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:

NITHIN K PATIL

1BM22CS184

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	Complete scanned Observation Book	12/12/2023 - 20/02/2024
2	Lab 1	12/12/2023
3	Lab 2	19/12/2023
4	Lab 3	26/12/2023
5	Lab 4	02/01/2024
6	Lab 5	09/01/2024
7	Lab 6	16/01/2024
8	Lab 7	23/01/2024
9	Lab 8	30/01/2024
10	Lab 9	06/02/2024
11	Lab 10	20/02/2024

```
r1 = (-b) / (2*a);
System.out.println("Roots are equal and real");
System.out.println("Root1 = root2 = "+r1);
}
else if (d>0)
{
r1 = ((-b) + (Math.sqrt(d))) / (double)(2*a);
r2 = ((-b) - (Math.sqrt(d))) / (double)(2*a);
System.out.println("Roots are real and distinct");
System.out.println("Root1 = "+r1+ "Root2 = "+r2);
}
else if (d<0)
{
System.out.println("Roots are imaginary");
r1 = (-b) / (2*a);
r2 = Math.sqrt(-d) / (2*a);
System.out.println("Root1 = "+r1+ " +i" +r2);
System.out.println("Root2 = "+r1+ " -i" +r2);
}
}
}
class QuadraticMain
{
public static void main(String args[])
{
Quadratic q = new Quadratic();
q.getd();
q.compute();
}
}
```

11/12/23

Output

(a) Nithin K Patil 1BM22CS184

Enter values of a, b, c

1

2

1

roots are real and equal

root1 = root2 = -1

(b) Enter values of a, b, c

1

-3

2

roots are real and distinct

root 1 = 2.0

root 2 = 1.0

(c) Enter values of a, b, c

1

1

2

roots are imginary

(d) Enter values of a, b, c

0

4

2

roots cannot be formed.

12/12/23

week 2

(2) Calculate SGPA Program

```
import java.util.Scanner;

class Subject{
 int subjectmarks, Credits, grade;
}
class student {
  String name, USN;
 double Sgpa;
Scanner S;
Subject[] Subject;

Student() {
   int i;
   Subject = new Subject[9];
   for(i=0; i<9; i++){
      Subject[i] = new Subject();
   }
    S = new Scanner(System.in);
  }


void getStudentdetails(){
System.out.println("enter the name the student:");
 name = S.nextLine();
System.out.println("enter USN:");
 USN = S.nextLine();
}

void getmarks(){
  for(int i=0; i<9; i++){
   System.out.println("enter the marks for "+(i+1)+"th Subject");
   Subject[i].Subjectmarks = S.nextInt();
  System.out.println("enter the credits for "+(i+1)+"th Subject");
   Subject[i].subjectmarks = S.nextInt();
```

```
if (Subject[i].SubjectMarks >= 90) {
        Subject[i].grade = 10;
}
else if (Subject[i].SubjectMarks >= 80) {
        Subject[i].grade = 9;
}
else if (Subject[i].SubjectMarks >= 70) {
        Subject[i].grade = 8;
}
else if (Subject[i].SubjectMarks >= 60) {
        Subject[i].grade = 7;
}
else if (Subject[i].SubjectMarks >= 50) {
        Subject[i].grade = 6;
}
else if (Subject[i].SubjectMarks >= 40) {
        Subject[i].grade = 5;
}
else if (Subject[i].SubjectMarks < 40) {
        Subject[i].grade = 0;
}
}
}
void Sgpa() {
    int creditsxgrades = 0;
    int totalCredits = 0;
    for (int i = 0; i < 9; i++) {
        creditsxgrade += (Subject[i].grade * Subject[i].credits);
        totalcredits += Subject[i].credits;
    }
    System.out.println("Total
    Sgpa = creditsxgrade / totalcredits;
}
```

```
void display() {
  System.out.println("name:" + name);
System.out.println("USN:" + usn);
System.out.println("Sgpa:" + Sgpa);
 }
}

class main{
   public static void main(String[] args) {
      Student S1 = new Student();
    S1.getStudentdetails();
    S1.getmarks();
    S1.Sgpa();
    S1.display();
}
}
```

output

```
enter name:
Nithin K Patil
enter USN:
1BM 22CS184
enter marks and credits
 95
  4
 97
  4
 97
  3
 96
  3
 90
  3
 98
  1
```

