        a = sc.nextDouble();
    }
    void printArea()
    {
        System.out.println("Area=" +(3.14*a*a));
    }
}

public class Main
{
    public static void main(String[] args)
    {
        Rectangle r = new Rectangle();
        r.getInput();
        r.printArea();
```

```
Triangle t = new Triangle();
t.getInput();
t.printArea();

Circle c = new Circle();
c.getInput();
c.printArea();

System.out.println("Shruti Khandelia: 1BM22CS274");
}
}
```

Output

```
Enter dimension
4
5
Area = 20.0
Enter dimension
4
3
Area = 6.0
Enter dimensions for circle
3
Area = 28.2599999
Shruti Khandelia: 1BM22CS274
```

02/01/24

Lab-5

Page No.
Date

Q Develop a java program to create a class Bank that maintains two kinds of account for its customers, one saving other current. The saving account provides compound interest and withdrawal facilities but no cheque. Current account holders should also maintain a minimum balance, and if the balance falls below this level, a service charge is imposed

```
import java.util.Scanner;

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

    Account(String customerName, int accountNumber, String
    accountType){
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = 0;
    }
    void deposit(double amount)
    {
        balance += amount;
        System.out.println("Deposit successful.
        Update balance:" + balance);
    }

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

void withdraw(double amount)
{ }

}

class CurAcct extends Account{
double minBalance, ServiceCharge;

CurAcct(String customerName, int accountNumber)
{
super(customerName, accountNumber, "Current");
this.minBalance = 1000;
this.serviceCharge = 50;
}

@Override
void deposit(double amount)
{
super.deposit(amount);
checkMinimumBalance();
}

private void checkMinimumBalance()
{
if (balance < minBdano)
{
balance -= service charge;
System.out.println("Service charge
imposed. Updated balance:" + balance);
}
}

class SavAcct extends Account{
double interestRate;

```
SavAcct(String customerName, int accountNumber)
{
    super(customerName, accountNumber, "Saving");
    this.interestRate = 0.05;
}
@Override
void deposit(double amount)
{
    super.deposit(amount);
    computeInterest();
}
void computeInterest() {
    double interest = balance * InterestRate;
    balance += interest;
    System.out.println("Interest computed
    and deposited. Updated balance:" + balance);
}
void withdraw(double amount)
{
    if (amount <= balance)
    {
        balance -= amount;
        System.out.println("Withdrawal successful
        .Update balance:" + balance);
    }
    else
        System.out.println("Insufficient funds for
        withdrawal");
    }
}
public class Main{
```

```
public static void main (String [] args)
{
    Scanner sc = new Scanner (System.in);
    int ch;
    Account userAccount = null;
    System.out.println(" Enter customer name:");
    String customerName = sc.next();
    System.out.println(" Enter account number:");
    String accountType = sc.next ();
    if (accountType.equals ("saving")){
        userAccount = New SavAcct (customerName,
        accountNumber);
}
    else if (accountType.equals ("Current"))
{
        UserAccount = New CurAcct (customerName,
        accountNumber);
}
    else{
        System.out.println("Invalid account type.
        Existing program");
        sc.close();
        System.exit(0);
}
while (true){
        System.out.println("---MENU---\n
Deposit\n2. Withdraw\n3. Display Account
Details\n4. Exit");
System.out.println("Enter your choice:");
ch = sc.nextInt();

switch (ch) {
```

```
case 1: System.out.println("Enter dep. amt");
        double depositAmount = sc.nextDouble();
        userAccount.deposit(depositAmount);
        break;

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

case 3: System.out.println("Customer name:" + userAccount.
        customerName);
        System.out.println("Account Number:" +
        userAccount.accountNumber);
        System.out.println("Type of Account:"
        + userAccount.accountType);
        System.out.println("Balance:" + userAccount
        .balance);
        break;

case 4: System.out.println("Existing program.");
        sc.close();
        System.exit(0);

default: System.out.println("Invalid choice.
         Please enter a valid option");
    }
  }
}
```

Output

Enter customer name: Shruthi
Enter account number: 1
Enter the type of account: saving
---MENU---
1. Deposit
2. Withdraw
3. Display Account Details
4. Exit
Enter your choice: 1
Enter deposit amount: 1000
Deposit successful
Interest computed and deposited: 1050
Enter your choice: 2
Enter withdrawal amount: 500
Withdrawal successful Balance: 550
Enter your choice: 3
Customer name: Shruti
Account number: 2
Type of account: Savings
Balance: 550

