B.M.S COLLEGE OF ENGINEERING BENGALURU

Autonomous Institute, Affiliated to VTU

LAB REPORT

23CS3PCOOJ

Submitted in partial fulfilment of the requirements for Lab

Bachelor of Engineering

In

Computer Science and Engineering

Submitted by:

NITHYA LAKSHMI. V

(1BM22CS186)

Department of Computer Science and Engineering,

B.M.S College of Engineering,

Bull Temple Road, Basavanagudi, Bangalore, 560 019

2023-2024.

INDEX

Sl.No.	TITLE	DATE
1.	LAB – 1	12/12/2023
2.	LAB – 2	19/12/2023
3.	LAB – 3	26/12/2023
4.	LAB – 4	02/01/2024
5.	LAB – 5	09/01/2024
6.	LAB – 6	16/01/2024
7.	LAB – 7	23/01/2024
8.	LAB – 8	30/01/2024
9.	LAB – 9	06/02/2024
10.	LAB - 10	20/02/2024
11.	Complete scanned Observation Book	12/12/2023 - 20/02/2024

LAB – 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.

SOURCE CODE:

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

OUTPUT:

```
PS C:\Users\aDMIN\Desktop\1BM22CS186> javac QuadraticMain.java
PS C:\Users\aDMIN\Desktop\1BM22CS186> java QuadraticMain
Enter the coefficients of a,b,c
1 2 1
Roots are real and equal
Roo1 = Root2 = -1.0
PS C:\Users\aDMIN\Desktop\1BM22CS186> javac QuadraticMain.java
PS C:\Users\aDMIN\Desktop\1BM22CS186> java QuadraticMain
Enter the coefficients of a,b,c
2 6 2
Roots are real and distinct
Roo1 = -0.3819660112501051 Root2 = -2.618033988749895
PS C:\Users\aDMIN\Desktop\1BM22CS1  > javac QuadraticMain.java
PS C:\Users\aDMIN\Desktop\1BM22CS1  > java QuadraticMain
Enter the coefficients of a,b,c
1 1 1
Roots are imaginary
Root1 = 0.0 + i0.8660254037844386
Root1 = 0.0 - i0.8660254037844386
PS C:\Users\aDMIN\Desktop\1BM22CS186> javac QuadraticMain.java
PS C:\Users\aDMIN\Desktop\1BM22CS186> java QuadraticMain
Enter the coefficients of a,b,c
0 1 5
Not a quadratic equation
Enter a non zero value for a:
2
Roots are imaginary
Root1 = 0.0 + i1.5612494995995996
Root1 = 0.0 - i1.5612494995995996
```

LAB – 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.

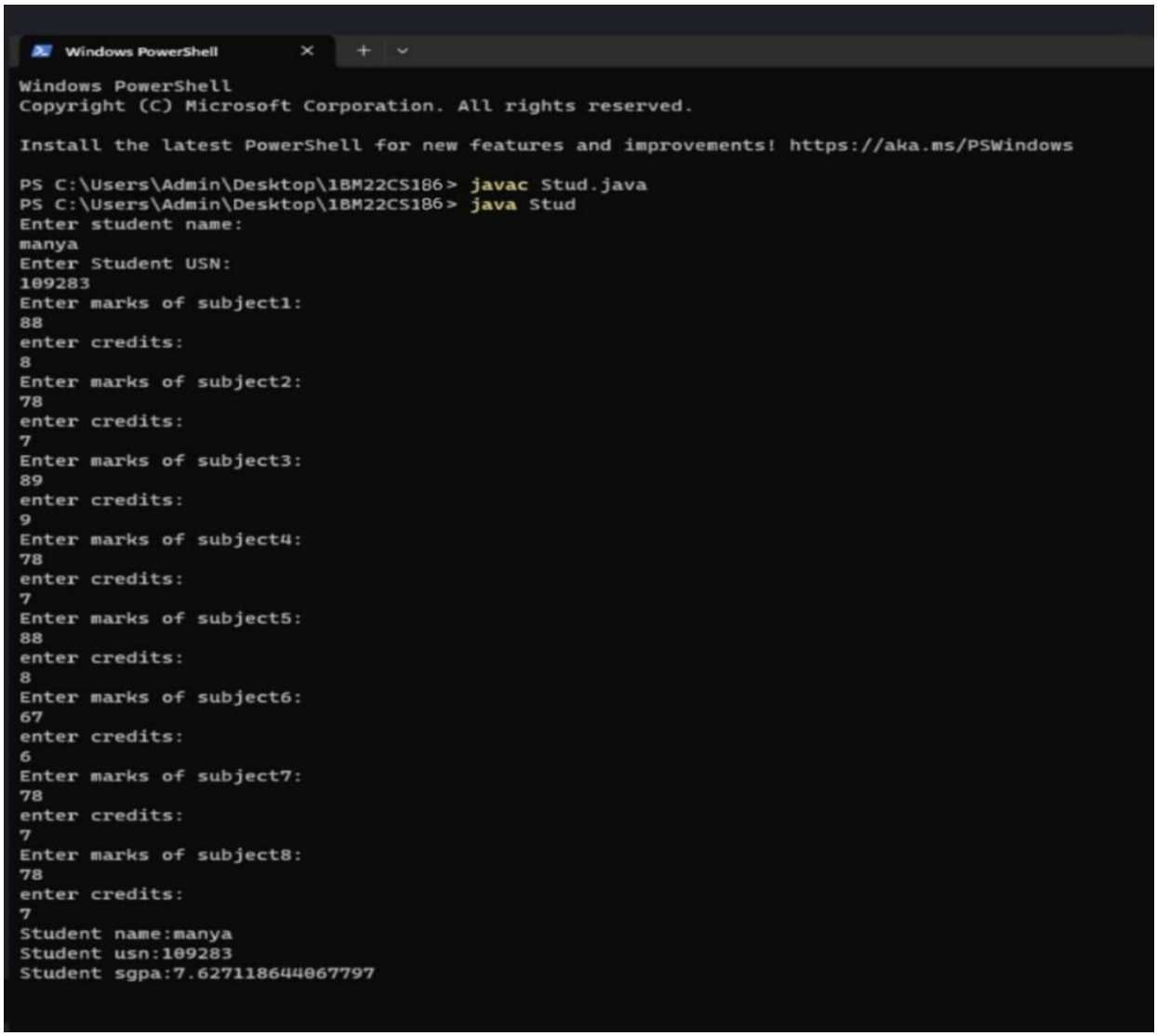
SOURCE CODE:

```
import java.util.Scanner;
 class subject
{
 int subjectMarks, credits, grade;
}
class Student
{
String name;
String usn;
double SGPA;
 Scanner s;
subject subjects[];
Student()
{
 int i;
subjects = new subject[9];
 for(i=0;i=40&&subjects[i].subjectMarks<=100)
{
 subjects[i].grade=calculateGrade(subjects[i].subjectMarks);
}
else
{
 System.out.println("Invalid Marks. Marks should be between 40 and 100");
}
System.out.println("enter credits:");
```

```
subjects[i].credits=s.nextInt();
}
}
public int calculateGrade(int marks)
{
if (marks>=90) return 10;
 else if(marks>=70&&marks<=80)
return 9;
 else if(marks>=60&&marks<=70)
return 8;
 else if(marks>=50&&marks<=60)
return 7;
 else
return 6;
 }
 public void computeSGPA()
{
 int totalscore = 0;
 int totalcred = 0;
 for (int i = 0; i < 8; i++)
{
 totalscore += subjects[i].grade * subjects[i].credits;
 totalcred += subjects[i].credits;
 }
 SGPA = (double) totalscore / (double) totalcred;
 }
 }
 class Stud
{
 public static void main(String args[])
```

```
{
Student s1=new Student();
 s1.getStudentDetails();
 s1.getMarks();
s1.computeSGPA();
 System.out.println("Student name:"+s1.name);
 System.out.println("Student usn:"+s1.usn);
System.out.println("Student sgpa:"+s1.SGPA);
}
 }
```

OUTPUT:



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.