98

1

92

1

Name: Nithin K Patil

USN : 1BM22CS286

SGPA : 9.55

1/12/23

26/12/23

week 3

① Book details code

```
import java.util.Scanner;

class book {
  String name;
  String author;
  int numpages;
  int price;

Book (String name, String author, int price, int numPages) {
  this.name = name;
  this.author = author;
  this.price = price;
this.numPages = numpages;
}
public String toString() {
String bookDetails = "Book name:" + this.name + "\n" + "Author name:"
                     + this.author + "\n" + "price:" + this.price + "\n" +
                     "Number of pages: " + this.numPages + "\n";
return bookDetails;
}
}

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

System.out.println ("enter the no of books:");
  int n = s.nextInt();

Book[] books = new book[n];
```

```
for (int i=0; i<n; i++){
System.out.println("enter details for book "+(i+1));
System.out.println("Name:");
String name = S.next();
System.out.println("Author:");
String author = S.next();
System.out.println("Price:");
int price = S.nextInt();
System.out.println("no of pages:");
int numPages = S.nextInt();

books[i] = new book(name, author, price, numPages);
}
System.out.println("\n Details of the books:");
for(int i=0; i<n; i++){
    System.out.println("Book"+(i+1)+":\n"+book[i].toString());
}
}
}
```

Output

```
Enter the no of books: 2
Enter details for book 1
Name: Nithin
Author: dude
price: 349
no of pages: 800
Enter details for book 2
Name: Manny
Author: dudez
price: 479
no of pages: 700
```

Details of the books:

Book 1:

Book name: Nithin

author name: dude

price : 349

Number of pages : 800

Book 2:

Book name : Manoj

author name : dude2

price : 479

Number of pages: 700

21/1/2024

2/1/24

Week-4

(1) abstract class program for printing area of rectangle, triangle, circle.

```
import java.util.Scanner;

abstract class Shape {
    double dim1, dim2;
Shape (double a, double b) {
    dim1 = a;
    dim2 = b;
}
abstract void printArea();
}

class Rectangle extends Shape {
    Rectangle (double a, double b) {
        super (a, b);
    }
    void printArea () {
    System.out.println ("Area of Rectangle =" + (dim1 * dim2);
    }
}

class Triangle extends Shape {
    Triangle (double a, double b) {
        super (a, b);
    }
    void printArea () {
        System.out.println ("area of triangle is" +
                            (dim1 * dim2) / 2);
    }
}
```

```
class circle extends Shape {
    Triangle (double a, double b) {
        super (a, b);
    }
    void printArea() {
        System.out.println("area of circle = " + pi
```

```
    Circle (double a, double b) {
        super (a, b);
    }
    final double pi = 3.14159;
    void printArea() {
    System.out.println("area of circle = " + pi * Math.pow(dim1,2));
    }
}
```

```
class main {
    Public State void main (String args[]) {
    Scanner sc = new Scanner (System.in);
    double a, b;

    System.out.println("enter side of rectangle :");
    a = sc.nextDouble();
    b = sc.nextDouble();
    Rectangle r = new Rectangle (a, b);

    System.out.println("enter height and base length of triangle :");
        a = sc.nextDouble();
        b = sc.nextDouble();
    Triangle t = new Triangle (a, b);

System.out.println("enter radius of circle :");
    a = sc.nextDouble;
    b = sc.nextDouble;
```

```
Circle C = new Circle (a, 1);

Shape S;
S = r;
S. printArea ();
S = t;
S. printArea ();
S = C;
S. printArea ();
}
}
```

output

```
Enter sides of rectangle :
1
2
Enter height and base of Triangle :
5
2
Enter radius of circle :
5
Area of rectangle = 2.0
Area of Triangle = 5.0
Area of circle = 78.53975
```

