LAB: 1

Develop a Java program that prints all real solutions to the quadratic equation $ax^2+bx+c = 0$. Read a, b, c and use the quadratic formula. If the discriminate b^2-4ac is negative, display a 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("Roo1 = 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);
}

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

q.compute();
}
}
```

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

```
public class Subject
{
int subjectmarks;
int credits;
int grade;
}
import java.util.Scanner;
public class Student
{
Subject sub[];
String name;
String usn;
double sgpa;
Scanner scan=new Scanner(System.in);
Student()
{
sub=new Subject[8];
for(int i=0;i<8;i++)
sub[i]=new Subject();
scan=new Scanner(System.in);
}
public void getStudentDetails()
{
System.out.println("Enter name:");
name=scan.nextLine();
System.out.println("Enter USN:");
usn=scan.nextLine();
}

public void getMarks()
{
for(int i=0;i<8;i++)
{
System.out.println("Enter Subject "+(i+1)+" marks:");
sub[i].subjectmarks=scan.nextInt();
System.out.println("Enter Subject "+(i+1)+" credits:");
sub[i].credits=scan.nextInt();
if(sub[i].subjectmarks==100)
sub[i].grade=10;
else if(sub[i].subjectmarks<40)
```

```
sub[i].grade=0;
else
sub[i].grade=(sub[i].subjectmarks/10)+1;
}
}

public void computeSGPA()
{
int sumc=0;
double prod=0;
for(int i =0;i<8;i++)
{
sumc=sumc+sub[i].credits;
prod=prod+(sub[i].grade*sub[i].credits);
}
sgpa=prod/sumc;
}
}
public class SGPA
{
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("S.no\tSubject Marks \tCredits\tGrade");
for(int i=0;i<8;i++)
{
System.out.println((i+1)+"\t"+s1.sub[i].subjectmarks+"\t"+s1.sub[i].credits+"\t"+
s1.sub[i].grade);
}
System.out.println("SGPA="+s1.sgpa);
System.out.println("
--------------------------------------------------------------------------------");
System.out.println("Sinchana R - 1BM22CS278");
}
}
```

LAB: 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;
}
}

class Main
{
public static void main(String args[])
{
Scanner s= new Scanner(System.in);
int n;
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];
System.out.println("Enter details of the book");
for(int i=0;i<n;i++){
System.out.println("Enter name of the book");
name=s.next();
System.out.println("Enter the author");
```

```
author=s.next();
System.out.println("Enter price");
price=s.nextInt();
System.out.println("Enter no of pages");
numPages=s.nextInt();
b[i]=new Books(name,author,price,numPages);
}
System.out.println("Book details:");
System.out.println("Book Name \t Author \t Price \t + No of pages");
for(int i=0;i<n;i++){
System.out.println(b[i].name + "\t" + b[i].author + "\t" + b[i].price + "\t" +
b[i].numPages);
}
System.out.println("**********");
System.out.println("SINCHANA R  1BM22CS278");
}
}
```

LAB: 4

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

```
import java.util.Scanner;
class InputScanner {
    Scanner s;
    InputScanner() {
        s = 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 length and width of Rectangle:");
        a = s.nextDouble();
        b = s.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of Rectangle: " + (a * b));
    }
}
class Triangle extends Shape {

    void getInput() {
        System.out.println("Enter base and height of Triangle:");
        a = s.nextDouble();
        b = s.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of Triangle: " + (0.5 * a * b));
    }
}
class Circle extends Shape {
    void getInput() {
        System.out.println("Enter radius of Circle:");
        a = s.nextDouble();
    }
    void displayArea() {
        System.out.println("Area of Circle: " + (Math.PI * a * a));
```

```
    }
}

public class ShapeMain {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle();
        rectangle.getInput();
        rectangle.displayArea();

        Triangle triangle = new Triangle();
        triangle.getInput();
        triangle.displayArea();

        Circle circle = new Circle();
        circle.getInput();
        circle.displayArea();
    }
}
```

LAB: 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.

```
import java.util.Scanner;
```

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

    public Account(String customerName, int accountNumber, String
accountType) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = 0;
    }

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

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

class CurAcct extends Account {
    double minBalance;
    double serviceCharge;

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

    public void deposit(double amount) {
        super.deposit(amount);
        checkMinBalance();
    }
    public void displayBalance() {
        super.displayBalance();
        checkMinBalance();
    }

    private void checkMinBalance() {
        if (balance < minBalance) {
            balance -= serviceCharge;
```

```
            System.out.println("Service charge applied. Updated balance: " +
balance);
        }
    }
}

class SavAcct extends Account {
    double interestRate;

    public SavAcct(String customerName, int accountNumber) {
        super(customerName, accountNumber, "Savings");
        this.interestRate = 0.05;
    }

    public void computeInterest() {
        double interest = balance * interestRate;
        balance += interest;
        System.out.println("Interest computed and deposited. Updated balance: "
+ balance);
    }
    public void displayBalance() {
        super.displayBalance();
        computeInterest();
    }

