LAB-2 Program Source Code

(Q) A java program to calculate the SGPA of the student with the given conditions.

```
import java.util.Scanner;

class Student {
    private String usn;
    private String name;
    private int[] credits;
    private int[] marks;

    public Student() {
        usn = "";
        name = "";
    }

    public void acceptDetails() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter USN: ");
        usn = scanner.nextLine();

        System.out.print("Enter Name: ");
        name = scanner.nextLine();

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

```
        credits = new int[numSubjects];
        marks = new int[numSubjects];

        for (int i = 0; i < numSubjects; i++) {
            System.out.print("Enter credits for subject " + (i + 1) + ": ");
            credits[i] = scanner.nextInt();

            System.out.print("Enter marks for subject " + (i + 1) + ": ");
            marks[i] = scanner.nextInt();
        }
    }

    public void displayDetails() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Subject-wise details:");
        for (int i = 0; i < credits.length; i++) {
            System.out.println("Subject " + (i + 1) + " - Credits: " + credits[i] + ", Marks: " + marks[i]);
        }
    }

    public double calculateSGPA() {
        double totalCredits = 0;
        double totalGradePoints = 0;

        for (int i = 0; i < credits.length; i++) {
            totalCredits += credits[i];
            totalGradePoints += calculateGradePoints(marks[i]) * credits[i];
```

```
            }

            return totalGradePoints / totalCredits;
        }

        private double calculateGradePoints(int marks) {
            if (marks >= 90) {
                return 10.0;
            } else if (marks >= 80) {
                return 9.0;
            } else if (marks >= 70) {
                return 8.0;
            } else if (marks >= 60) {
                return 7.0;
            } else if (marks >= 50) {
                return 6.0;
            } else if (marks >= 40) {
                return 5.0;
            } else {
                return 0.0;
            }
        }
}

public class Main {
    public static void main(String[] args) {
        Student student = new Student();
        student.acceptDetails();
        student.displayDetails();
```

```
            System.out.println("SGPA: " + student.calculateSGPA());
        }
}
```

LAB-3 Program Source Code

(Q) A java program to create a class Book and implement the given scenario.

```
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 String getName(){
return name;
}
```

```
public void setName(String name){

this.name = name;

}


public String getAuthor(){

return author;

}


public void setAuthor(String author){

this.author = author;

}


public double getPrice(){

return price;

}


public void setPrice(double price){

this.price = price;

}


public int getNumPages(){

return numPages;

}


public void setNumPages(int numPages){

this.numPages = numPages;

}


public String toString() {
```

```
return "Book Details - Nane: " + name + ", Author: " + author + ", Price: $" + price + ", Number of Pages: " + numPages;
}
}

public class Main{
public static void main(String[] args){
int n=3;

Book[] books = new Book[n];

books[0] = new Book("The Catcher","J.D",15.99,224);
books[1] = new Book("To Kill","Lee",12.50,336);
books[2] = new Book("1984","Orwell",9.99,328);

System.out.println("Rishabh, 1BM22CS221");

for(int i =0;i<n;i++){
System.out.println(books[i].toString());
System.out.println();
}
}
}
```

LAB-4 Program Source Code

(Q) A java program to create a class Shape and implement the given scenario.

```
abstract class Shape{
protected int dim1;
protected int dim2;

public Shape(int dim1, int dim2){
this.dim1 = dim1;
this.dim2 = dim2;
}

public abstract void printArea();

}

class Rectangle extends Shape{
public Rectangle(int len, int wid){
super(len,wid);
}

public void printArea(){
int area = dim1 * dim2;
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 * dim1 * dim2;
System.out.println("Area of Triangle:" + area);
}

}
class Circle extends Shape{
public Circle(int rad){
super(rad,0);
}

public void printArea(){
double area = Math.PI * dim1 * dim1;
System.out.println("Area of Circle:" + area);
}
}

public class Main{
public static void main(String[] args){
Rectangle rectangle = new Rectangle(5,10);
Triangle triangle = new Triangle(4,6);
Circle circle = new Circle(7);

rectangle.printArea();
triangle.printArea();
circle.printArea();

}
```

}

LAB-5 Program Source Code

(Q) A java program to create a class Bank and implement the given scenario.