LAB 1

```
import java.util.Scanner;
class Quadratic
{
    int a,b,c;
    double r1,r2,d;
    void getd()
    {
        Scanner s=new Scanner(System.in);
        System.out.println("Enter the coefficients of a,b,c");
        a=s.nextInt();
        b=s.nextInt();
        c=s.nextInt();
    }
    void compute()
    {
        while(a==0)
        {
            System.out.println("Not a quadratic equation");
            System.out.println("Enter a non zero value for a:");
            Scanner s=new Scanner(System.in);
            a=s.nextInt();
        }
    }
    d=b*b-4*a*c;
    if(d==0)
    {
        r1=(-b)/(2*a);
        System.out.println("Roots are real and equal");
        System.out.println("Root1=Root2=" +r1);
```

Q. Write a Java program to create a generic class Stack which hold 5 integers and 5 double values

```
                    ↳ type
class GenericStack<T>{
        private Object[] stackArray;
        private int top=-1;
        private static final int MAX_SIZE=5;

public GenericStack(){
        stackArray=new Object[MAX_SIZE];
}

public void push(T value){
        if(top<MAX_SIZE-1)
            stackArray[++top]=value;
        else
            System.out.println("StackFull");
}
@SuppressWarnings("unchecked")
public T pop(){
return ↙ if(top>=0)
 may      return(T)stackArray[top--];
 type T
        else{
            System.out.println("Empty stack");
            return null;
        }
}
public boolean isEmpty(){
        return top==-1;
}
```

```
public boolean isFull(){
{
    return top == MAX_SIZE - 1;
}
class Main{
    public static void main(String[] args)
    {
        GenericStack<Integer> integerStack = new GenericStack();
        GenericStack<Double> doubleStack = new GenericStack();

        for(int i = 1; i <= 5; i++)
        {
            integerStack.push(i);
        }
        for(double i = 1.0; i <= 5.0; i++) {
            doubleStack.push(i);
        }
        System.out.println("Popped integers:");
        while(!integerStack.isEmpty()){
            System.out.println(integerStack.pop());
        }

        System.out.println("Popped doubles:");
        while(!doubleStack.isEmpty())
        {
            System.out.println(doubleStack.pop());
        }
    }
}
```

Output

Popped Integers from the stack:

5

4

3

2

1

Popped doubles:

5.0

4.0

3.0

2.0

1.0

Q. Write a Java program to create an abstract class Bird with abs. Methods fly() and makesound(). Create subclasses Eagle and Hawk that extend the Bird class and implement the respective methods to describe how each bird flies and makes a sound.

```
abstract class Bird
        public abstract void fly();
        public abstract void makesound();
}

class Eagle extends Bird{
        @override
        public void fly(){
                System.out.println("Eagle soars high");
        }
        @override
        public void makesound(){
                System.out.println("Eagle screeches loudly");
        }
}
class Hawk extends Bird{
        @Override
        public void fly(){
                System.out.println("Hawk glides gracefully");
        }
        @override
        public void makesound(){
                System.out.println("Hawk emits a piercing
        }
}
public class BirdDemo{
        public static void main(String[] args){
```

```
Eagle eagle = new Eagle();
Hawk hawk = new Hawk();

System.out.println("Eagle:");
eagle.fly();
eagle.makeSound();

System.out.println("\nHawk:");
hawk.fly();
hawk.makeSound();
    }
}
```

Output

Eagle:
Eagle soars high.
Eagle screeches loudly

Hawk:
Hawk glides gracefully
Hawk emits a piercing cry

14) Write a Java program to demonstrate concat() for s1 = "Hello" and s2 = "world"

```
public class ConcatenationDemo
    public static void main(String[] args)
    {
        String s1 = "Hello";
        String s2 = "world";

        String result = s1.concat(s2);
        System.out.println("Concatenated String: " + result);
    }
}
```

Output

Concatenated String: HelloWorld

15) Write a Java program to demonstrate trim() for "Hello friends".

```
public class Trimsf
    public static void main (String [] args)
    {
        String new = "Hello friends";
        String trimmed = new trim();
        System.out.println("Old String:" + new);
        System.out.println("New string:" + trim);
    }
}
```

Output

Old string: Hello friends

New string: HelloFriends

16/01/24

Lab-6

Page No.
Date

Q- Create a package CIE which has two classes Student and Internals. The class Student has members like usn, name, sem. The class Internals derived from Student has an array that stores the internal marks scored in five courses of 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 course 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;
import java.util.Scanner;

public class Student{
    protected String usn = new String();
    protected String name = new String();
    protected int sem;

    public void inputStudentDetails(){
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter USN: ");
        usn = scanner.next();
        System.out.print("Enter name: ");
        name = scanner.next();
        System.out.print("Enter semester: ");
        sem = scanner.nextInt();
    }