Rm 21/2024

dlt

Lab-5

- Develop a java program to create a class bank that maintains two kinds of account for its customers, one called saving account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provided 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 tasks:

(a) Accept deposit from customer and update the balance

(b) Display the balance

(c) compute and deposit interest

(d) permit withdrawal and update the balance

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

```
Import Java.util.Scanner;

class Account {
    private String customerName;
    private String accountNumber;
    private String accountType;
    protected double balance;

Public Account (String customerName, String accountNumber, String
                accountType, double balance) {
    this.customerName = customerName;
    this.accountNumber = accountNumber;
    this.accountType = accountType;
    this.balance = balance;
    }

Public void deposit (double amount) {
    balance += amount;
System.out.println ("Amount deposited successfully. current balance:
                    balance");
    }

Public void displayBalance() {
System.out.println ("Account type: " + accountType);
System.out.println ("customerName: " + customerName);
System.out.println ("Account number: " + AccountNumber);
System.out.println ("current Balance: " + balance);
    }
}

class SavingsAccount extends Account {
    private double interestRate;

Public SavingsAccount (String customerName, String accountNumber,
                       double balance, double interestRate) {
```

```
super (customerName, accountNumber, "Savings", balance);
    this.interestRate = interestRate;
}

Public void computeAndDepositInterest() {
    double interest = balance * interestRate / 100;
    deposit (interest);
    System.out.println ("interest computed and deposited, current balance:" +
                        balance);
}

Public void withdraw (double amount) {
    if (balance >= amount) {
        balance -= amount;
    System.out.println ("Insufficient funds. Withdrawal failed.");
}
}
}

class CurrentAccount extends Account {
    private static final double Service_charge = 250;
    private double minimumBalance;

public current

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

    System.out.println ("enter cust name");
    String custName = sc.nextLine();

    System.out.println ("acc no");
    String accNum = s.nextLine();
```

```
Syslin.out.println("enter initial balance");
double initialBalance = S.nextDouble();

SavingsAccount SavingAccount = new SavingsAccount(custName, accNum,
  initialBalance, 2.5);
SavingsAccount
CurrentAccount CurrentAcount = new CurrentAccount(custName, accNum,
  initialBalance, 500);

int choice;
do{
  Syslin.out.println("\n Select an option");
  Syslin.out.println("1. Deposit to Savings \n 2. compute and
  deposit interest for saving \n 3. withdraw from savings \n 4.
  deposit to current account \n 5. withdraw from current \n
  6. Display balance \n 7. exit");
Syslin.out.println("Enter your choice");
  choice = Scanner.nextInt();

Switch (choice){
  case 1:
    syslin.out.println("enter amount to deposit");
  double savingsDepositAmount = S.nextDouble();
  SavingAccount.deposit(savingsDepositAmount);
  break;
case 2:
  SavingsAccount.computeAndDepositInterest();
  break;
case 3:
  Syslin.out.println("enter amount to withdraw");
  double savingsDepositWithdrawAmount = S.nextDouble();
  ~~Current~~ deposited(~~current~~DepositAccount);
break;
  SavingsAccount withdraw(savingsWithdrawAmount);
  break;
```

```
case 4:
System.out.println("enter the amount to deposit");
 double currentDepositAmount = S.nextDouble();
 currentAccount.deposit(currentDepositAmount);
 break;
case 5:
System.out.println("enter the amount to withdraw from current
                    account");
double currentwithdrawAmount = S.nextDouble();
currentAccount.withdraw(currentwithdrawAmount);
 break;
case 6:
System.out.println("\n Savings account Details;");
 SavingsAccount.displayBalance();

case 7:
System.out.println("Thank you");
 break;
default:
  System.out.println("invalid choice");
}
```

