



SCHOOL OF  
COMPUTING

# LAB RECORD

23CSE111- Object Oriented Programming

*Submitted by*

CH.SC.U4CSE24127 – M DHEERAJ VAMSI KRISHNA

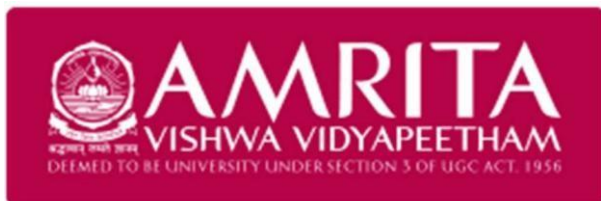
BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND  
ENGINEERING

AMRITA VISHWA  
VIDYAPEETHAM AMRITA  
SCHOOL OF COMPUTING  
CHENNAI

March - 2025



**SCHOOL OF  
COMPUTING**

**AMRITA VISHWA VIDYAPEETHAM  
AMRITA SCHOOL OF COMPUTING, CHENNAI**

## **BONAFIDE CERTIFICATE**

This is to certify that the Lab Record work for 23CSE111- Object Oriented Programming Subject submitted by **CH.SC.U4CSE24127 – medarametla dheeraj vamsi krishna** in “**Computer Science and Engineering**” is a Bonafide record of the work carried out under my guidance and supervision at Amrita School of Computing, Chennai.

This Lab examination held on 08/04/2025

Internal Examiner 1

Internal Examiner 2

## INDEX

S.NO	TITLE	PAGE. NO
	<b>UML DIAGRAM</b>	
<b>1.</b>	<b>TITLE OF UML DIAGRAM -1</b>	
	1.a) Use Case Diagram	4
	1.b) Class Diagram	5
	1.c) Sequence Diagram	6
	1.d) activity Diagram	7
	1.e) State Diagram	8
<b>2.</b>	<b>TITLE OF UML DIAGRAM -2</b>	
	2.a) Use Case Diagram	9
	2.b) Class Diagram	10
	2.c) Sequence Diagram	11
	2.d) activity Diagram	12
	2.e) State Diagram	13
<b>3.</b>	<b>BASIC JAVA PROGRAMS</b>	
	3.a) Armstrong Number	14
	3.b) automorphic number	15
	3.c) count digits	15
	3.d) diamond pattern	16
	3.e) GCD	17
	3.f) Number to words	17
	3.g) perfect number	18
	3.h) Reverse Number	18
	3.i) Reverse string	19
	3.j) sum even odd digits	19