SOURCE CODE:

```
import java.util.Scanner;
 class Book
 {
 private String name;
private String author;
 private double price;
private int numPages;
 public Book(String name, String author, double price, int numPages)
 {
 this.name = name;
this.author = author;
 this.price = price;
 this.numPages = numPages;
}
 public void setName(String name)
{
 this.name = name;
 }
 public String getName()
{
 return name;
 }
 public void setAuthor(String author)
{
```

```
 this.author = author;
}
 public String getAuthor()
{
 return author;
 }
 public void setPrice(double price)
 {
 this.price = price;
 }
 public double getPrice()
{
 return price;
 }
 public void setNumPages(int numPages)
 {
 this.numPages = numPages;
}
 public int getNumPages()
{
 return numPages;
}
 public String toString()
 {
 return "Book Details: \nName: " + name + "\nAuthor: " + author + "\nPrice: INR" + price +
"\nNumber of Pages: " + numPages;
}
 }
 public class Main
{
```

```
public static void main(String[] args)
 {
Scanner scanner = new Scanner(System.in);
System.out.println("Enter the number of books: ");
int n = scanner.nextInt();
 Book[] books = new Book[n];
for (int i = 0; i < n; i++)
{
 System.out.println("\nEnter details for Book " + (i + 1) + ":");
 scanner.nextLine();
System.out.println("Enter name: ");
 String name = scanner.nextLine();
 System.out.println("Enter author: ");
 String author = scanner.nextLine();
 System.out.println("Enter price: ");
 double price = scanner.nextDouble();
System.out.println("Enter number of pages: ");
 int numPages = scanner.nextInt();
 books[i] = new Book(name, author, price, numPages);
 }
 System.out.println("\nDetails of all books:");
for (int i = 0; i < n; i++)
 {
 System.out.println("\nBook " + (i + 1) + ":\n" + books[i]);
}
 scanner.close();
}
}
```

OUTPUT:

Command Prompt

```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin>cd desktop

C:\Users\admin\Desktop>javac Mainbook.java

C:\Users\admin\Desktop>java Main
Enter the number of books:
2

Enter details for Book 1:
Enter name:
The theory of everything
Enter author:
Nithya
Enter price:
450
Enter number of pages:
200

Enter details for Book 2:
Enter name:
Harry Potter
Enter author:
J K Rowling
Enter price:
500
Enter number of pages:
300

Details of all books:

Book 1:
Book Details:
Name: The theory of everything
Author: Nithya
Price: INR450.0
Number of Pages: 200

Book 2:
Book Details:
Name: Harry Potter
Author: J K Rowling
Price: INR500.0
Number of Pages: 300

C:\Users\admin\Desktop>
```

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 classShape. Each one of the classes contain only the method printArea( ) that prints the area of the given shape.

SOURCE CODE:

```
import java.util.Scanner;
 abstract class Shape
{
 protected int sidea;
 protected int sideb;
public Shape(int sidea, int sideb)
 {
 this.sidea = sidea;
 this.sideb = sideb;
 }
 public abstract void printArea();
 }
 class Rectangle extends Shape
{
 public Rectangle(int length, int width)
{
 super(length, width);
 }
 public void printArea()
{
 int area = sidea * sideb;
System.out.println("Area of Rectangle: " + area);
 }
 }
```

```
class Triangle extends Shape
{
 public Triangle(int base, int height)
 {
 super(base, height);
 }
 public void printArea()
{
 double area = 0.5 * sidea * sideb;
 System.out.println("Area of Triangle: " + area);
}
 }
 class Circle extends Shape
{
 public Circle(int radius)
 {
 super(radius, 0);
}
 public void printArea()
 {
double area = Math.PI * sidea * sideb;
 System.out.println("Area of Circle: " + area);
 }
 }
 public class Shapes
{
 public static void main(String[] args)
{
 Scanner scanner = new Scanner(System.in);
 System.out.print("Enter length and width for Rectangle: ");
```

```
int rectLength = scanner.nextInt();
int rectWidth = scanner.nextInt();
 Rectangle rectangle = new Rectangle(rectLength, rectWidth);
 System.out.print("Enter base and height for Triangle: ");
int triBase = scanner.nextInt();
int triHeight = scanner.nextInt();
 Triangle triangle = new Triangle(triBase, triHeight);
System.out.print("Enter radius for Circle: ");
 int circleRadius = scanner.nextInt();
 Circle circle = new Circle(circleRadius);
 scanner.close();
 rectangle.printArea();
 triangle.printArea();
 circle.printArea();
 }
 }
```

OUTPUT:

```
Enter length and width for Rectangle: 2
3
Enter base and height for Triangle: 4
5
Enter radius for Circle: 9
Area of Rectangle: 6
Area of Triangle: 10.0
Area of Circle: 254.46900494077323
```

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. 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.