Lab - 6

Student.Java

```
Package CIE;
public class student
{
  public static name;
  public string USN;
  public int Sem;

  public Student (String name, String USN, int Sem)
  {
     this.name = name;
     this.USN = USN;
     this.Sem = Sem;
  }
}
```

Externals.Java

```
Package SEE;
import CIE.Student;

public class Externals extends Student
{
   public int [] SeeMarks;
   public Externals (String name, String USN, int Sem, int [] seemarks)
   {
      Super (name, USN, Sem);
      this.SeeMarks = SeeMarks;
   }
}
```

Internals.Java

```
Package CIE;

Public class Internals extends CIE.Student
{
  Public int [] InternalMarks;

Public Internals (String name, String USN, int sem,
    int [] InternalMarks)
{
   Super (name, USN, Sem);
   this. InternalMarks = InternalMarks;
 }
}
```

FinalMarks

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

Public class FinalMarks
{
 public static void main (String [] args)
 {
   Scanner S1 = new Scanner (System.in);

   System.out.println (" enter no of Studs");
   int n = S1. nextInt();

String [] names = new String[n];
String [] USN = new String[n];
int [] Sem = new int [n] [5];
```

```
int [] [] seemarks = new int [n] [5];

for (int i=0 ; i<n ; i++)
{
    System.out.println("inter details for Student " + (i+1) + ":");
    System.out.print("Name :");
    names[i] = sc.next();
    System.out.print("USN :");
    usn[i] = sc.next();
    System.out.println("SEM :");
    sem = sc.nextInt
    in[i]= new.Internals(name, USN, Sem);
    ex[i] = new.Externals();

    System.out.println("enter 5 internals marks");
      for ( i =0 ; i < 5 ; i++) {
          ex[i] =emarks[i] = sc.nextInt ; }
      }

for (int i=0; i<n; i++) {
System.out.println("Details of Student" + (i+1) ":");
System.out.println("Name :" in[i].name + "\t" + "USN" +
    in[i].USN + "\t" + "Sem:" + in[i].sem);

    System.out.println("external mark:");

    for (int j=0 ; j<5 ; j++) {
        System.out.println("Subject" + (i+1) + ":" +
        in[i] = (marks[j] + "\t")}
        System.out.println("external marks:");
        for (int j=0 ; j<5 ; j++) {
        System.out.println("Subject" + (i+1) + in[i].imarks[j]
                + ex[i].marks[j]); } } }
```

Output

enter no of Students : 1
enter name of student 1 : Nidhi
enter USN of 1 : 12345
enter sem of 1 : 2

enter the internals marks of student 1 : 5 subjects
45
34
43
28
48

enter external marks of student 1 in 5 subjects
48
49
47
49
47

Details of student 1 :
Name : Nidhi USN : 12345 Sem : 2
internal marks :
Subject 1 : 45 Subject 2 : 34 Subject 3 : 43
Subject 4 : 28 Subject 5 : 48
external marks
Subject 1 : 48 Subject 2 : 49 Subject 3 : 47
Subject 4 : 49 Subject 5 : 49
Total marks
Subject 1 : 93
subject 2 : 83
subject 3 : 90
Subject 4 : 77
Subject 5 : 97

30/11/24

Lab 07

Father - Son Age

```
import java.util.Scanner;
class WrongAge extends Exception {
 WrongAge (String error) {
 System.out.println(error); }
   }

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

class Son extends Father {
    int age;
  Son (int age, int s_age) throws WrongAge {
   super (age);
    if (s_age >= age)
   throw new WrongAge ("Son's age cannot be
     greater than Father's Age");
      this.age = age; }

class main {
  public static void main (String args[]) {
  Scanner sc = new Scanner (System.in);
    try {
      System.out.println("enter Father's age");
      int ~~f~~ age = sc.nextInt();
      System.out.println("enter Son's age");
```