```
import java.util.Scanner;

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

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

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

```
class SavingsAccount extends Account {

double interestRate;


public SavingsAccount(String customerName, int accountNumber, String accountType, double balance, double interestRate){

super(customerName,accountNumber, accountType, balance);

this.interestRate = interestRate;

}


public void computeInterest(){

balance += (balance * interestRate) / 100;

System.out.println("Interest rate added");

}


public void withdraw(double amount){

if (amount <= balance){

balance -= amount;

System.out.println("Withdrawal Successful");

}

else{

System.out.println("Insufficient Fund");

}

}

}



class CurrentAccount extends Account{

double minBalance;

double serviceCharge;
```

```
public CurrentAccount(String customerName, int accountNumber, String accountType, double balance ,double minBalance,
double serviceCharge){

super(customerName,accountNumber, accountType, balance);


this.minBalance = minBalance;

this.serviceCharge = serviceCharge;

}


public void withdraw(double amount){

if (balance - amount >= minBalance){

balance -= amount;

System.out.println("Withdrawal Successful");

}

else{

System.out.println("Insufficient Fund, Service charge will be applied");

balance -= serviceCharge;

}

}

}


public class Bank{

public static void main(String args[]){

SavingsAccount savings = new SavingsAccount("Jhon",1001,"Savings",5000,5);

CurrentAccount current = new CurrentAccount("Alice",2001,"Current",8000,1000,50);


savings.computeInterest();

savings.displayBalance();

savings.withdraw(2000);

savings.displayBalance();
```

```
current.displayBalance();

current.withdraw(5000);

current.displayBalance();

}

}
```

LAB-6 Program Source Code

(Q) A java program to create classes and packages, and implement the given scenario.

```
import CIE.Internals;

import SEE.External;


public class CalculateFinalMarks{

                                                                                public static void
main(String[] args) {

                        Internals student1 = Internals("123","John Doe", 5, new int[]{85,90,78,92,88});

                        External student1 = External("189","Rock", 5, new int[]{75,60,98,92,68});


                        int finalMarks1 = calculateFinalMarks(student1.internalMarks);

                        int finalMarks2 = calculateFinalMarks(student.seeMarks);


                        System.out.println("Student 1 - Final Marks: " + finalMarks1);

                        System.out.println("Student 2 - Final Marks: " + finalMarks2);

}


      private static int calculateFinalMarks(int[] marks) {
```

```
            int totalMarks = 0;
            for (int mark : marks) {
                totalMarks += mark;
            }
            return totalMarks;
        }
}
```

```
//Student.java
package CIE;

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

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

```
// CIE/Internals.java
package CIE;

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

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

```
//External.java
package SEE;

import CIE.Student;

public class External extends Student {
    int[] seeMarks;

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

LAB-7 Program Source Code

(Q) A java program to create class Father and inherit Son, and implement the given scenario.

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

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

class InputScanner {
	static Scanner scanner = new Scanner(System.in);
}

class Father extends InputScanner {
	int fatherAge;

	Father() throws WrongAge {
		System.out.print("Enter Father's age: ");
		fatherAge = scanner.nextInt();

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

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

class Son extends Father {
```

```
    int sonAge;

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

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

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

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

LAB-8 Program Source Code

(Q) A java program to create two threads and implement the given scenario.

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

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

	//@override
	public void run(){
		while(true){
			System.out.println(message);
			try{
				Thread.sleep(sleepTime * 1000);
			}
			catch(InterruptedException e){
				e.printStackTrace();
			}
		}
	}
}
```

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

LAB-9 Program Source Code

(Q) A java program to create an Event Handling Function and implement the given scenario.

```
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 dividend:");


// 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

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());
                        if (b == 0) {
                                throw new ArithmeticException();
                        }
                        int ans = a / b;

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

```
                }
            }
        });

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

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

LAB-10 Program Source Code

(Q) A java program to create an IPR and implement the given scenario.

```
public class DeadlockExample {

    public static void main(String[] args) {
        final SharedResource sharedResource = new SharedResource();
```

```
        Thread process1 = new Thread(() -> {
            try {
                sharedResource.method1();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        });

        Thread process2 = new Thread(() -> {
            try {
                sharedResource.method2();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        });

        process1.start();
        process2.start();
    }
}

class SharedResource {
    private final Object lock1 = new Object();
    private final Object lock2 = new Object();

    public void method1() throws InterruptedException {
        synchronized (lock1) {
            System.out.println("Method 1 acquired lock1");
```

```
                Thread.sleep(1000);


                synchronized (lock2) {
                    System.out.println("Method 1 acquired lock2");
                    // Perform some task using both lock1 and lock2
                }
            }
        }

        public void method2() throws InterruptedException {
            synchronized (lock2) {
                System.out.println("Method 2 acquired lock2");
                Thread.sleep(1000);


                synchronized (lock1) {
                    System.out.println("Method 2 acquired lock1");
                    // Perform some task using both lock1 and lock2
                }
            }
        }
}
```