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawal successful. Updated balance: " +
balance);
        } else {
            System.out.println("Insufficient funds for withdrawal.");
        }
    }
}

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

        System.out.print("Enter your name: ");
        String currentName = scanner.nextLine();
        System.out.print("Enter current account number: ");
        int currentNumber = scanner.nextInt();
        CurAcct currentAccount = new CurAcct(currentName, currentNumber);
```

```
        System.out.print("Enter the amount to deposit into the current account: ");
        double currentDeposit = scanner.nextDouble();
        currentAccount.deposit(currentDeposit);
        currentAccount.displayBalance();


        scanner.nextLine();
        System.out.print("\nEnter your name: ");
        String savingsName = scanner.nextLine();
        System.out.print("Enter savings account number: ");
        int savingsNumber = scanner.nextInt();
        SavAcct savingsAccount = new SavAcct(savingsName, savingsNumber);

        System.out.print("Enter the amount to deposit into the savings account:
");
        double savingsDeposit = scanner.nextDouble();
        savingsAccount.deposit(savingsDeposit);
        savingsAccount.displayBalance();

        System.out.print("Enter the amount to withdraw from the savings
account: ");
        double savingsWithdraw = scanner.nextDouble();
        savingsAccount.withdraw(savingsWithdraw);

        scanner.close();
System.out.println("**********************************************************************
*******************************");
System.out.println("Sinchana R - 1BM22CS278");
    }
}
```

LAB: 6
1.Demonstrate various string constructor with proper Java programs.

```
class SubStringCons {
public static void main(String args[]) {
byte ascii[] = {65, 66, 67, 68, 69, 70 };
String s1 = new String(ascii);
System.out.println(s1);
String s2 = new String(ascii, 2, 3);
System.out.println(s2);
```

```
}
}
```

2. Using getchars(), extract BMSCE from “Welcome to BMSCE College”.

```
public class ExtractSubstring {
    public static void main(String[] args) {
        String inputString = "Welcome to Bmsce college";
        int startIndex = 11;
        int endIndex = 16;
        char[] extractedChars = new char[endIndex - startIndex];
        inputString.getChars(startIndex, endIndex, extractedChars, 0);
        String extractedString = new String(extractedChars);
        System.out.println(extractedString);
    }
}
```

3. Write a Java program to create a generic class tax which holds 5 integers and 5 double values.

```
import java.util.ArrayList;
import java.util.List;

public class GenericStack<T> {
    private List<T> stack;

    public GenericStack() {
        this.stack = new ArrayList<>();
    }

    public void push(T item) {
        if (stack.size() < 5) {
            stack.add(item);
            System.out.println("Pushed: " + item);
        } else {
            System.out.println("Stack is full. Cannot push more elements.");
        }
    }

    public T pop() {
        if (!stack.isEmpty()) {
            T poppedItem = stack.remove(stack.size() - 1);
            System.out.println("Popped: " + poppedItem);
            return poppedItem;
        } else {
            System.out.println("Stack is empty. Cannot pop elements.");
            return null;
        }
    }

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

```
        // Create a generic stack for integers
        GenericStack<Integer> integerStack = new GenericStack<>();
        integerStack.push(1);
        integerStack.push(2);
        integerStack.push(3);
        integerStack.pop();
        integerStack.pop();

        // Create a generic stack for doubles
        GenericStack<Double> doubleStack = new GenericStack<>();
        doubleStack.push(1.1);
        doubleStack.push(2.2);
        doubleStack.push(3.3);
        doubleStack.pop();
        doubleStack.pop();
    }
}
```

4. Write a Java program to create an abstract class, Bird with abstract methods fly () and makeSound ().Create subclasses, Eagle and Hawk that extends the Bird class and implement the respective methods to describe how each bird flies and makes a sound.

```
import java.util.ArrayList;
import java.util.List;

public class GenericStack<T> {
    private List<T> stack;

    public GenericStack() {
        this.stack = new ArrayList<>();
    }

    public void push(T item) {
        if (stack.size() < 5) {
            stack.add(item);
            System.out.println("Pushed: " + item);
        } else {
            System.out.println("Stack is full. Cannot push more elements.");
        }
    }

    public T pop() {
        if (!stack.isEmpty()) {
            T poppedItem = stack.remove(stack.size() - 1);
            System.out.println("Popped: " + poppedItem);
            return poppedItem;
        } else {
            System.out.println("Stack is empty. Cannot pop elements.");
```

```
            return null;
        }
    }

    public static void main(String[] args) {
        // Create a generic stack for integers
        GenericStack<Integer> integerStack = new GenericStack<>();
        integerStack.push(1);
        integerStack.push(2);
        integerStack.push(3);
        integerStack.pop();
        integerStack.pop();

        // Create a generic stack for doubles
        GenericStack<Double> doubleStack = new GenericStack<>();
        doubleStack.push(1.1);
        doubleStack.push(2.2);
        doubleStack.push(3.3);
        doubleStack.pop();
        doubleStack.pop();
    }
}
```