SOURCE CODE:

```
import java.util.Scanner;
 class Account
{
 String customerName;
int accountNumber;
String accountType;
 double balance;
 Account(String name, int number, String type, double initialBalance)
 {
 customerName = name;
 accountNumber = number;
 accountType = type; balance = initialBalance;
 }
 void deposit(double amount)
 {
 balance += amount;
 System.out.println("Deposit of INR " + amount + " successful");
```

```
}
}
void displayBalance()
{
 System.out.println("Account Number: " + accountNumber);
System.out.println("Customer Name: " + customerName);
System.out.println("Account Type: " + accountType);
 System.out.println("Balance: INR " + balance);
 }
 void withdraw(double amount)
{
 if (balance >= amount)
{
 balance -= amount;
 System.out.println("Withdrawal of INR " + amount + " successful");
 }
 else
 {
 System.out.println("Insufficient funds");
 }
 }
void computeInterest()
 {
 }
void checkMinimumBalance(double minBalance, double serviceCharge)
 {
 }
 }
class SavAcct extends Account
{
```

```
double interestRate = 0.05;
 SavAcct(String name, int number, String type, double initialBalance)
{
 super(name, number, type, initialBalance);
}
 void computeInterest()
 {
 double interest = balance * interestRate;
balance += interest;
System.out.println("Interest of INR " + interest + " added to the account");
}
 }
 class CurAcct extends Account
{
 double minBalance = 1000;
double serviceCharge = 50;
 CurAcct(String name, int number, String type, double initialBalance)
{
 super(name, number, type, initialBalance);
}
 void checkMinimumBalance(double minBalance, double serviceCharge)
 {
 if (balance < minBalance)
 {
 System.out.println("Service charge of INR " + serviceCharge + " imposed");
 balance -= serviceCharge;
 }
 }
 }
 public class Bank
```

```
{
public static void main(String[] args)
 {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter the number of users: ");
int numUsers = scanner.nextInt();
 Account[] accounts = new Account[numUsers];
for (int i = 0; i < numUsers; i++)
{
 System.out.println("\nUser " + (i + 1));
System.out.print("Enter customer name: ");
scanner.nextLine();
 String name = scanner.nextLine();
 System.out.print("Enter account number: ");
 int accNumber = scanner.nextInt();
 System.out.print("Enter initial deposit amount: INR ");
 double initialDeposit = scanner.nextDouble();
 System.out.print("Enter account type (Savings/Current): ");
 scanner.nextLine();
String accType = scanner.nextLine();
 if (accType.equalsIgnoreCase("Savings"))
 {
 accounts[i] = new SavAcct(name, accNumber, accType, initialDeposit);
}
else if (accType.equalsIgnoreCase("Current"))
{
accounts[i] = new CurAcct(name, accNumber, accType, initialDeposit);
}
else
{
```

```
System.out.println("Invalid account type entered. Defaulting to Account.");
 accounts[i] = new Account(name, accNumber, "Account", initialDeposit);
}
 }
 boolean exit = false;
 while (!exit)
 {
 System.out.println("\nChoose an option:");
 System.out.println("1. Deposit");
System.out.println("2. Withdraw");
System.out.println("3. Display Balance");
 System.out.println("4. Compute Interest (Savings only)");
System.out.println("5. Exit");
 System.out.print("Enter your choice: ");
 int choice = scanner.nextInt();
 switch (choice)
{
 case 1:
            System.out.print("Enter account number: ");
            int accNum = scanner.nextInt();
            System.out.print("Enter deposit amount: INR ");
            double depositAmount = scanner.nextDouble();
            for (Account acc : accounts)
             {
                 if (acc.accountNumber == accNum)
                    {
                          acc.deposit(depositAmount);
                     }
               }
             break;
```

```
case 2:
            System.out.print("Enter account number: ");
            accNum = scanner.nextInt();
            System.out.print("Enter withdrawal amount: INR ");
            double withdrawAmount = scanner.nextDouble();
            for (Account acc : accounts)
              {
               if (acc.accountNumber == accNum)
                    {
                       acc.withdraw(withdrawAmount);
                    }
               }
              break;
case 3:
             System.out.print("Enter account number: ");
             accNum = scanner.nextInt();
             for (Account acc : accounts)
             {
               if (acc.accountNumber == accNum)
                  {
                       acc.displayBalance();
                   }
               }
          break;
 case 4:
             System.out.print("Enter account number (for Savings account): ");
             accNum = scanner.nextInt(); for (Account acc : accounts)
             {
                if (acc.accountNumber == accNum && acc instanceof SavAcct)
                   {
```

```
                (
                    (SavAcct) acc).computeInterest();
                }
            }
            break;
case 5:
            exit = true;
            break;
default:
            System.out.println("Invalid choice. Please enter a valid option.");
}
}
}
}
```

OUTPUT:

```
C:\Users\bmsce\Desktop>java Bank

Select an option:
1. Deposit
2. Display Balance
3. Compute Interest (Savings Account only)
4. Withdraw
5. Exit
Enter your choice: 1
Enter amount to deposit: 10000
Select account (1. Current, 2. Savings): 2
Deposit of $10000.0 successful. Updated balance: $12000.0

Select an option:
1. Deposit
2. Display Balance
3. Compute Interest (Savings Account only)
4. Withdraw
5. Exit
Enter your choice: 2
Select account (1. Current, 2. Savings): 1
Account Balance: $1000.0

Select an option:
1. Deposit
2. Display Balance
3. Compute Interest (Savings Account only)
4. Withdraw
5. Exit
Enter your choice: 3
Interest computed and deposited: $600.0
Account Balance: $12600.0

Select an option:
1. Deposit
2. Display Balance
3. Compute Interest (Savings Account only)
4. Withdraw
5. Exit
Enter your choice: 4
Enter amount to withdraw: 3000
Select account (1. Current, 2. Savings): 1
Insufficient funds. Withdrawal failed.

Select an option:
1. Deposit
2. Display Balance
3. Compute Interest (Savings Account only)
4. Withdraw
5. Exit
Enter your choice: 5
```

LAB – 6

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

SOURCE CODE:

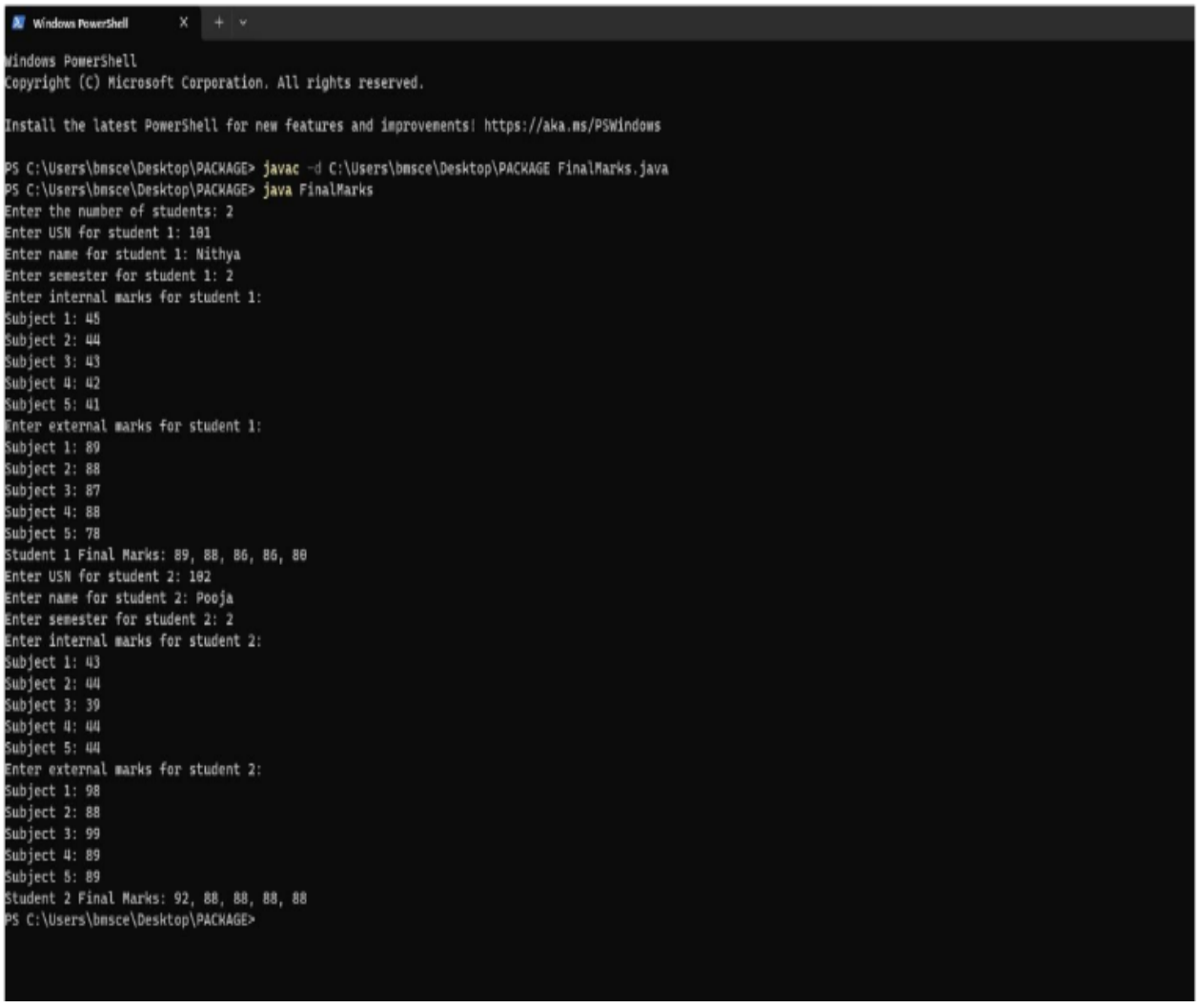
```
#Internals
package CIE;
public class Internals {
private int[] internalMarks = new

int[5];  public Internals() {

}
public void setInternalMarks(int[] internalMarks)

{  this.internalMarks = internalMarks;  }

public int[] getInternalMarks() {
return internalMarks;
}
}
#Student
package CIE;
public class Student {
public String usn;
 public String name;
 public int sem;

 public Student() {

 this("", "", 0);

 }

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

 }
 public void setUsn(String usn)

{  this.usn = usn;
```

```
 }
 public void setName(String name)

{  this.name = name;

 }
 public void setSem(int sem)

{  this.sem = sem;

 }
 public String getUsn() {
 return usn;
 }
 public String getName() {
 return name;
 }
 public int getSem() {
 return sem;
 }
}
#External
package SEE;
import CIE.Student;
public class External extends Student {
 public int[] seeMarks = new int[5];
 public External() {
 this("", "", 0, new int[5]);
 }
 public External(String usn, String name, int sem, int[] seeMarks)

{  super(usn, name, sem);

 this.seeMarks = seeMarks;
 }
 public void setSeeMarks(int[] seeMarks) {
 this.seeMarks = seeMarks;
 }
 public int[] getSeeMarks() {
 return seeMarks;
 }
}
#FinalMarks
import CIE.Student;
import CIE.Internals;
import SEE.External;
import java.util.Scanner;
public class FinalMarks {
 public static void main(String[] args) {
 Scanner scanner = new Scanner(System.in);
```

```
 // Allow the user to enter the number of

students  System.out.print("Enter the number of

students: ");

 int n = scanner.nextInt();
 Student[] students = new Student[n];
 Internals[] internals = new Internals[n];
 External[] externals = new External[n];
 // Initialize students, internals, and externals
 for (int i = 0; i < n; i++) {
 students[i] = new Student();
 System.out.print("Enter USN for student " + (i + 1) + ": ");
 students[i].setUsn(scanner.next());
 System.out.print("Enter name for student " + (i + 1) + ": ");
 students[i].setName(scanner.next());
 System.out.print("Enter semester for student " + (i + 1) + ": ");
 students[i].setSem(scanner.nextInt());
 internals[i] = new Internals();
 // Assuming a simple method to input internal marks with

validation  internals[i].setInternalMarks(inputMarksWithValidation("internal", i, scanner, 0,

50));

 externals[i] = new External(students[i].getUsn(), students[i].getName(), students[i].getSem(),  new
int[5]);
 // Assuming a simple method to input external marks with

validation  externals[i].setSeeMarks(inputMarksWithValidation("external", i, scanner, 0,

100));

 // Calculate final marks for the ith student and display
 int[] finalMarks = new int[5];
 for (int j = 0; j < 5; j++) {
          finalMarks[j] = internals[i].getInternalMarks()[j] + externals[i].getSeeMarks()[j] / 2;
 }
 System.out.println("Student " + (i + 1) + " Final Marks: " +
 finalMarks[0] + ", " + finalMarks[1] + ", " + finalMarks[2] + ", " +
 finalMarks[3] + ", " + finalMarks[4]);
 }
 scanner.close();
 }
 private static int[] inputMarksWithValidation(String type, int studentIndex, Scanner scanner,
int  min, int max) {

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