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

2

```
// Internals.java
package CIE;
import java.util.Scanner;

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

    public void inputCIEmarks() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter Internal marks for " + name);
        for(int i=0; i<5; i++) {
            System.out.print("Subject" + (i+1) + "
                marks: ");
            marks[i] = scanner.nextInt();
        }
    }
}
```

```
// Externals.java
package SEE;
import CIE.Internals;
import java.util.Scanner;

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

    public External() {
```

```
        marks = new int [5];
        finalMarks = new int [5];
    }

    pubiic void inputSEEmarks(){
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter SEE marks for" + name);
        for (int i=0; i<5; i++){
            System.out.print("Subject " + (i+1) + " marks:");
            marks[i] = scanner.nextInt();
        }
    }

    public void calculateFinalMarks(){
        for(int i=0; i<5; i++)
            finalMarks[i] = marks[i]/2 + sum Marks[i];
    }
    display
    public void calculateFinalMarks(){
        for(int i=0; i<5; i++)
            System.out.println("Subject" + (i+1) + ":"
            + finalMarks[i]);
    }
}
```

```
//Main.java
import SEE.Externals;

public class Main{
    public static void main(String args[]){
        int numofStudents=2;
        Externals finalMarks[] = new Externals[numofStudents];

        for(int i=0; i<numofStudents; i++)
        {
            finalMarks[i] = new Externals();
            finalMarks[i].inputStudentDetails();
            System.out.println("\n Enter CIE marks");
```

```
                finalMarks[i].inputCIEMarks();
                System.out.println("Enter SEE marks");
                finalMarks[i].inputSEEMarks();
            }
            System.out.println("Displaying data:\n");
            for(int i=0; i<numofStudents; i++){
                finalMarks[i].calculateFinalMarks();
                finalMarks[i].displayFinalMarks();
            }
        }
    }
```

Output

```
Enter USN: 1BM22CS274
Enter name: Shruti B
Enter semester: 3
Enter CIE marks
Enter Internal marks for shruti
Subject 1 marks: 40
Subject 2 marks: 35
Subject 3 marks: 42
Subject 4 marks: 38
Subject 5 marks: 45
Enter USN: 1BM22CS273
Enter name: Shreyash
Enter semester: 5
Enter CIE marks
Enter Internal marks for Shreyash
Subject 1 marks: 36
Subject 2 marks: 48
Enter SEE marks
Enter SEE marks for shruti
Subject1Marks: 75
Subject 2 " : 80
Subject 3 " : 65
" 4 " : 88
" 5 " : 85
```

subject 3 marks : 50
subject 4 marks : 42
subject 5 marks : 55
Enter SEE marks
Enter SEE marks for shreyash
subject 1 marks : 80
subject 2 marks : 72
subject 3 marks : 85
subject 4 marks : 78
subject 5 marks : 96
Displaying data:
USN: 1BM22CS274
Name: Shruti
semester: 3
subject 1 : 97
subject 2 : 95
subject 3 : 76
subject 4 : 78
subject 5 : 87
USN: 1BM22CS274
Name: Shreyash
semester: 3
subject 1 : 98
subject 2 : 84
subject 3 : 92
subject 4 : 81
subject 5 : 100

30.01.24

Lab-7

Page No.
Date

Q Write a program that demonstrate handling of exception in inheritance tree. Create base class father and derived class son who extends father. In father class, implement a constructor that takes both father and son's age and throws exception if son's age >= father's age.

```
import java.util.Scanner;

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

class InputScanner{
        Scanner s=new Scanner(System.in);
        public int nextInt(){
            return s.nextInt();
        }
}

class Father extends InputScanner{
        int fatherAge;
        public Father() throws WrongAge{
            System.out.println("Enter father's age:");
            fatherage=nextInt();
            if (fatherAge<0){
                throw new WrongAge("Age can't be negativ
            }
        }
        public void display(){
            System.out.println("Father's age:" + fatherAge);
        }
```

```
}
class Son extends Father{
    int sonAge;
    public Son() throws WrongAge{
        super();
        System.out.println("Enter son's age:");
        sonAge = nextInt();
        if (sonAge >= FatherAge){
            throw new WrongAge("Son's Age can't be
            greater than father's Age");
        }
        else if (sonAge < 0){
            throw new WrongAge("Age can't be negative");
        }
    }
}

public void display(){
    super.display();
    System.out.println("Son's age:" + sonage);
}
}

public class Main{
    public static void main(String[] args){
        try{
            Son son = new Son();
            son.display();
        } catch (WrongAge e){
            System.out.println("Error:" + e.getMessage
        }
    }
}
```

Output

Enter father's age : 23
Enter son's age : 29
Error: son's age can't be greater than father's age