```
int s_age = sc.nextInt();
Son a = new Son(f_age, s_age);
System.out.println("Father's age:" + f_age);
System.out.println("son's age:" + s_age);
}
catch (WrongAge e){
System.out.println("Wrong age entered");}

catch (Exception ee) {
System.out.println("unexpected error:" + ee);}
}
}
```

Output

enter the father's age: 53
enter the son's age: 15
Father's age : 53
Son's age : 15

enter Father's age: 5
enter son's age: 23
Son's age: cannot be greater than Father's age
wrong age entered
enter the father's age: -15
enter the son's age: 15
Father's age cannot be negative
wrong age entered

30/1/2024

6/2/24

6/2/24

Program 10
IPC

```
class Q {
int n;
boolean valueSet = false;
synchronized int get() {

while (!valueSet)
try {

System.out.println("Interrupted \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;
}

synchronized void put(int n) {
while (valueSet)

try {
system.out.println("\n producer waiting \n");
wait();
}
Catch (InterruptedException e) {
System.out.println("Exception caught");
}
```

```
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("In 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 < 15) {
int r = q.get();
System.out.println ("consumed:" + r);
i++;
}
}
}
class Main {
public static void main (String args[]) {
Q q = new Q();
new producer (q);
new consumer (q);
System.out.println ("press control-c to stop.");
}
}
```

Output

Press control - C to stop.

Put : 0
Intimate Consumer

Produced waiting

Got : 0
Intimate producer

Put : 1
Intimate consumer

Produced waiting

Consumed : 0
Got : 1

Intimate producer

Consumed : 1
Got : 2

Intimate Consumer

Produced waiting

Got : 2

Intimate producer

Consumed : 2

consumer waiting
put : 3
Initiate consumer
producer waiting
Out - 3
consumed : 3

Ans
6/2/2024

6/2/24

Lab - 8

Program - 8

```
class DisplayMessageThread extends Thread {

private final String message;
private final long interval;

DisplayMessageThread (String message, long interval) {

this.message = message;
this.interval = interval;
}

Public void run () {

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

public class TwoThread {
   public static void main (String[] args) {
DisplayMessageThread thread1 = new DisplayMessageThread
                                ("BMSCE", 10000);
DisplayMessageThread thread2 = new DisplayMessageThread ("CSE", 2000);
```

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

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

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

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

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

Output

BMSCE
CSE
CSE
CSE
CSE
BMSCE
CSE
CSE
CSE
CSE
Main thread exiting.
Thread 1 interrupted.
Thread 2 interrupted.

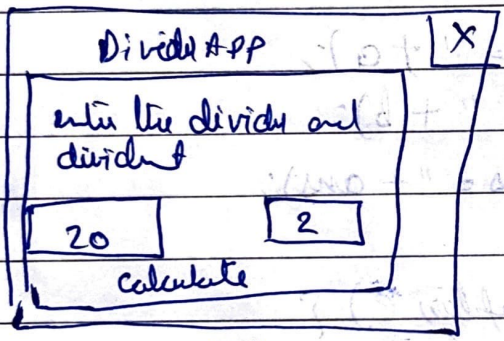
Program 9
Divider APP

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

class SwingDemo {

JFrame jfrm = new JFrame ("Divider App");
 jfrm.setSize(275, 150);
 jfrm.setDefaultLayout (new FlowLayout());
 jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 JLabel jlab = new JLabel ("enter the divide and
                            divident:");

 JTextField ajtf = new JTextField(8);
 JTextField bjtf = new JTextField(8);

JButton button = new JButton ("calculate");

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

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

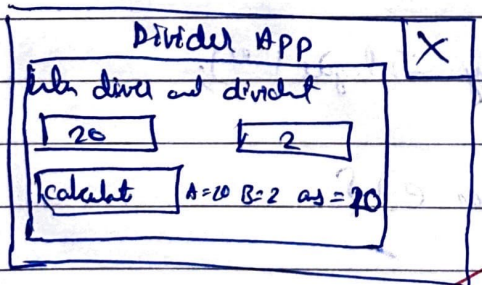
```
ActionListener l = new ActionListener(){
public void actionPerformed(ActionEvent evt){
  System.out.println("action event from a text field");
  }
};
ajtf.addActionListener(l);
bjtf.addActionListener(l);
button.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent evt){
    try{
      int a = Integer.parseInt(ajtf.getText());
      int b = Integer.parseInt(bjtf.getText());
      int ans = a/b;
    alab.setText("\nA=" +a);
    blab.setText("\nB=" +b);
    anslab.setText("\nAns=" +ans);
  }
  catch(NumberFormatException e){
      alab.setText("");
      blab.setText("");
    anslab.setText("");
    res.setText("Enter only Integers!");
  }
  catch(ArithmeticException e){
      alab.setText("");
      blab.setText("");
    anslab.setText("");
    res.setText("B should be Non zero!");
  }
}
});
```