```
System.out.println("Enter " + type + " marks for student " + (studentIndex + 1) + ":

");  for (int i = 0; i < 5; i++) {

 int mark;
 do {
 System.out.print("Subject " + (i + 1) + ": ");
 mark = scanner.nextInt();
 if (mark < 0 || mark > max) {
System.out.println("Invalid input. " + type + " marks should be between 0 and " + max + ".  Please try
again.");
 }
 } while (mark < 0 || mark > max);
 marks[i] = mark;
 }
 return marks;
 }
}
```

OUTPUT:



LAB – 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 class, implement a constructor that cases both father and son's age and throws an exception if son's age is >=father's age.

SOURCE CODE:

```
import java.util.Scanner;
            class WrongAge extends Exception {
             public WrongAge(String message) {
             super(message);
             }
            }
class Father {
 protected int fatherAge;
 public Father(int age) throws WrongAge {
 fatherAge = age;
 if (fatherAge < 0) {
 throw new WrongAge("Father's age cannot be negative");
     }
    }
    }
class Son extends Father {
 private int sonAge;
 public Son(int fatherAge, int sonAge) throws WrongAge {
 super(fatherAge);
 this.sonAge = sonAge;
 if (sonAge <= 0) {
 throw new WrongAge("Son's age cannot be negative or
zero");  }
 if (sonAge >= fatherAge) {
 throw new WrongAge("Son's age cannot be greater than or equal to father'sage");  }
     }
}
public class Main {
 public static void main(String[] args) {
 Scanner scanner = new Scanner(System.in);
 try {
 System.out.print("Enter father's age: ");
 int fatherAge = scanner.nextInt();
 System.out.print("Enter son's age: ");
 int sonAge = scanner.nextInt();
 Son son = new Son(fatherAge, sonAge);
 System.out.println("Father's age: " + fatherAge);
 System.out.println("Son's age: " + sonAge);
 } catch (WrongAge e) {
 System.out.println("Exception caught: " + e);
```

```
System.out.println("Exception caught: " + e.getMessage());
} catch (Exception e) {
System.out.println("Error: " + e);
System.out.println("Error: " + e.getMessage());
} finally {
scanner.close();
}
}
}
```

OUTPUT:

```
Enter father's age: 35
Enter son's age: 12
Father's age: 35
Son's age: 12
```

```
Enter father's age: 68
Enter son's age: 78
Exception caught: WrongAge: Son's age cannot be greater than or equal to father's age
Exception caught: Son's age cannot be greater than or equal to father's age
```

```
Enter father's age: -56
Enter son's age: 23
Exception caught: WrongAge: Father's age cannot be negative
Exception caught: Father's age cannot be negative
```

```
Enter father's age: 56
Enter son's age: -23
Exception caught: WrongAge: Son's age cannot be negative or zero
Exception caught: Son's age cannot be negative or zero
```

LAB – 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.

SOURCE CODE:

```
class DisplayThread extends Thread
{
 private String message;
 private int interval;
 private boolean running = true;
 public DisplayThread(String message, int interval)
{
 this.message = message;
this.interval = interval;
 }
 public void run()
 {
 while (running)
 {
 System.out.println(message);
 try
{
 Thread.sleep(interval);
}
catch (InterruptedException e)
 {
 e.printStackTrace();
 }
 }
 }
```

```
public void stopThread()
{
 running = false;
 }
 }
 public class ThreadEx
{
 public static void main(String[] args)
{
 DisplayThread bmsThread = new DisplayThread("BMS College of Engineering", 10000);
 DisplayThread cseThread = new DisplayThread("CSE", 2000);
 bmsThread.start();
 cseThread.start();
 System.out.println("Press Enter to stop the threads...");
 Try
 {
System.in.read();
 }
 catch (Exception e)
{
 e.printStackTrace();
}
 bmsThread.stopThread();
 cseThread.stopThread();
}
 }
```

OUTPUT:

```
PS C:\Users\Admin\Desktop\1BM22CS186> javac Thread1.java
PS C:\Users\Admin\Desktop\1BM22CS186> java Thread1
Press Enter to stop the threads...
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
```

LAB – 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.

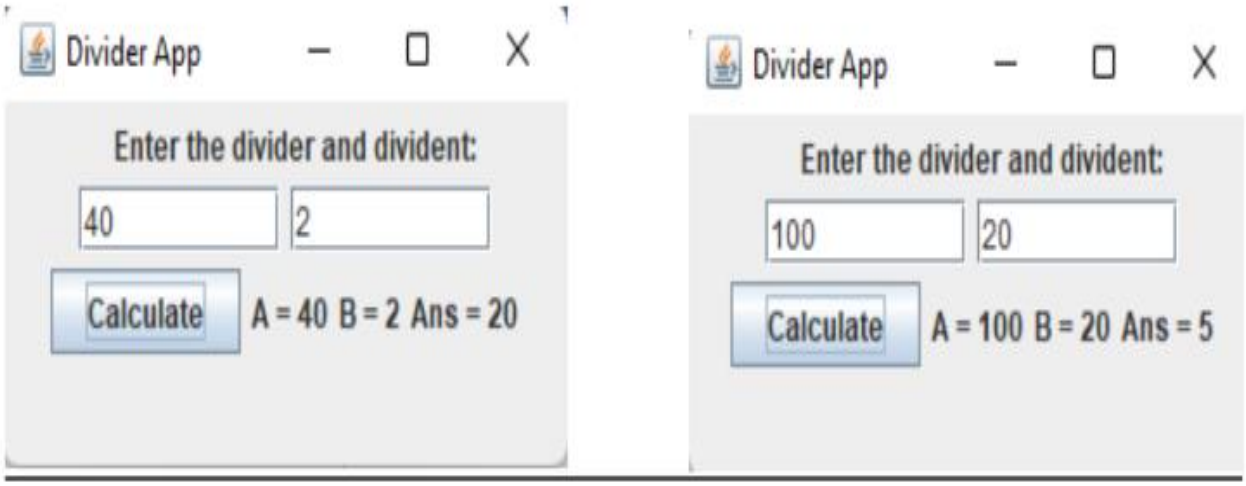
SOURCE CODE:

```
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(&amp;quot;Enter the divider and divident:&amp;quot;);

      JTextField ajtf = new JTextField(8);
      JTextField bjtf = new JTextField(8);
      JButton button = new JButton(&amp;quot;Calculate&amp;quot;);

      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 evt) {
      System.out.println(&quot;Action event from a text field&quot;);
   }
};
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(&quot;\nA = &quot; + a);
         blab.setText(&quot;\nB = &quot; + b);
         anslab.setText(&quot;\nAns = &quot;+ ans);
      }
```

```
catch(NumberFormatException e){
                alab.setText(&amp;quot;&amp;quot;);
                blab.setText(&amp;quot;&amp;quot;);
                anslab.setText(&amp;quot;&amp;quot;);
                err.setText(&amp;quot;Enter Only Integers!&amp;quot;);
            }
            catch(ArithmeticException e){
                alab.setText(&amp;quot;&amp;quot;);

                blab.setText(&amp;quot;&amp;quot;);
                anslab.setText(&amp;quot;&amp;quot;);
                err.setText(&amp;quot;B should be NON zero!&amp;quot;);
            }
        }
      });

      jfrm.setVisible(true);
   }

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

OUTPUT:


Divider App
Enter the divider and divident:
40
2
Calculate
A = 40 B = 2 Ans = 20
Divider App
Enter the divider and divident:
100
20
Calculate
A = 100 B = 20 Ans = 5

LAB – 10

Demonstrate Inter process Communication and deadlock.

SOURCE CODE:

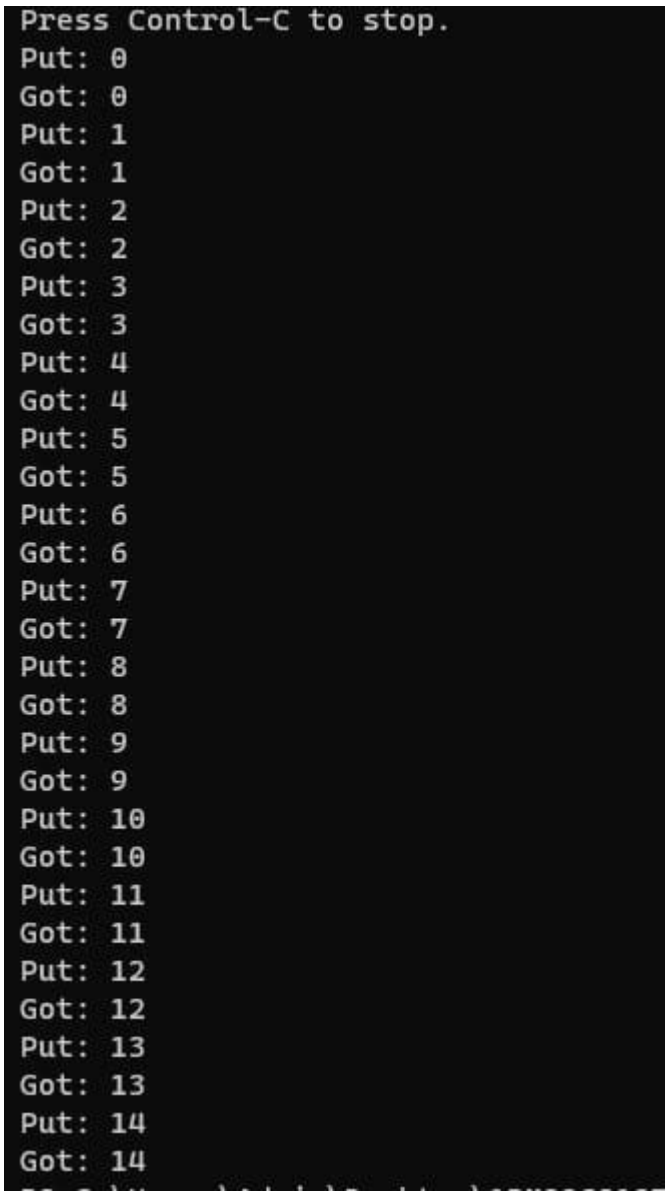
IPC

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

```
wait();
 }
catch(InterruptedException e)
 {
 System.out.println("InterruptedException caught");
 }
 this.n = n;
 valueSet = true;
 System.out.println("Put: " + n); notify();
 }
 }
 class Producer implements Runnable
 {
 Q q;
 Producer(Q q)
 {
 this.q = q;
 new Thread(this, "Producer").start();
 }
 public void run()
 {
 int i = 0;
 while(i<15)
{
q.put(i++);
}
}
}
class Consumer implements Runnable
 {
```

```
 Q q;
 Consumer(Q q)
 {
 this.q = q;
 new Thread(this, "Consumer").start();
}
public void run()
 {
int i=0;
while(i<15)
{
int r=q.get();
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
Got: 0
Put: 1
Got: 1
Put: 2
Got: 2
Put: 3
Got: 3
Put: 4
Got: 4
Put: 5
Got: 5
Put: 6
Got: 6
Put: 7
Got: 7
Put: 8
Got: 8
Put: 9
Got: 9
Put: 10
Got: 10
Put: 11
Got: 11
Put: 12
Got: 12
Put: 13
Got: 13
Put: 14
Got: 14

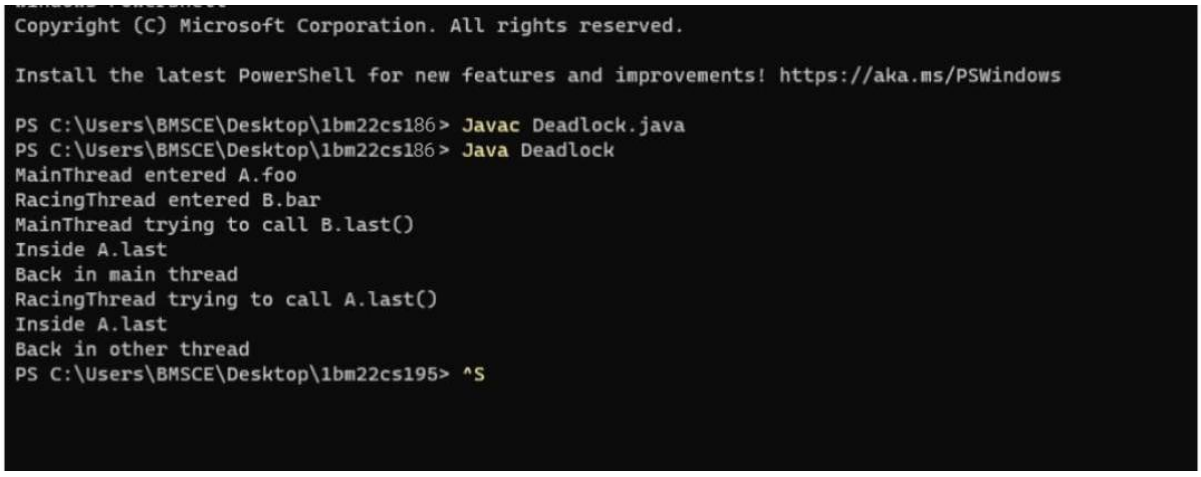
Deadlock

```
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 A.last");

}

}

class Deadlock implements Runnable
{

A a = new A();

B b = new B();

Deadlock() {
Thread.currentThread().setName("M
ainThread");

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();
}
}
```

OUTPUT:



-x-x-x-x-x-x-x-x-x-x-x-x-x-x-

12/12/23

Program - 1 (Quadratic)

```
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
                co-efficients 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 of
                for a:");
            Scanner s = new Scanner(System.i
            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("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("Roots are real and
                            distinct");
        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 2 =" + r1 + "-i" + r2);
    }

    }

}

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

Output:

```
Enter the coefficients of a,b,c
2 3 4 4 5 5
Roots are imaginary
Root 1 = 0.0 + i 1.2150598793462242
Root 2 = 0.0 + -i 1.2150598793462242

Enter the coefficients of a,b,c
0 0 0
Not a quadratic equation
Enter a non-zero value for a
```

19-12-23

19/12/23

Program-2 (Calculate SGPA)

```
import java.util.Scanner;
class subjects
{
    int subjectMarks, credits, grade;
}

class student
{
    String name;
    String USN;
    double SGPA;
    Scanner s;
    Subject subjects[];

    Student()
    {
        int i;
        Subjects = new Subject[9];
        for (i=0; i<8; i++)
        subject[i] = new subject();
        s = new Scanner(System.in);
    }
    public void getStudentDetails()
    {
        System.out.println("Enter Student
                                name");
        name = s.nextLine();

    System.out.println("Enter student USN");
    USN = s.nextLine();
}
```

```
public void getMarks()
{
    int i;
    for(i=0; i<8; i++)
    {
        System.out.println("Enter marks of
            Subject" + (i+1) + ":");

    Subjects[i].SubjectMarks = s.nextInt();
    if(Subjects[i].SubjectMarks >= 40 &&
            subjectsMarks <= 100)

    {
        Subjects[i].grade = calculateGrade
        (Subject[i].SubjectMarks);

    }
    else
    {
        System.out.println("Invalid Marks
            Marks should be between 40
            and 100");
    }

    System.out.println("enter credits");
    Subjects[i].credits = s.nextInt();
}
}
```

```
public int calculateGrade(int marks)
{
    if (marks >= 90)
    return 10;
    else if (marks >= 70 && marks <= 80)
    return 9;
    else if (marks >= 60 && marks <= 70)
    return 8;
    else if (marks >= 50 && marks <= 60)
    return 7;
    else
    return 6;
    }

public void computeSGPA()
{
    int totalScore = 0;
    int totalCredit = 0;
    for (int i = 0; i < 8; i++)
    {
        totalScore += subjects[i].grade *
                      subjects[i].credits;
        totalCredit += subject[i].credits;
    }
    SGPA = (double) totalScore /
           (double) totalCredit;
}
}
```

```
public class stud
{
    public static void main(String args[])
    {
        Student S1 = new Student();
        S1.getStudentDetails();
        S1.getMarks();
        S1.computeSGPA();
        System.out.println("Student name:"
                           + S1.name);
        System.out.println("Student.USN"
                           + S1.USN);
        System.out.println("Student.sgpa"
                           + S1.Sgpa);

    }
```

OUTPUT :

```
Enter student name:
manya
Enter student USN:
107891
Enter marks of subject1:
88
Enter credits:
8
Enter marks of subject2:
78
Enter credits:
7
Enter marks of subject3:
89
Enter credits:
9
Enter marks of subject 4:
78
Enter credits:
7
Enter marks of subject 5:
87
Enter credits:
8
Enter marks of subject 6:
67
Enter credits:
7
Enter marks of subject 7:
78
Enter credits:
7
```

Enter marks of subjects:
78
Enter credits:
7
Student name : manya
Student USN : 107891
Student sgpa : 7.62

19-12-23

Program - 3

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

```
import java.util.*;
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 + numpage;
}

}

class Main
{
    public static void main (String args[])
    {
        Scanner s = new Scanner (System.in);
        int n, i;
        String name;
        String author;
        int price;
        int numpages;
        System.out.println("Enter the no. of
                                  books");
        n = s.nextInt();
        Books b[];
        b = new Books[n];
        for (i=0; i<n; i++)
        {
            name = s.next();
            author = s.next();
            price = s.nextInt();
            pnumpages = s.nextInt();
            b[i] = new Books (name, author,
                              price, numpages);
        }
```

```
for(i=0; i<n; i++)
{
    System.out.println("name of Book "+name);
    System.out.println("author = "+author);
    System.out.println("price = "+price);
    System.out.println("No. of pages = "+
                                    numpages);
}
}
}
```

output

```
Enter the number of books:
2
Enter details for Book 1:
Enter name of Book:
The Theory of Everything
Author:
Nithya
Price:
200
No. of Pages:
400
```

26/12/2023

2/1/24

LAB-4 Area of Triangle, Rectangle and circle

```
import java.util.Scanner;
abstract class Shape{
protected int sidea;
protected int sideb;
public Shape(int sidea, int sideb)
  {
     this.sidea = sida;
     this.sideb = sideb;
  }
   public abstract void printArea();
}
class Rectangle extends Shape {
public Rectangle (unit length,
int width) super (length, width);
}
 public void printArea(){
    int area = side a * sideb;
 System.out.println("Area of Rectangle:"+
                                area);
 }
 }
 class triangle extends Shape{
 public Triangle (int base, int
  height) {
      super(base, height);
  }
  public void printArea(){
    double area= 0.5* side a* sideb;

 System.out.println("Area of Triangle:"
                                + area);
 }
```

class Circle extends shape{
public Circle (int radius) {
super.(radius, 0);
}

public void printArea(){
double area = MATH.PI * side a*
side b;

System.out.println("Area of circle
+ area);