LAB: 7

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

```
protected String name = new String();
protected int sem;
public void inputStudentDetails() {
Scanner scan=new Scanner(System.in);
System.out.println("Enter name:");
name=scan.next();
System.out.println("Enter usn:");
usn=scan.next();
System.out.println("Enter sem:");
sem=scan.nextInt();

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

//Internals class
package CIE;
import java.util.Scanner;
public class Internals extends Student
 {
protected int marks[] = new int[5];
public void inputCIEmarks()

{
Scanner scan=new Scanner(System.in);
System.out.println("Enter internal marks of 5 subjects:");
for(int i=0;i<marks.length;i++)
{
marks[i]=scan.nextInt();
}
}
}

//External class
package SEE;
import CIE.Internals;
import java.util.Scanner;
public class External extends Internals {
protected int marks[];
protected int finalMarks[];

public External() {
marks = new int[5];
finalMarks = new int[5];
 }

public void inputSEEmarks() {
Scanner s = new Scanner(System.in);
```

```
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]);
}
}

//MainMarks
import SEE.External;

class MainMarks
{
public static void main(String args[])
{
int numOfStudents = 2;
External finalMarks[] = new External[numOfStudents];
for(int i=0;i<numOfStudents;i++)
{
finalMarks[i] = new External();
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();
} //end of for loop
}
}
```

LAB: 8
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 Father {
    int fatherAge;

        Father() throws WrongAge {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter father's age: ");
```

```
        fatherAge = s.nextInt();
        if (fatherAge < 0) {
            throw new WrongAge("Age cannot be negative");
        }
    }


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


class Son extends Father {
  int sonAge;


     Son() throws WrongAge {
        super();
        Scanner s = new Scanner(System.in);
        System.out.print("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 == fatherAge){
            throw new WrongAge("Son's age cannot be equal to 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 Mains {
    public static void main(String[] args) {
        try {
            Son s = new Son();
            s.display();
        } catch (WrongAge e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

LAB: 9

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 {
    private String message;
    private int delay;

    public DisplayThread(String message, int delay) {
        this.message = message;
        this.delay = delay;
    }

    public void run() {
        try {
            while (true) {
                System.out.println(message);
                Thread.sleep(delay * 1000);
            }
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted.");
        }
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        DisplayThread thread1 = new DisplayThread("BMS College of Engineering", 10);
        DisplayThread thread2 = new DisplayThread("CSE", 2);

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

2. Demonstrate Inter Process Communication and Deadlock.

i)Inter Process Communication

```
class Q {
int n;
boolean valueSet = false;
synchronized int get() {
while(!valueSet)
try {
System.out.println("Siri Sathish 1BM22CS280");
System.out.println("Consumer waiting");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
System.out.println("Got: " + n);
valueSet = false;
System.out.println("Intimate Producer");
notify();
return n;
}
synchronized void put(int n) {
while(valueSet)
try {
System.out.println("Producer waiting");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("Intimate Consumer");
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.");
System.out.println("*************************");
System.out.println("SINCHANA R 1BM22CS278");
}
}
```

ii)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 mains(String args[]) {
        new Deadlock();
    }
}
```

LAB: 10
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, 200);
      jfrm.setLayout(new FlowLayout());
      // to terminate on close
      jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

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

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

    button.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
            try {
                int a = Integer.parseInt(ajtf.getText());
                int b = Integer.parseInt(bjtf.getText());
                if (b == 0) {
                    throw new ArithmeticException("B should be non-zero!");
                }
                int ans = a / b;

                alab.setText("\nDividend (A) = " + a);
                blab.setText("\nDivisor (B) = " + b);
                anslab.setText("\nResult = " + ans);
                err.setText("");
            } catch (NumberFormatException e) {
                err.setText("Enter Only Integers!");
                alab.setText("");
                blab.setText("");
                anslab.setText("");
            } catch (ArithmeticException e) {
                err.setText("B should be non-zero!");
                alab.setText("");
                blab.setText("");
                anslab.setText("");
            }
        }
    });

    // 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();
        }
```

```
        });
    }
}
```

1. Quadratic Equation :

```
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 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 = nextInt();
}
d = b*b - 4*a*c;
if(d==0){
    r1 = (-b)/(2*a);
System.out.println("Roots are real & equal");
System.out.println("Root 1 = Root 2 =" + r1);
}
else if(d>0){
    r1 = ((-b)+(math.sqrt(d)))/(double)(2*a);
    r2 = ((-b)-(math.sqrt(d)))/(double)(2*a);
System.out.println("Root 1 =" + r1 + "Root 2 =
                    " + 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 1="+r1+"-i"+r2);
}
}
}
class Quadratic Main{
    public static void main(String args[]){
    Quadratic q = new Quadratic();
    q.getd();
    q.compute();
}}
```

Output:

Enter the coeffiients of a, b, c

1
2
3

Roots are imaginery.