```
    f.frm.setVisible(true);
}
public static void main(String args[]){

SwingUtilities.invokeLater(new Runnable(){
    public void run(){
        new SwingDemo();
    }
});
}}
```

Output



After click



20/2/24

LAB-01 QUADRATIC EQUATION

Question

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

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&gt;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&lt;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();
}
}
```

LAB-02- STUDENT SGPA CALCULATION

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.

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<8;i++)
subjects[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&&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;
    }
}

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);}
}
```

LAB-03 BOOK PROGRAM

Question:

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.

CODE:

```
import java.util.Scanner;


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

    // Parameterized constructor
    Book(String name, String author, int price, int numPages) {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }

    // toString method to display book details
    public String toString() {
        String bookDetails = "Book name: " + this.name + "\n"
                + "Author name: " + this.author + "\n"
                + "Price: " + this.price + "\n"
                + "Number of pages: " + this.numPages + "\n";
        return bookDetails;
```

```
    }
}

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

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

        // Define array of Book objects
        Book[] books = new Book[n];

        // Input details for each book
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for Book " + (i + 1));
            System.out.print("Name: ");
            String name = s.next();
            System.out.print("Author: ");
            String author = s.next();
            System.out.print("Price: ");
            int price = s.nextInt();
            System.out.print("Number of pages: ");
            int numPages = s.nextInt();

            // Create a new Book object with the entered details
            books[i] = new Book(name, author, price, numPages);
        }
```

```
        // Display details of each book
        System.out.println("\nDetails of the books:");
        for (int i = 0; i < n; i++) {
            System.out.println("Book " + (i + 1) + ":\n" + books[i].toString());
        }
    }
}
```

LAB-04 SHAPE PROGRAM

Question:

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 contains only the method printArea( ) that prints the area of the given shape.

CODE:

```
import java.util.Scanner;
class InputScanner{
Scanner s= new Scanner(System.in);
double getInput(String prompt){
System.out.println(prompt);
return s.nextDouble();}}
abstract class Shape extends InputScanner{
double side1, side2;
abstract void area();}

class Rectangle extends Shape{
Rectangle(){
side1=getInput("Enter length of rectangle:");
side2=getInput("Enter breadth of rectangle:");}
void area(){
double area=side1*side2;
System.out.println("Area of the Rectangle ="+area);}}
class triangle extends Shape{
    triangle(){
    side1=getInput("Enter base of the triangle:");
    side2=getInput("Enter height of the triangle:");}
     void area(){
```

```
        double area=side1*side2/2;
System.out.println("Area of the triangle="+area);}}
class circle extends Shape{
circle(){
side1=getInput("Enter the radius of the circle:");}
void area(){
double area=Math.PI*side1*side1;
System.out.println("Area of the circle="+area);}}

class Main{
public static void main(String args[]){
Rectangle rectangle= new Rectangle();
triangle Triangle=new triangle();
circle Circle=new circle();

rectangle.area();
Triangle.area();
Circle.area();