30.01.24

Lab8

1) 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 DisplayThread extends Thread{
    String message;
    int interval; // in milliseconds
    public DisplayThread(String message, int interval){
        this.message = message;
        this.interval = interval;
    }
    @Override
    public void run(){
        while(true){
            System.out.println(message);
            try{
                Thread.sleep(interval);
            }catch(InterruptException e){
                e.printStackTrace();
            }
        }
    }
}

public class Main{
    public static void main(String[] args){
        DisplayThread thread1 = new DisplayThread("BMS College of Engineering", 10000);
        DisplayThread thread2 = new DisplayThread("CSE", 2000);
    }
}
```

```
        thread1.start();
        (thread).start();
    }
}
```

Output

BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE

Lab-10

Q. Demonstrate InterProcess Communication and deadlock.

1)
```
class Q{
    int n;
    boolean valueSet = false;
    synchronized int get(){
        while(!valueSet)
        try{
            System.out.println("\n Consumer waiting\n");
            wait();
        }
        catch(InterruptedException e){
            System.out.println("Interrupted Exception caught");
        }
        System.out.println("Got:" + n);
        valueSet = false;
        System.out.println("\n Intimate Producer\n");
        notify();
        return n;
    }
    synchronized void put(int n){
        while(valueSet)
            try{
                System.out.println("\n Producer waiting\n");
                wait();
            }
            catch(InterruptedException e){
                System.out.println("Interrupted
```

```
        this.n = n;
        valueSet = true;
        System.out.println("Put:" + n);
        System.out.println("\nIntimate Consumer\n");
        notify();
    }
}

class Producer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }

    public void run() {
        int i=0;
        while (i<5) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;
    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }

    public void run() {
        int i=0;
        while (i<5) {
```

```
            int r = q.get();
            System.out.println("Consumed:" + r);
            i++;
        }
    }
}

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

Output

```
Press Control-C to stop.
Put: 0
Intimate Consumer
Producer waiting
Got: 0
Intimate Producer
Consumed: 0
Consumer waiting
Put: 1
Intimate Consumer
Producer waiting
Got: 1
Intimate Producer
Consumed: 1
```

put=2
Intimate Consumer
Producer waiting
Got=2
Intimate Producer
consumed=2
Put=3
Intimate Consumer
Producer Waiting
Got=3
Intimate Producer
consumes=3
Put=4
Intimate Consumer
Got=4
Intimate Producer
consumes=4

Q) class A {

synchronized void foo(B b) {

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

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

try {

Thread.sleep(1000);

}

catch (Exception e) {

System.out.println("A interrupted");

}

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

b.last();

}

}

void last() {

System.out.println("Inside A.last");

}

}

class B {

synchronized void bar(A a) {

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

System.out.println(name + "entered B.bar");

try {

Thread.sleep(1000);

}

catch (Exception e) {

System.out.println("B Interrupted");

}

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

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

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

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

        a.foo(b);
        System.out.println("Back in main thread");
    }
    public void run(){
        b.bar(a);
        System.out.println("Back in other thread");
    }
}

class Main{
    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()
Inside A.last
Racing Thread trying to call A.last()
Inside A.last
Back in other thread
Back in main thread

13.08.24

Lab-9

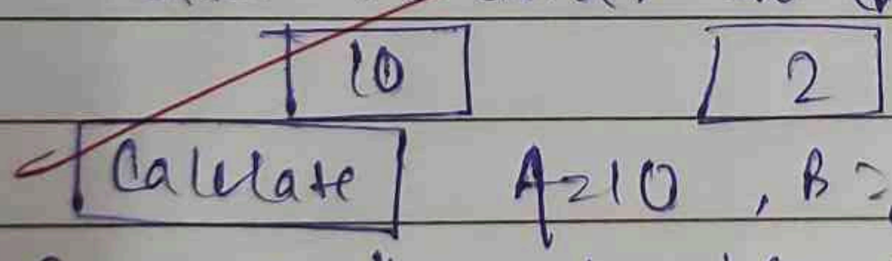
Q Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide Button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an ArithmeticException, 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(275, 150);
        jfrm.setLayout(new FlowLayout());
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel jlab = new JLabel("Enter the divider and dividend");
    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(err);
    jfrm.add(jlab);
    jfrm.add(ajtf);
    jfrm.add(bjtf);
```

```
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(ansLab);

ActionListener l=new ActionListener(){
        public void actionPerformed(ActionEvent ev){
                System.out.println("Action event
from a text field");
        }
};

aj.tf.addActionListener(l);
bj.tf.add ActionListener(l);

button.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent evt){
try{
        int a=Integer.parseInt(aj.tf.getText());
        int b=Integer.parseInt(bj.tf.getText());
        int ans=a/b;

        alab.setText("\nA="+a);
        blab.setText("\nB="+b);
        ansLab.setText("\nAns="+ans);
}
catch(NumberFormatException e){
        alab.setText("");
        blab.setText("");
        ansLab.setText("");
        err.setText("Enter only Integers");
}
```

```
catch(ArithmeticException e){
        alab.setText(" ");
        blab.setText(" ");
        anslab.setText(" ");
        err.setText("B should be Non-Zero!");
    }
}
});
jfrm.setVisible(true);
}

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

Output



JFrame → Represents the main window of the application.