Root 1 = -1.0 + i1.41421356
Root 2 = -1.0 - i1.41421356

Enter the coefficients of a, b, c
1
-4
4

Roots are real & equal.
Root 1 = Root 2 = 2.0

Enter the coefficients of a, b, c
5
6
1

Roots are real & distinct.
Root 1 = -0.2 Root 2 = -1.0.

A 19-12-23

19-12-23

1. Develop a Java program to create class Student with members USN, Name, an array credits & an array marks. Include methods to accept & display details & a method to calculate SGPA of a student.

$$SGPA = \frac{\sum[(Course\ Credits)(Grade\ Points)]}{\sum[(Course\ Credits)]}$$

considering all courses registered in that semester (including those with F Grade).

$$CGPA = \sum[(Course\ Credits)(Grade\ Points)]$$

→
```
public class Subject
{
int subjectmarks;
int credits;
int grade;
}
import java.util.Scanner;
public class Student
{
Subject sub[];
String name;
String usn;
double sgpa;
Scanner scan = new Scanner(System.in);
Student()
{
sub = new Subject[8];
for(i=0; i<8; i++)
sub[i] = new Subject();
scan = new Scanner(System.in);
```

```
}
public void getStudentDetails()
{
System.out.println("Enter name:");
name = scan.nextLine();
System.out.println("Enter USN:");
usn = scan.nextLine();
}
public void getMarks;
{
for(int i=0; i<8; i++)
{
System.out.println("Enter Subject "+(i+1)+" marks:");
sub[i].subjectmarks = scan.nextInt();
System.out.println("Enter Subject "+(i+1)+" credits:");
sub[i].credits = scan.nextInt();
if (sub[i].subjectmarks==100)
sub[i].grade=10;
else if (sub[i].subjectmarks<40)
sub[i].grade==0;
else
sub[i].grade=(sub[i].subjectmarks/10)+1;
}
}

public void computeSGPA()
{
int sumc=0;
double prod=0;
for(int i=0; i<8; i++)
{
sumc=sumc+sub[i].credits;
prod=prod+(sub[i].grade*sub[i].credits);
}
```

```
sgpa = (prod/sum);
}
}
public class SGPA
{
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("S.no \t Subject Marks \t Credits \t
    Grade");
for(int i=0; i<8; i++)
{
System.out.println((i+1)+"\t"+s1.sub[i].subjectmarks +
  "\t" + s1.sub[i].credits + "\t" + s1.sub[i].grade);
}
System.out.println("SGPA="+s1.sgpa);
System.out.println("**********");
System.out.println("Sinchana.R - 1BM22CS278");
}
}
```

OUTPUT:

Enter name:
Sinchana R — ~~1BM22~~
Enter USN:
1BM22CS278
Enter Subject 1 marks:
98
Enter Subject 1 credits:
4
Enter Subject 2 marks:
93
Enter Subject 2 credits:
4
Enter Subject 3 marks:
89
Enter Subject 3 credits:
3
Enter Subject 4 marks:
91
Enter Subject 4 credits:
3
Enter Subject 5 marks:
93
Enter Subject 5 credits:
3
Enter Subject 6 marks:
96
Enter Subject 6 credits:
3
Enter Subject 7 marks:
93
Enter Subject 7 credits:
1
Enter Subject 8 marks:
94

Enter Subject & credits:

1.

Name: Sinchana .R.

USN: 1BM22CS278

S.No	Subject	Credits	Grade
1	98	4	10
2	93	4	10
3	89	3	9
4	91	3	10
5	93	3	10
6	96	3	10
7	93	1	10
8	94	1	10

SGPA = 9.863636

****

Sinchana .R - 1BM22CS278.