}
}
```

LAB -05 BANK ACCOUNT(abstract classes)

Question

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.

CODE:

```
import java.util.Scanner;


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

    public Account(String customerName, long accountNumber, String accountType, 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("Account Balance: $" + balance);
```

```
    }
    public void withdraw(double amount) {
        System.out.println("Withdrawal not supported for this account type.");
    }
}

class CurAccount extends Account {
    double minBalance;
    double serviceCharge;

    public CurAccount(String customerName, long 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("Minimum balance not maintained. Service charge of $" +
 serviceCharge + " imposed.");
            displayBalance();
        }
    }
    @Override
    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 * (interestRate / 100);
        balance += interest;
        System.out.println("Interest computed and deposited: $" + interest);
        displayBalance();
    }
    @Override
    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);

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

        System.out.print("Enter initial amount: ");
        double initialAmount = scanner.nextDouble();

        System.out.print("Select account type (1. Current, 2. Savings): ");
        int accountTypeChoice = scanner.nextInt();

        Account userAccount = null;  // Declare userAccount outside the if-else blocks

        if (accountTypeChoice == 1) {
            System.out.print("Enter minimum balance for the current account: ");
            double minBalance = scanner.nextDouble();
            System.out.print("Enter service charge for the current account: ");
            double serviceCharge = scanner.nextDouble();

            userAccount = new CurAccount(customerName, System.currentTimeMillis(),
initialAmount, minBalance, serviceCharge);
        } else if (accountTypeChoice == 2) {
            System.out.print("Enter interest rate for the savings account: ");
            double interestRate = scanner.nextDouble();

            userAccount = new SavAccount(customerName, System.currentTimeMillis(),
```

```
initialAmount, interestRate);
        } else {
            System.out.println("Invalid account type choice. Exiting the program.");
            scanner.close();
            return;
        }

        int choice;
        do {
             System.out.println("\nSelect an option:");
            System.out.println("1. Deposit");
            System.out.println("2. Display Balance");
            System.out.println("3. Compute Interest (Savings Account only)");
            System.out.println("4. Withdraw");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter amount to deposit: ");
                    double depositAmount = scanner.nextDouble();
                    userAccount.deposit(depositAmount);
                    break;
                case 2:
                    userAccount.displayBalance();
                    break;
                case 3:
                    if (userAccount instanceof SavAccount) {
                        ((SavAccount) userAccount).computeInterest();
                    } else {
```

```
                System.out.println("Invalid option for current account.");
            }
            break;
        case 4:
            System.out.print("Enter amount to withdraw: ");
            double withdrawAmount = scanner.nextDouble();
            userAccount.withdraw(withdrawAmount);
            break;
        case 5:
            System.out.println("Exiting the program. Thank you!");
            break;
        default:
            System.out.println("Invalid choice. Please enter a valid option.");
      }
    } while (choice != 5);

    scanner.close();
  }
}
```

LAB-06 PACKAGES

Question:

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.

CODE:

```
package SEE;
public class External extends CIE.Student{
	public int emarks[]=new int[5];
}

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

package CIE;
public class Internals extends CIE.Student{
	public int imarks[]=new int[5];
	public Internals(){}
	public Internals(String name,int usn,int sem){
		super(name,usn,sem);
	}
}
```