JLabel → Used to display text labels on the GUI

Open

jfrm.setSize() → It sets the size of the JFrame. Parameters passed here specifies the width and height of the JFrame.

jfrm.setLayout() → It sets the layout manager for the JFrame. It maintains a natural flow of components based on the order they are added to the container

jfrm.setDefaultCloseOperation() → It is a method used to specify what should happen when the user closes the window

jfrm.add(); → It is used to add various swing components to the JFrame

alab.setText(); → It is used to set the text content of a JLabel component.

jfrm.setVisible(true); → It ~~make~~ is used to show or hide the frame.

SwingUtilities.invokeLater() → It is a method using in swing application to ensure that GUI related tasks are executed on EDT, which is responsible for handling user interface events and updates

20.02.24

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 java.util.Scanner;


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


   void getd() {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter the coefficients of a,b,c");
      a = s.nextInt();
      b = s.nextInt();
      c = s.nextInt();
   }


   void compute() {
      while (a == 0) {
         System.out.println("Not a quadratic equation");
         System.out.println("Enter a non zero value for a:");
         Scanner s = new Scanner(System.in);
         a = s.nextInt();
      }
      d = b * b - 4 * a * c;
      if (d == 0) {


         r1 = (-b) / (2 * a);
         System.out.println("Roots are real and equal");
         System.out.println("Root1 = Root2 = " + r1);
      } else if (d > 0) {
         r1 = ((-b) + (Math.sqrt(d))) / (double) (2 * a);
```

```
            r2 = ((-b) - (Math.sqrt(d))) / (double) (2 * a);
            System.out.println("Roots are real and distinct");
            System.out.println("Roo1 = " + r1 + " Root2 = " + r2);
        } else if (d < 0) {
            System.out.println("Roots are imaginary");
            r1 = (-b) / (2 * a);
            r2 = Math.sqrt(-d) / (2 * a);
            System.out.println("Root1 = " + r1 + " + i" + r2);
            System.out.println("Root1 = " + r1 + " - i" + r2);
        }

    }
}

class QuadraticMain {
    public static void main(String args[]) {
        Quadratic q = new Quadratic();
        q.getd();

        q.compute();
        System.out.println("Shruti Khandelia  1 B M 2 2 C S 2 7 4 ");
    }
}
```

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.*;


class Subject {
   int subjectMarks;
   int credits;
   int grade;
}


class Student {
   Subject subject[];
   String name;
   String usn;
   double sgpa;
   Scanner sc;


   Student() {
      int i;
      subject = new Subject[8];
      sc = new Scanner(System.in);
      for (i = 0; i < 8; i++)
         subject[i] = new Subject();
   }


   void getstudentdetails() {
      System.out.println("Enter your name:");
      this.name = sc.next();
      System.out.println("Enter your USN:");
      this.usn = sc.next();
   }
```

```
    void getMarks() {
        for (int i = 0; i < 8; i++) {
            System.out.println("Enter marks for subject " + (i + 1) + ":");
            subject[i].subjectMarks = sc.nextInt();
            System.out.println("Enter credits for subject" + (i + 1) + ":");
            subject[i].credits = sc.nextInt();
            subject[i].grade = subject[i].subjectMarks / 10 + 1;
            if (subject[i].grade == 11)
                subject[i].grade = 10;
            if (subject[i].grade <= 4)
                subject[i].grade = 0;
        }
    }

    void computeSGPA() {
        int effectiveScore = 0;
        int totalCredits = 0;
        for (int i = 0; i < 8; i++) {
            effectiveScore += (subject[i].grade * subject[i].credits);
            totalCredits += subject[i].credits;
        }
        sgpa = (double) effectiveScore / (double) totalCredits;
    }

}

class Main {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.getstudentdetails();
```

```
        s1.getMarks();
        s1.computeSGPA();
        System.out.println("Name=" + s1.name);
        System.out.println("USN:" + s1.usn);
        System.out.println("SGPA=" + s1.sgpa);
    }
}
```