26-12-23

LAB-3

1.
```
import java.util.Scanner;
class Book {
    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;
```

System.out.println("Enter no. of books");

```
int price;
int numPages;
n = s.nextInt();
Books b[];
b = new Books[n];
system.out.println("Enter details of the book")
for(int i=0; i<n; i++){
System.out.println("Enter name of the book:");
name = s.nextLine();
system.out.println("Enter the author:");
author = s.nextLine();
system.out.println("Enter price:");
price = s.nextInt();
system.out.println("Enter no. of pages:");
numPages = s.nextInt();
b[i] = new Books(name, author, price, numPages);
}
System.out.println("Book Details:");
System.out.println("Book Name \t Author \t Price
\t + No: of pages");
for(int i=0; i<n; i++){
System.out.println(b[i].name + "\t" + b[i].
author + "\t" + b[i].price + "\t" + b[i].
numPages);
}
}
}
```

Output:

```
Enter number of books
3
Enter details of the book:
Enter name of the book:
Slow
Enter price author:
Siri
Enter price:
450
Enter no. of pages:
350
Enter name of the book:
Sunset
Enter author:
Samir
Enter price:
890
Enter no. of pages
246
Enter name of the book:
Glory
Enter author:
Sahana
Enter price:
750
Enter no. of pages
346
Book details:
```

classmate
Date
Page

Book Name	Author	Price	No of pages
Slow	Siri	350	350
Sunset	Sami	246	246
Glory	Sahana	346	346

LAB 2-1-23

2-1-24

LAB - 4.

1. Develop a Java program to create an abstract class named Shape that contains 2 integers & an empty method named printArea(). Provide 3 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;
class InputScanner{
        Scanner s;
          Input InputScanner(){
         s = 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 length & width of
          Rectangle");
      a = s.nextDouble();
      b = s.nextDouble();
    }
    void displayArea(){
      System.out.println("Area of Rectangle :"+(a*b));
    }
}
```

```
class Triangle extends Shape {
    void getInput() {
        System.out.println("Enter base & height of
            Triangle : ");
        a = s.nextDouble();
        le = s.nextDouble();
    }
    void displayArea() {
        System.out.println("Area of Triangle : " + (0.5 * a * le));
    }
}
class Circle extends Shape {
    void getInput() {
        System.out.println("Enter radius of circle:");
        a = s.nextDouble();
    }
    void displayArea() {
        System.out.println("Area of circle: " + (Math.PI * a
                                                  * a));
    }
}
public class Main {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle();
        rectangle.getInput();
        rectangle.displayArea();

        Triangle triangle = new Triangle();
        triangle.getInput();
        triangle.displayArea();

        Circle circle = new Circle();
        circle.getInput();
        circle.displayArea();
    }
}
```

OUTPUT:

Enter length and width of Rectangle:
4 5
Area of Rectangle : 20.0
Enter base and height of Triangle:
8 6
Area of Triangle : 24.0
Enter radius of Circle:
6
Area of Circle : 113.09733552923255.

9-1-24

1. Develop a Java Program to create a class Bank that maintains 2 kinds of account 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 & if the balance falls below this level, a service charge is imposed. Create a class Account that stores customer name, account number & type of account. From this derive the classes Cur-acct & 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 & update the balance.

Check for the minimum balance, impose penalty if necessary & update the balance.

```
import java.util.Scanner{
class Account{
    String customerName;
    int accountNumber;
    String accountType;
    double balance;

    public Account(String customerName, int accountNumber,
String accountType, double balance){
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = 0;
    }
public void deposit(double amount){
        balance += amount;
        System.out.println("Account Deposit successful. Updated
    Balance: " + balance);
}
public void displayBalance(){
    System.out.println("Account Balance: " + balance);
    }
}
class CurAcct extends Account{
    double minBalance;
    double serviceCharg;
  public CurAcct(String customerName, int accountNumber){
    super(customerName, accountNumber, "Current");
  this.minBalance = 1000;
  this.serviceCharge = 50;
}
  public void deposit(double amount){
        super.deposit(amount);
        checkMinBalance();
}
```

```
public void displayBalance(){
    super.displayBalance();
    checkMinBalance();
}
private void checkMinBalance(){
    if(balance < minBalance){
        balance -= serviceCharge;
        System.out.println("Service charge applied.
Updated balance:"+balance);
    }
    }
}
class SavAcct extends Account{
    double interestRate;

    public SavAcct(String CustomerName, int
accountNumber){
    super(customerName, accountNumber, "Savings");
    this.interestRate = 0.05;
}
public void computeInterest(){
    double interest = balance * interestRate;
    balance += interest;
    System.out.println("Interest ~~compounded~~ computed
and deposited. Updated balance:"+balance);
}
public void displayBalance(){
    super.displayBalance();
    computeInterest();
}
public void withdraw(double amount){
    if(balance >= amount){
        balance -= amount;
        System.out.println("Withdrawal succesful.
```

```
        Updated balance:" + balance);
    }else{
        System.out.println("Insufficient funds for
        withdrawal.");
    }
}
}
public class Bank{
    public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter your name:");
    String currentName = scanner.nextLine();
    System.out.print("Enter Account number:");
    int currentNumber = scanner.nextInt();
    CurAcct currentAccount = new CurAcct(currentName,
    currentNumber);