```
import java.util.Scanner;
import CIE.Student;
import CIE.Internals;
import SEE.External;
class LabQ6{
	public static void main(String args[]){
		Scanner sc=new Scanner(System.in);
		String name;int usn,sem;
		System.out.println("Enter the number of students:");
		int n=sc.nextInt();
		Internals in[]=new Internals[n];
		External ex[]=new External[n];
		for(int i=0;i<n;i++){
			System.out.println("Enter the name of the student "+(i+1)+":");
			name=sc.next();
			System.out.println("Enter the USN of the student "+(i+1)+":");
			usn=sc.nextInt();
			System.out.println("Enter the sem of the student "+(i+1)+":");
			sem=sc.nextInt();
			in[i]=new Internals(name,usn,sem);
			ex[i]=new External();
			System.out.println("Enter the internal marks of the student in 5
			subjects:");
			for(int j=0;j<5;j++)
				in[i].imarks[j]=sc.nextInt();
			System.out.println("Enter the external marks of the student in 5
			subjects:");
			for(int j=0;j<5;j++)
				ex[i].emarks[j]=sc.nextInt();
		}
```

```
        for(int i=0;i<n;i++){
                System.out.println("Details of student "+(i+1)+":");
                System.out.println("Name:"+in[i].name+"\t"+"USN:"+in[i].usn+"\t"+"Sem:"+i
                n[i].sem);
                System.out.println("Internal marks:");
                for(int j=0;j<5;j++)
                        System.out.print("Subject "+(i+1)+":"+in[i].imarks[j]+"\t");
                System.out.println("External marks:");
                for(int j=0;j<5;j++)
                        System.out.print("Subject "+(j+1)+":"+ex[i].emarks[j]+"\t");
                System.out.println("\nTotal marks:");
                for(int j=0;j<5;j++)
                        System.out.println("Subject"+(j+1)+":"+(in[i].imarks[j]+ex[i].emarks[j])
                );
        }
}}
```

LAB -07 EXCEPTIONS (FATHER-SON)

Question:

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.

CODE:

```
import java.util.Scanner;
class WrongAge extends Exception{
	WrongAge(String error){
		System.out.println(error);
	}
}

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

class Son extends Father{
	int age;
	Son(int age,int s_age) throws WrongAge{
		super(age);
		if(s_age>=age)
			throw new WrongAge("Son'age cannot be greater than Father's age");
		this.age=s_age;
```

```
        }
}

class LabQ7{
        public static void main(String args[]){
                Scanner sc=new Scanner(System.in);
                try{
                        System.out.println("Enter the Father's age:");
                        int f_age=sc.nextInt();
                        System.out.println("Enter the Son's age:");
                        int s_age=sc.nextInt();
                        Son a=new Son(f_age,s_age);
                        System.out.println("Father's age:"+f_age);
                        System.out.println("Son's age:"+s_age);
                }
                catch(WrongAge e){
                        System.out.println("Wrong Age entered");}
                catch(Exception ee){
                        System.out.println("Unexpected error :"+ee);}
                }
}
```

LAB-08 THREADS

Question:

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.

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 ThreadExample {
    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();
    }
}
```

LAB-09 CREATE USER INTERFACE

Question:

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.

CODE:

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

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

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

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

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

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


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

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

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

```
                int ans = a/b;
                alab.setText("\nA = " + a);
                blab.setText("\nB = " + b);
                anslab.setText("\nAns = "+ ans);
            }
            catch(NumberFormatException e){
                alab.setText("");
                blab.setText("");
                anslab.setText("");
                err.setText("Enter Only Integers!");
            }
            catch(ArithmeticException e){
                alab.setText("");

                blab.setText("");
                anslab.setText("");
                err.setText("B should be NON zero!");
            }
        }
    });
    jfrm.setVisible(true);
  }
  public static void main(String args[]){
    // create frame on event dispatching thread
    SwingUtilities.invokeLater(new Runnable(){
      public void run(){
        new SwingDemo();
      }
    });
  }
}
```

LAB-10 IPC AND DEADLOCK

IPC CODE:

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

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

class Producer implements Runnable {
        Q q;
        Producer(Q q) {
                this.q = q;
                new Thread(this, "Producer").start();
        }
        public void run() {
                int i = 0;
                while(i<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("Press Control-C to stop.");
        }
}
```

DEADLOCK code:

```
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("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();
}
}
```