}

public class shapes{

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

System.out.println("Enter length &
width of Rectangle
");
int rectlength = Scanner.nextInt();
int rectwidth = Scanner.nextInt();
Triangle triangle = new Rectangle
(rectlength,
rectwidth);
System.out.println("Enter b & b
for triangle");
int tribase = Scanner.nextInt();
int rectwidth = Scanner.nextInt();
Triangle triangle = new triangle
(tribase, triheight);

```
System.out.println("Enter radius :");
int circleRadius = scanner.nextInt();
Circle circle = new Circle(circleRadius);

Scanner.close();
rectangle.printArea();
triangle.printArea();
Circle.printArea();
}
}
```

output :-

Enter length and Breadth for rectangle
: 2 3

Enter base and height for triangle :
5 4

Enter radius for circle = 9
Area of Rectangle : 6
Area of triangle : 10.0
Area of circle : 254.47

AB

12/1/2024

9/1/24

LAB - 5 Bank Account

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

  public Account(String customerName,
    long accountNumber, String accountTyp
    double balance)
  {
    this.CustomerName = customerName;
    this.AccountNumber = AccountNumber;
    this.accountType = accountType;
    this.Balance = balance;
  }
  public void deposit(double amount)
  {
    balance += amount;
    System.out.println("Deposit of $"
                       + amount + "
                       Successful updated
                       balance $" +
                       balance);
  }
  public void displayBalance()
  {
    System.out.println("Acc
                       balance: $"
                       + balance);
  }
}
```

```
class CurAccount extends Account{

    double minBalance;
    double serviceCharge;

public CurAccount (String CustomerName,
    long ~~ban~~ accountNumber, double
    balance, double minBalance,
    double serviceCharge)
    {
      super(CustomerName, accountNumber,
      "Current", balance);
      this.minBalance = minBalance;
      this.serviceCharge = serviceCharge;
}
public void checkminBalance()
  {
      if(balance < minBalance)
        {
          balance -= serviceCharge;
        System.out.println("Min. balance
        maintained. service charge of $" +
          serviceCharge + imposed");
          displayBalance();
        }
}
public void withdraw(double amount)
    if(amount > balance){
      System.out.println("Insufficient
      funds. Withdrawal failed");
}
  else{

    balance -= amount;
```

```
        System.out.println("Withdrawal of $" +
                amount + " Successful.
                Updated
                balance: $" + balance);

        checkMinBalance();
    }
}

class SavAccount extends Account{
    double interestRate;

    public SavAccount(String customerName,
    long accountNumber, double balance,
    double interestRate)
    {
        super(customerName, accountNumber,
        "Savings", balance);
        this.interestRate = interestRate;
    }
    public void computeInterest
                                    (){
        double interest = balance * (int
                                    Rate/
                                    100);
        balance += interest;
        System.out.println("Interested
                            computed and
                            deposited: $" +
                            interest);
        displayBalance();
    }
}
```

```
public void withdraw (double amount)
  if (amount > balance)
     System.out.println ("Insufficient
     funds. Withdrawal failed");
  }
  else {
     balance = - amount;
     System.out.println ("Withdrawal of
        $" + amount + " successful.
                              updated
                              balance: $" +
                              balance);

     CheckMinBalance();
  }
}

class SavAccount extends Account {
double interestRate;
public savaccount { String customer
                        Name,
                    long accountNumber,
                    double balance,
                    double interestRate

  super (customerName, accountNumber,
     "Savings", balance);
     this.interestRate = interestRate;
  }
  public void computeInterest() {
     double interest = balance
                        (intRate/100);
     balance += interest;
```

```
System.out.println("interest computed and
                    deposited = $" +
                    interest);

displayBalance();

}
public void withdraw(double
                     amount){

if (amount > balance)
{
    System.out.println("Insufficient
                        funds- withdrawal
                        failed");

}
else{

balance -= amount;
System.out.println("Withdrawal
    of $" + amount + "Successful.
    updated balance: $" + balance);

}}}

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

```
CurAccount currentAccount =
        new CurAccount("J",
                123456789,
                1000, 18);

SavAccount SavingsAcount = new
                SavAccount(
                "D", 9845234,
                2000, 5);

    int choice;
    do {

    System.out.println("Select an option");
    System.out.println("1. Deposite");
    System.out.println("2. Display Balance");
    System.out.println("3. Compute Interest");
    System.out.println("4. Withdraw");
    System.out.println("5. Exit");
    System.out.println("Enter your
                choice");
    choice = Scannee.nextInt();

switch choice(){

Case 1:
    System.out.println("Enter amount
                to deposit");

    double depositAmount =
                Scannes.nextDouble();

    System.out.println("Select account
                (1. Current,
                2. Savings)");
```

```
int acctype = scanner.nextInt();
if (acctype == 1)
{
    currentAccount.displayBalance();
}
else if (acctype == 2)
{
    SavingsAccount.displayBalance();
}
else
{
    System.out.println("Invalid acc type");
}
break;
case 2:
    System.out.println("Select acc (1. Current, 2. Savings)");
    int acctype == 1)
{
    currentAccount.displayBalance();
}
else if (acctype == 2)
{
    SavingsAccount.DisplayBalance();
}
else {
    System.out.println("Invalid");
}
break;
Case 3:
    if(Saving Account instance of SavAccount)
```

```
{
    ((SavAccount) SavingsAccount).
                    computeInterest();
}
else{
    System.out.println("Invalid");
}
break;
case 4:
    System.out.println("Enter
                    account to be
                    withdrawed");
    double withdrawAmount =
                    Scanner.next
                    Double();
    System.out.println("SelectAcc
                    (1. Current,
                    2. Savings):");
    int accType = Scanner.nextInt();
    if (accType == 1)
{
    currentAccount.withdraw(with
                    drawAmt);
}
    else if (AccType == 2)
{
    SavingsAccount.withdraw(withdraw
                    Amt);
}
    else {
        System.out.println("Invalide");
}
break;
case 5:
```

```
        System.out.println("Exiting");
        break;
    default:
        System.out.println("Invalid");
    }
}
while (choice != 5)
    scanner.close();
}
```

output

```
Select an option
1. Deposit
2. Display Balance
3. Compute Interest
4. Withdraw
5. Exit
Enter your choice: 1
Enter amount to deposit: 10000
Select account (1. Current 2. Savings): 2
Deposit of $10000.0 successful.
```

A B

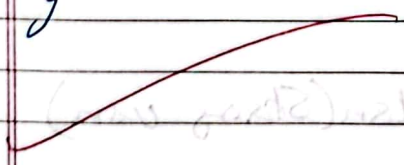
Pr 12/1/24

23/1/24

LAB-6

Internals

```
package CIE;

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

    public Internals() {
    }

    public void setInternalMarks(int[] internalMarks) {
        this.internalMarks = internalMarks;
    }

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

}
```



// Student

```
package CIE;

public class Student
{
    public String usn;
    public String name;
    public int sem;

    public Student()
    {
        this("", "", 0);
    }

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

    public void setUsn(String usn)
    {
        this.usn = usn;
    }
    public void setName(String name)
    {
        this.name = name;
    }
```

```
public void setSem(int sem)
{
    this.sem = sem;
}

public String getUsn()
{
    return usn;
}

public String getName()
{
    return name;
}

public int getSem()
{
    return sem;
}
}
```

External

```
package SEE;
import CIE.Student;
public class External extends Student{
 public int[] seeMarks = new int[5];
 public External()
 {
     this("", "", 0, new int[5]);
 }
 public External(String usn, String name,
                 int sem, int[] seeMarks)
 {
     super(usn, name, sem);
     this.seeMarks = seeMarks;
 }
 public void setSeeMarks(int[] seeMarks)
 {
     this.seeMarks = seeMarks;
 }
 public int[] getSeeMarks()
 {
     return seeMarks;
 }
}
```