    System.out.print("Enter the amount to deposit
    into current account:");
    double currentDeposit = scanner.nextDouble();
    currentAccount.deposit(currentDeposit);
    currentAccount.displayBalance();

    scanner.nextLine();
    System.out.print("\nEnter name:");
    String savingsName = scanner.nextLine();
    System.out.print("Enter savings account number:");
    int savingsNumber = scanner.nextInt();
    SavAcct savingAccount = new SavAcct(savingsName,
    savingsNumber);
```

```
System.out.print("Enter the amount to withdraw
from the savings account: ");
double savingsWithdraw = scanner.nextDouble();
savingAccount.withdraw(savingsWithdraw);
scanner.close();
    }
}
```

→ OUTPUT :

```
Enter your Name : Joe
Enter current account number : 2567
Enter the amount to deposit into the current account:
5000
Deposit successful. Updated balance : 5000.0
Account Balance : 5000.0

Enter your Name : Safa
Enter savings account number : 5678
Enter the amount to deposit into the savings accoun
4500
Deposit successful. Updated balance : 4500.0
Account Balance : 4500.0
Interest computed and deposited. Updated balance :472
Enter the amount to withdraw from the savings
account: 500
Withdrawal successful. Updated balance : 4225.0
```

3/11/24

16-1-24.

LAB - 6

1. Demonstrate various string constructor with proper java programs.

```
class SubStringCons{
    public static void main(String args[]){
    byte ascii[] = {65,66,67,68,69,70};
    String s1 = new String(ascii);
    System.out.println(s1);
    String s2 = new String(ascii,2,3);
    System.out.println(s2);
    }
}
```

Output:

ABCDEF

CDE

2. Using getchars(), extract BMSCE from "Welcome to BMSCE College".

```
public class ExtractSubstring{
    public static void main(String[] args){
    String inputString = "Welcome to Bmsce College";
    int startIndex = 11;
    int endIndex = 16;
    char[] extractedChars = new char[endIndex - startIndex];
    inputString.getChars(startIndex, endIndex, extractedChars, 0);
    String extractedString = new String(extractedChars);
    System.out.println(extractedString);
    }
}
```

Output:

Bmscl.

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

```
import java.util.ArrayList;
import java.util.List;
public class GenericStack<T>{
    private List<T> stack;
    public GenericStack(){
        this.stack = new ArrayList<>();
}
public void push(T item){
    if(stack.size() < 5){
        stack.add(item);
        System.out.println("Pushed: " + item); }
    else{
        System.out.println("Stack is empty. Cannot
        pop elements.");
        return null;
    }
}
public static void main(String[] args){
    GenericStack<Integer> integerStack = new GenericStack<>();
    integerStack.push(1);
    integerStack.push(2);
    integerStack.push(3);
    integerStack.pop();
    integerStack.pop();
```

```
GenericStack<Double> doubleStack = new GenericStack<>();
doubleStack.push(1.1);
doubleStack.push(2.2);
doubleStack.push(3.3);
doubleStack.pop();
doubleStack.pop();
}
}
```

Output:

```
Pushed: 1
Pushed: 2
Pushed: 3
Popped: 3
Popped: 2
Pushed: 1.1
Pushed: 2.2
Pushed: 3.3
Popped: 3.3
Popped: 2.2
```

4. Write a Java program to create an abstract class Bird with abstract methods fly() & 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 {
    abstract void fly();
    abstract void makeSound();
}
class Eagle extends Bird {
    void fly() {
        System.out.println("Eagle is soaring high in the sky.");
    }
    void makeSound() {
        System.out.println("Eagle makes a screeching sound.");
    }
}
class Hawk extends Bird {
    void fly() {
        System.out.println("Hawk is gliding gracefully.");
    }
    void makeSound() {
        System.out.println("Hawk makes a piercing cry.");
    }
}
public class BirdTest {
    public static void main(String[] args) {
        Eagle eagle = new Eagle();
        Hawk hawk = new Hawk();
        System.out.println("Eagle Actions:");
        eagle.fly();
        eagle.makeSound();
        System.out.println();
        System.out.println("Hawk Actions:");
        hawk.fly();
        hawk.makeSound(); } }
```

Output:

Eagle Actions:

Eagle is soaring high in the sky.
Eagle makes a screeching sound.

Hawk Actions:

Hawk is gliding gracefully.
Hawk makes a piercing cry.