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.

```
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 class Mainbook
{
	public static void main(String args[])
	{
		Scanner s=new Scanner(System.in);
		int n;
		int i;
		String name;
		String author;
		int price;
		int numPages;

		System.out.println("Enter the number of books:");
		n=s.nextInt();

		Books b[];
		b=new Books[n];

		for(i=0;i<n;i++)
		{
			System.out.println("Enter the details of book" + (i+1) + ":");
			System.out.println("Enter the name of the book:");
			name=s.next();
			System.out.println("Enter the author name:");
			author=s.next();
			System.out.println("Enter the price:");
			price=s.nextInt();
			System.out.println("Enter the number of pages:");
			numPages=s.nextInt();
```

```
                b[i]=new  Books(name,author,price,numPages);
            }

            System.out.println("Book Details:");
            for(i=0;i<n;i++)
            {
                System.out.println(b[i]);
            }
        }
}
```

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

```
import java.util.*;
import  java.math.*;

class  InputScanner  {

   Scanner sc;

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

}

abstract class Shape extends InputScanner {

   double a;

   double b;

   abstract void getInput();

   abstract  void  displayArea();

}

class Rectangle extends Shape {
   void getInput() {
```

```
        System.out.println("Enter the length and breadth:");
        a = sc.nextDouble();
        b = sc.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of rectangle is :" + (a * b));
    }
}

class Triangle extends Shape {
    void getInput() {
        System.out.println("Enter the length and height:");
        a = sc.nextDouble();
        b = sc.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of triangle is :" + (a * b * 0.5));
    }
}

class Circle extends Shape {
    void getInput() {
        System.out.println("Enter the radius:");
        a = sc.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of circle is :" + (a * a * Math.PI));
    }
```

```
}


class ShapeMain {
    public static void main(String[] args) {
        Rectangle r = new Rectangle();
        Triangle t = new Triangle();
        Circle c = new Circle();
        r.getInput();
        r.displayArea();
        t.getInput();
        t.displayArea();
        c.getInput();
        c.displayArea();
    }
}
```

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.**

```
import java.util.Scanner;

class account {
   String name;
   int accno;
   String type;
   double balance;

   account(String name, int accno, String type, double balance) {
      this.name = name;
      this.accno = accno;
      this.type = type;
      this.balance = balance;
   }

   void deposit(double amount) {
```

```
        balance += amount;
    }

    void withdraw(double amount) {
        if ((balance - amount) >= 0) {
            balance -= amount;
        } else {
            System.out.println("Insufficient  balance,cant  withdraw");
        }
    }

    void display() {
        System.out.println("Name:" + name + "\nAccno:" + accno + "\nType:" + type + "\nBalance:" +
balance);
    }
}

class savAcct extends account {

    private static double rate = 5;

    savAcct(String name, int accno, double balance) {
        super(name, accno, "Savings", balance);

    }

    void interest() {
        balance += balance * (rate) / 100;
        System.out.println("Balance:"  +  balance);
    }
```

```
}

class curAcct extends account {

   private double minBal = 500;
   private double serviceCharges = 50;

   curAcct(String name, int accno, double balance) {
      super(name, accno, "Current", balance);

   }

   void checkmin() {

      if (balance < minBal) {
         System.out.println("Balance is less than min balance,service charges imposed:" +
serviceCharges);
         balance -= serviceCharges;
         System.out.println("Balance is:" + balance);
      }

   }

}

class Bank {
   public static void main(String a[]) {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter the name,type(current/savings),account number,initial balance:");
      String name = s.next();
      String type = s.next();
```

```
int accno = s.nextInt();
double balance = s.nextDouble();
int ch;
double amount1, amount2;
account acc = new account(name, accno, type, balance);
savAcct sa = new savAcct(name, accno, balance);
curAcct ca = new curAcct(name, accno, balance);
while (true) {
   if (acc.type.equals("savings")) {
      System.out.println("\nMenu\n1.deposit 2.withdraw 3.compute interest 4.display");
      System.out.println("Enter the choice:");
      ch = s.nextInt();
      switch (ch) {
         case 1:
            System.out.println("Enter the amount:");
            amount1 = s.nextInt();
            sa.deposit(amount1);
            break;
         case 2:
            System.out.println("Enter the amount:");
            amount2 = s.nextInt();
            sa.withdraw(amount2);
            break;
         case 3:
            sa.interest();
            break;
         case 4:
            sa.display();
            break;
         case 5:
            System.exit(0);
```

```
        default:
            System.out.println("invalid input");
            break;
    }
} else {
    System.out.println("\nMenu\n1.deposit 2.withdraw 3.display");
    System.out.println("Enter the choice:");
    ch = s.nextInt();
    switch (ch) {
        case 1:
            System.out.println("Enter the amount:");
            amount1 = s.nextInt();
            ca.deposit(amount1);
            break;
        case 2:
            System.out.println("Enter the amount:");
            amount2 = s.nextInt();
            ca.withdraw(amount2);
            ca.checkmin();
            break;

        case 3:
            ca.display();
            break;
        case 4:
            System.exit(0);
        default:
            System.out.println("Invalid input");
            break;
    }
}
```

```
    }


  }
}
```