Final Marks

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

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

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

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

    for(int i=0; i<n; i++)
    {
        students[i] = new Student();
        System.out.print("Enter USN for
                          Student " +
                          (i+1) + " : ");
        students[i].setUsn(scanner.next();

        System.out.print("Enter name for
                          Student " + (i+1)
                          + " : ");
        students[i].setName(scanner.
                            nextInt());
```

```
System.out.print("Enter semester for student
        + (i+1) + " : ");
Students[i].setSem(scanner.nextInt());

internals[i] = new Internals();

internals[i].setInternalMarks(inputMarks
        withValidation
        ("internal",
        i, scanner,
        50));

externals[i].setSecMarks(inputMarks
        withValidation
        ("external",
        scanner, 0, 100);

int[] finalMarks = new int[5];
for(int j=0; j<5; j++)
{
    finalMarks[j] = internals[i].get
        InternalMarks()[j] +
        externals[i].
        getSecMarks()[j]/2;
}

System.out.println("Student " + (i+1) +
        " Final Marks: " +
        finalMarks[0] + " " +
        finalMarks[1] + " " +
        finalMarks[2] + " " +
        finalMarks[3] + " " +
        finalMarks[4]);
```

```
}
Scanner.close();
}

private static int[] inputMarks(String type, int studentIndex,
                                Scanner scanner,
                                int min, int max) {
    {
        int[] marks = new int[5];
        System.out.println("Enter " + type +
                           " marks of student "
                           + (studentIndex + 1) + ":");
        for (int i = 0; i < 5; i++) {
            int marks;
            do {
                System.out.println("Subject "
                                   + (i + 1) + ":");

                marks = scanner.nextInt();
                if (mark < 0 || mark > max)
                {

                System.out.println("Invalid Input");
                }
                while (mark < 0 || mark > max);
                marks[i] = marks;
            }
        return marks;
```

Output:

Enter the number of Students: 1
Enter USN for Student 1: 101
Enter name for Student 1: Nithya
Enter semester for Student 1: 2
Enter internal marks for Student 1:
Subject 1: 45
Subject 2: 44
Subject 3: 43
Subject 4: 42
Subject 5, 41
Enter external marks for Student 1:
Subject 1: 89
Subject 2: 88
Subject 3: 87
Subject 4: 88
Subject 5: 78
Student 1 Final Marks: 89, 88, 87, 88, 78

23/1/24

30/1/24

LAB PROGRAM - 7 (User Defined Exception)

```
import java.util.Scanner

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

class Father
{
    protected int fatherAge;

    public Father(int age) throws WrongAge
    {
        fatherAge = age;
        if(fatherAge < 0)
        {
            throw new WrongAge(message:
                "Father's age
                cannot be
                negative");
        }
    }
}

class Son extends Father
{
    private int sonAge;
    public Son(int fatherAge, int sonAge)
            throws WrongAge
```

```
{
    super(fatherAge);
    this.sonAge = sonAge;
    if (sonAge <= 0)
    {
        throw new WrongAge(message:
                "Son's age
                cannot be
                negative or
                zero");
    }

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

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

            System.out.println("Enter father's
                                age:");
            int fatherAge = scanner.nextInt();
```

```
System.out.println("Enter son's age:");
int sonAge = scanner.nextInt();
Son son = new Son(fatherAge, sonAge);
System.out.println("Father's age:" + fatherAge);
System.out.println("Son's age:" + sonAge);
}

catch (WrongAge e)
{
    System.out.println("Exception caught:" + e);
    System.out.println("Exception caught:" +
                       e.getMessage());
}
catch (Exception e)
{
    System.out.println("Error:" + e);
    System.out.println("Error:" + e.getMessage());
}
finally
{
    scanner.close();
}
}
}
```

Output:

```
Enter father's age: 35
Enter son's age: 12
Father's age: 35
Son's age: 12
```

Enter father's age : 67
Enter son's age : 78
Exception caught: WrongAge: Son's age cannot be greater than or equal to father's age

Ans
30.01.24

6/2/24

LAB-08

```
class displaythread extends thread
{
    private String Message;
    private int interval;
    private boolean running = true;
    public Displaythread (String message, int interval)
    {
        this.message = message;
        this.interval = interval;
    }
    public void run ()
    {
        while (running)
        {
            System.out.println(message);
        try{
            Thread.sleep(interval);
        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }
    }

public void stopThread ()
{
    running = false;
}
}
```

```
public class Thread example
{
    public static void main (String args[])
    {
        DisplayThread bMSThread = new
            DisplayThread ("BMS college
                of engineering", 1000);
        DisplayThread cseThread = new Display
            Thread ("CSE", 2000);
        bMSThread.start();
        CSEThread.start();
        System.out.println("press Enter to
                stop the threads...");
        try{
            System.in.read();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        bMSThread.stopThread();
        CSEThread.stopThread();
    }
}
```

output

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

6/2/24

LAB-10

```
class Q
{
  int n;
  boolean valueSet = false;
  Synchroniged 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;
  }
  Synchroniged void put(int n)
  {
    while (valueSet)
    {
      try {
        System.out.println("\n Producer
                            waiting\n");
        wait();
      }
```

```
catch (InterruptedException e)
{
    System.out.println("Interrupted Exception
                        caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("\n Intimate consumer
                    \n\n");
notify();
}
}
class Produce implements Runnable
{
    Q q;
    Produce(Q, q)
    {
        this.q = q;
        new Thread(this, "Produce").start();
    }
    public void run()
    {
        int i = 0;
        while (i<15)
        {
            q.put(i++);
        }
    }
}
class Consumer implements Runnable
{
    Q q;
    Consumer(Q q){
```

```
        this.q = q;
        new Thread(this, "consumer").start();
    }
    public void run()
    {
        int i=0;
        while(i<15)
        {
            int r = q.get();
            System.out.println("Consumed: " + r);
            i++;
        }
    }
}
class PCfixed{
    public static void main(String args[])
    {
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Ctrl+C to stop");
    }
}
```

Output

```
Put: 0
Got: 0
Put: 1
Got: 1
Put: 2
Got: 2
Put: 3
Got: 3
```

13/2/24

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.
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");
}
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
Back in main thread
Racing Thread trying to call A.last()
Inside A.last
Back in other thread
```

Rw
13/2/2024

20/2/24

LAB-9

Source Code:

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

class SwingDemo{
    SwingDemo()
    {
        JFrame jfrm = new JFrame(" Divides App");
        jfrm.setSize(275, 150);
        jfrm.setLayout(new FlowLayout());
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel jlab = new JLabel("Enter the divident and divides:");
        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);
```

```
jfm.add(button);
jfm.add(alab);
jfm.add(blab);
jfm.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.get
                                    Text());

            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!");
}
}};
jfm.setVisible(true);
}

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

OUTPUT:

Enter the divider and divident:

40	2

Calculate A=40 B=2 Ans=20

AWT Functions:

- JFrame: It is a class in Java that is part of the swing library, which is used for creating graphical user interfaces in Java application.
- SetSize(); setSize() is a method which is used with components such as JFrame, JPanel etc to set their size.
- setLayout(): It is a method which is used to set the layout manager for a container, such as JFrame or any other container component.
- setDefaultCloseOperation(): It is a method which is used to specify default close operation for a JFrame.
- JEditable JLabel: It is a class which is used to display non-editable text or image of a GUI.