16/11/24

3-1-24

1. Create a package CIE which has 2 classes - Student & 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 5 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 5 courses of the current semester of the student. Import 2 packages in a file that declares the final marks of n students in all 5 courses.

```
Package CIE;
import java.util.Scanner();
public class Students {
protected String usn = new String();
protected String name = new String();
protected int sem;

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

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

```
package CIE;
import java.util.Scanner;
public class Internals extends Student
{
protected int marks[] = new int[5];
public void inputCIEmarks()
{
Scanner scan = new Scanner(System.in);
System.out.println("Enter internal marks of 5
subjects :");
for(int i =0; i < marks.length; i++)
{
marks[i]=scan.nextInt();
}
}
}
```

```
package SEE;
import CIE.Internals;
import java.util.Scanner;
public class External extends Internals{
protected int marks[];
protected int finalMark[];

public External(){
marks = new int[5];
finalMarks = new int[5];
}

public void inputSEEmarks(){
Scanner s = new Scanner(System.in);
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.External;
class MainMarks
{
public static void main(String args[])
{
int numOfStudents = 2;
Externals finalMarks[] = new Externals[numOfStudents];
~~for(int i = 0; i<5; i++)~~
~~System.out.println("Sub~~
for(int i = 0; i < numOfStudents; i++){
finalMarks[i] = new External();
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();
}
}
}
```

Output:

Enter USN:
252
Enter Name:
Sahana
Enter Semester:
3
Enter CIE marks
Enter the marks of 5 subjects:
Enter marks for Subject 1: 80
Enter marks for Subject 2: 75
Enter marks for Subject 3: 92
Enter marks for Subject 4: 89
Enter marks for Subject 5: 93

Enter SEE marks
Subject 1 marks: 90
Subject 2 marks: 79
Subject 3 marks: 83
Subject 4 marks: 89
Subject 5 marks: 93

Enter USN:
293
Enter Name:
Shravya
Enter Semester:
3
Enter CIE marks
Enter the marks of 5 subjects:
Enter marks for Subject 1: 91
Enter marks for Subject 2: 78
Enter marks for Subject 3: 89

Enter marks for Subject 4 : 98
Enter marks for Subject 5 : 83

Enter SEE marks
Subject 1 marks : 86
Subject 2 marks : 92
Subject 3 marks : 76
Subject 4 marks : 91
Subject 5 marks : 98
Displaying data :

Name : Sahana
USN : 252
Sem : 3
Subject 1 : 125
Subject 2 : 114
Subject 3 : 133
Subject 4 : 133
Subject 5 : 139

3+11+24

Name : Shravya
USN : 293
Sem : 3
Subject 1 : 134
Subject 2 : 124
Subject 3 : 127
Subject 4 : 143
Subject 5 : 132

30-1-24

1. 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 Father {
    int fatherAge;
    Father() throws Wrongage {
    Scanner s = new Scanner(System.in);
    System.out.print("Enter father's age: ");
    fatherAge = s.nextInt();
    if (fatherAge < 0){
        throw new Wrongage("Age cannot be
    negative");
    }
}
void display(){
    System.out.println("Father's age: " + fatherAge);
}
}
```

```
class Son extends Father {
    int sonAge;

    Son() throws WrongAge {
        super();
        Scanner s = new Scanner(System.in);
        System.out.print("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 == fatherAge) {
            throw new WrongAge("Son's age cannot be
            equal to 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 Main {
    public static void main(String[] args) {
        try {
            Son s = new Son();
            s.display();
        } catch (WrongAge e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

OUTPUT:

(i) Enter father's age : 68
Enter son's age : 34
Father's age : 68
Son's age : 34

(ii) Enter father's age : 56
Enter son's age : -6
Exception: Age cannot be negative.

(iii) Enter father's age : 45
Enter son's age : 57
Exception: Son's age cannot be greater than father's age.

(iv) Enter father's age : 46
Enter son's age : 46
Exception : Son's age cannot be equal to father's age.

30/12/24

16-2-24

1. Write a program which creates 2 threads, one thread displaying "BMS College of Engineering" once every 10 seconds & another displaying "CSE" once every two seconds.

```
class DisplayThread extends Thread {
    private String message;
    private int delay;

    public DisplayThread(String message, int delay) {
        this.message = message;
        this.delay = delay;
    }
    public void run() {
        try {
            while(true) {
                System.out.println(message);
                Thread.sleep(delay * 1000);
            }
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted.");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        DisplayThread thread1 = new DisplayThread
        ("BMS College of Engineering", 10);
        DisplayThread thread2 = new DisplayThread
        ("CSE", 2);
        thread1.start();
        thread2.start();
    }
}
```

O/P:

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

2. Demonstrate Inter Process Communication and Deadlock.

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

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

O/P:

```
Bincham IBM22CS278
Mainthread entered A.foo
Racingthread entered B.bar
Racingthread trying to call A.last()
Mainthread trying to call B.last()
Inside A.last
Back in main thread
Inside A.last
Back in other thread
```

6). class Q {
int n;
boolean valueSet = false;
synchronized int get() {
while (!valueSet)
try {
System.out.println("Sinchana R 1BM22CS278");
System.out.println("Consumer Waiting");
wait();
} catch (InterruptedException e) {
System.out.println("InterruptedException caught");
}
System.out.println("Got:" + n);
valueSet = false;
System.out.println("Intimate Producer");
notify();
return n;
}
synchronized void put(int n){
while (valueSet)
try {
System.out.println("Producer Waiting");
wait();
} catch (InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Put:" + n);
System.out.println("Intimate Consumer");
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.");
}
}
```

O/P:

Put : 0
Intimate Consumer
Producer Waiting
Got : 0
Intimate Producer
Put : 1
Intimate Consumer
Producer Waiting
Consumed : 0
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
Consumed : 3
Put : 4
Intimate Consumer
Got : 4
Intimate Producer
Consumed : 4.