GENERICS

```
class GenericStack<T> {
    private Object[] stackArray;
    private int top = -1;
    private static final int MAX_SIZE = 5;

    public GenericStack() {
        stackArray = new Object[MAX_SIZE];
    }

    public void push(T value) {
        if (top < MAX_SIZE - 1) stackArray[++top] = value;
        else System.out.println("Stack is full. Cannot push more elements.");
    }

    @SuppressWarnings("unchecked")
    public T pop() {
        if (top >= 0)
            return (T) stackArray[top--];
        else {
            System.out.println("Stack is empty. Cannot pop more elements.");
            return null;
        }
    }

    public boolean isEmpty() {
        return top == -1;
    }

    public boolean isFull() {
        return top == MAX_SIZE - 1;
```

```
    }
}
class Main{
public static void main(String[] args) {
        GenericStack<Integer> integerStack = new GenericStack<>();
        GenericStack<Double> doubleStack = new GenericStack<>();

        // Push integers to the integer stack
        for (int i = 1; i <= 5; i++) {
            integerStack.push(i);
        }

        // Push doubles to the double stack
        for (double i = 1.0; i <= 5.0; i++) {
            doubleStack.push(i);
        }

        // Pop and print integers from the integer stack
        System.out.println("Popped integers from the stack:");
        while (!integerStack.isEmpty()) {
            System.out.println(integerStack.pop());
        }

        // Pop and print doubles from the double stack
        System.out.println("Popped doubles from the stack:");
        while (!doubleStack.isEmpty()) {
            System.out.println(doubleStack.pop());
        }
    }
}
```

Abstract class prog

```
import java.lang.Math;

abstract class Shape {
   double a;
   double b;
   double c;

   abstract void calculateArea();

   abstract void calculatePerimeter();
}

class Triangle extends Shape {
   Triangle(double x, double y, double z) {
      a = x;
      b = y;
      c = z;
   }

   void calculateArea() {
      double s = (a + b + c) / 2;
      System.out.println("Area=" + (Math.sqrt(s * (s - a) * (s - b) * (s - c))));
   }

   void calculatePerimeter() {
      System.out.println("Perimeter=" + (a + b + c));
   }
}

class Circle extends Shape {
```

```
    Circle(double r) {
        a = r;
    }

    void calculateArea() {
        System.out.println("Area=" + (Math.PI * a * a));
    }

    void calculatePerimeter() {
        System.out.println("Perimeter=" + (2 * Math.PI * a));
    }
}

class ShapeM {
    public static void main(String[] args) {
        Triangle t = new Triangle(2.0, 3.0, 5.0);
        Circle c = new Circle(5.0);
        t.calculateArea();
        t.calculatePerimeter();
        c.calculateArea();
        c.calculatePerimeter();
    }
}
```

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

import java.util.Scanner;

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

   public void inputCIEmarks() {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter Internal Marks for " + name);
      for (int i = 0; i < 5; i++) {
         System.out.print("Subject " + (i + 1) + " marks: ");
         marks[i] = s.nextInt();
      }
   }
}
package CIE;
import java.util.Scanner;

public class Student {
   protected String usn = new String();
```

```
    protected String name = new String();
    protected int sem;

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

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

import CIE.Internals;

import  java.util.Scanner;

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

    public Externals() {
        marks = new int[5];
```

```
        finalMarks = new int[5];
    }

    public void inputSEEmarks() {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter SEE Marks for " + name);
        for (int i = 0; i < 5; i++) {
            System.out.print("Subject " + (i + 1) + " marks: ");
            marks[i] = s.nextInt();
        }
    }

    public void calculateFinalMarks() {
        for (int i = 0; i < 5; i++)
            finalMarks[i] = marks[i] / 2 + super.marks[i];
    }

    public void displayFinalMarks() {
        displayStudentDetails();
        for (int i = 0; i < 5; i++)
            System.out.println("Subject " + (i + 1) + ": " + finalMarks[i]);
    }
}
import  SEE.Externals;

public class Pkgmain {
    public static void main(String args[]) {
        int numOfStudents = 2;
        Externals finalMarks[] = new Externals[numOfStudents];

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

```
            finalMarks[i] = new Externals();
            finalMarks[i].inputStudentDetails();
            System.out.println("Enter CIE marks");
            finalMarks[i].inputCIEmarks();
            System.out.println("Enter  SEE  marks");
            finalMarks[i].inputSEEmarks();
        }

        System.out.println("Displaying data:\n");

        for (int i = 0; i < numOfStudents; i++) {
            finalMarks[i].calculateFinalMarks();
            finalMarks[i].displayFinalMarks();
        }
    }
}
```