13.02.24

20-2-24.

1. Write a program that creates interface to perform integer divisions. The user enters 2 numbers in the text fields, Num1 & Num2. The division of Num1 & Num2 is displayed in the Result field when the Divide button is clicked. If Num1 & 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(){
      JFrame jfrm = new JFrame("Divider app");
      jfrm.setSize(275,200);
      jfrm.setLayout(new FlowLayout());
      jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_
      CLOSE);
      JLabel jlab = new JLabel("Enter 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);
```

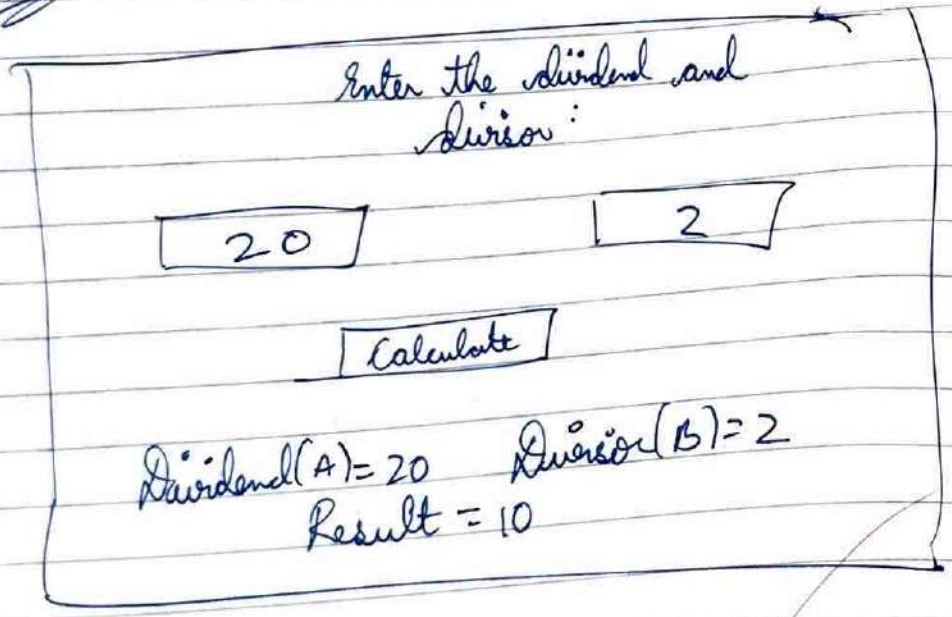
```
jfram.add(lbltf);
jfrm.add(button);
jfrm.add(err);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);

button.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent evt){
  try{
    int a = Integer.parseInt(aftf.getText());
    int b = Integer.parseInt(bftf.getText());
    if(b==0){
      throw new ArithmeticException
        ("B should be non-zero!");
    }
    int ans = a/b;

    alab.setText("\n Dividend (A) = " + a);
    blab.setText("\n Divisor (B) = " + b);
    anslab.setText("\n Result = " + ans);
    err.setText(" ");
  }catch(NumberFormatException e){
    err.setText("Enter only Integers");
    alab.setText(" ");
    blab.setText(" ");
    anslab.setText(" ");
  }catch(ArithmaticException e){
    err.setText("B should be non-zero!");
    alab.setText(" ");
    blab.setText(" ");
    anslab.setText(" ");
  }
}});
```

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

O/P:



★ Functions used in the program:-

1. SwingDemo() - Constructor for initializing the Swing interface.
2. setSize - for setting the size of JFrame.
3. JFrame() - Constructor for creating a new JFrame.

4. setLayout(LayoutManager mgr) - Sets the layout manager for the JFrame.

5. JLabel(String txt) - Constructor method for creating JLabels with required text content.

6. JTextField(int columns): Creates a JTextFields with specified number of columns.

7. JButton(String text): Creates a JButton with specified text.

8. addActionListener(ActionListener Listener): Method to add an action listener to the button to detect and handle an action.

9. setText(String text): Sets the text of JLabel/JTextfield to the specified text.

10. setVisible(boolean b): Sets the visibility of the JFrame.

28/2/24