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 cases both father and son's age and throws an exception if son's age is

>=father's age.

```
import java.util.Scanner;


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

class InputScanner
{
 protected Scanner s;
 public InputScanner()
 {
  s = new Scanner(System.in);
 }
}



class Father extends InputScanner
{
 protected int fatherAge;
 public Father() throws WrongAge
```

```
{
 System.out.println("Enter  Father's  Age:");
 fatherAge=s.nextInt();

 if(fatherAge<0)
 {
  throw new WrongAge("Age cannot be negetive:");
 }
}

public void display()
{
 System.out.println("Father's Age:" + fatherAge);
}

}


class Son extends Father
{
 private int sonAge;

 public Son() throws WrongAge
 {
  super();
  System.out.println("Enter  Son's  age:");
  sonAge=s.nextInt();


  if(sonAge>fatherAge)
  {
```

```
   throw new WrongAge("Son's age cannot be greater than father's age");
  }
  else if (sonAge<0)
  {
   throw new WrongAge("Age cannot be negative");
  }
 }

 public void display()
 {
  super.display();
  System.out.println("Son's Age: " + sonAge);
 }

}



public class FatherSonAge
{
 public static void main(String args[])
 {
  try
  {
   Son son=new Son();
   son.display();
  }

  catch (WrongAge e)
  {
   System.out.println("Error: " + e.getMessage());
  }
```

```
    }

}
```

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 DisplayMessageThread extends Thread {
   private final String message;
   private final long interval;

   DisplayMessageThread(String message, long interval) {
      this.message = message;
      this.interval = interval;
   }

   public void run() {
      try {
         while (true) {
            System.out.println(message);
            Thread.sleep(interval);
         }
      } catch (InterruptedException e) {
         System.out.println(Thread.currentThread().getName() + " interrupted.");
      }
   }
}

public class TwoThreadDemo {
   public static void main(String[] args) {
      DisplayMessageThread thread1 = new DisplayMessageThread("BMS College of Engineering",
10000);
      DisplayMessageThread thread2 = new DisplayMessageThread("CSE", 2000);

      thread1.setName("Thread  1");
      thread2.setName("Thread  2");
```

```
        thread1.start();
        thread2.start();

        try {
            Thread.sleep(30000);
        } catch (InterruptedException e) {
            System.out.println("Main  thread  interrupted.");
        }

        thread1.interrupt();
        thread2.interrupt();

        System.out.println("Main thread exiting.");
    }
}
```

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

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

class SwingDemo {
  SwingDemo() {
    // create jframe container
    JFrame jfrm = new JFrame("Divider App");
    jfrm.setSize(275, 150);
    jfrm.setLayout(new FlowLayout());
    // to terminate on close
    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // text label
    JLabel jlab = new JLabel("Enter the divider and divident:");

    // add text field for both numbers
    JTextField ajtf = new JTextField(8);
    JTextField bjtf = new JTextField(8);

    // calc button
    JButton button = new JButton("Calculate");

    // labels
    JLabel err = new JLabel();
    JLabel alab = new JLabel();
```

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

// add in order :)
jfrm.add(err); // to display error bois
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);

ActionListener l = new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
      System.out.println("Action event from a text field");
   }
};
ajtf.addActionListener(l);
bjtf.addActionListener(l);

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

         alab.setText("\nA = " + a);
         blab.setText("\nB = " + b);
         anslab.setText("\nAns = " + ans);
```

```
            } catch (NumberFormatException e) {
               alab.setText("");
               blab.setText("");
               anslab.setText("");
               err.setText("Enter Only Integers!");
            } catch (ArithmeticException e) {
               alab.setText("");
               blab.setText("");
               anslab.setText("");
               err.setText("B should be NON zero!");
            }
         }
      });

      // display frame
      jfrm.setVisible(true);
   }

   public static void main(String args[]) {
      // create frame on event dispatching thread
      SwingUtilities.invokeLater(new Runnable() {
         public void run() {
            new SwingDemo();
         }
      });
   }
}
```

10a) Interprocess communication using consumer and producer

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

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

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

```
        System.out.println("\nIntimate Consumer\n");
        notify();
    }
}

class Producer implements Runnable {
    Q q;

    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }

    public void run() {
        int i = 0;
        while (i < 6) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;

    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }

    public void run() {
        int i = 0;
```

```
        while (i < 6) {
            int r = q.get();
            System.out.println("consumed:" + r);
            i++;
        }
    }
}

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

10b) Deadlock

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

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

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

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