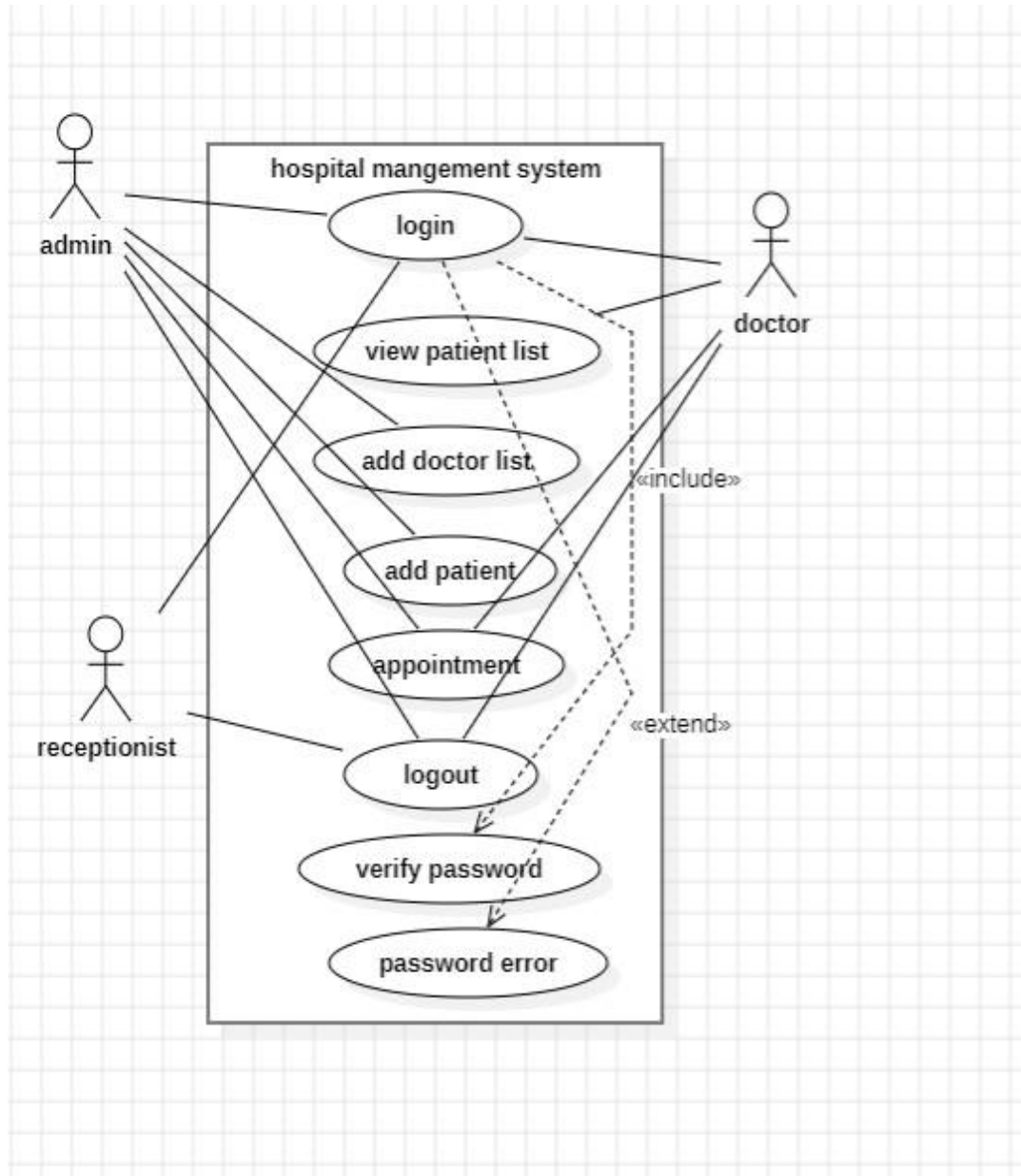
	<b>INHERITANCE</b>	23
4.	<b>SINGLE INHERITANCE PROGRAMS</b>	23
	4.a)Animal	24
	4.b)Person	25
5.	<b>MULTILEVEL INHERITANCE PROGRAMS</b>	25
	5.a)Animal	26
	5.b)Vehicle	27-28
6.	<b>HIERARCHICAL INHERITANCE PROGRAMS</b>	29
	6.a)Vehicle	29
	6.b)Shape(circle extends shape)	30
7.	<b>HYBRID INHERITANCE PROGRAMS</b>	31
	7.a) HybridInheritance_ChefWaiter	32
	7.b) HierarchicalInheritance1(Appliances)	34
	<b>POLYMORPHISM</b>	35
8.	<b>CONSTRUCTOR PROGRAMS</b>	35
	8.a) PatientCostructor	35
9.	<b>CONSTRUCTOR OVERLOADING PROGRAMS</b>	36
	9.a)Car-Main3	36
10.	<b>METHOD OVERLOADING PROGRAMS</b>	37
	10.a) DisplayMessage-Main4	37
	10.b) AreaCalculator-Main5	38
11.	<b>METHOD OVERRIDING PROGRAMS</b>	39
	11.a) Bank-Main6	39
	11.b) Animal-Main7	40
	<b>ABSTRACTION</b>	41
12.	<b>INTERFACE PROGRAMS</b>	41
	12.a) Vehicle-Main8	41
	12.b) Drawable-Main9	42
	12.c) Calculator-Main10	43
	12.d) Animal-Main11	44
13.	<b>ABSTRACT CLASS PROGRAMS</b>	45
	13.a) Bank-Main12	45
	13.b) Transport-Main13	46
	13.c) Payment-Main14	47
	13 .d) Person-Main15	47-48

	<b>ENCAPSULATION</b>	48
14.	<b>ENCAPSULATION PROGRAMS</b>	49
	14.a) BankAccount-Main16	50
	14.b) Temperature-Main17	51
	14.c) Marks-Main18	52
	14.d) User-Main19	53
15.	<b>PACKAGES PROGRAMS</b>	54
	15.a)User Defined Packages(contact management)	54-56
	15.b)User Defined Packages(task management;)	56-58
	15.c)Built – in Package(3 Packages)- ( MultiPackage)	59
	15.d)Built – in Package(3 Packages)- ( RandomSortSystemInfo)	60
16.	<b>EXCEPTION HANDLING PROGRAMS</b>	61
	16.a) CreateFile	62
	16.b) ReadFile	63
	16.c) DeleteFile	64
	16.d) UpdateFile	65-66
17.	<b>FILE HANDLING PROGRAMS</b>	67
	17.a) FileMetadata	68-69
	17.b) UpdateLineInFile	70-71
	17.c) CopyFile	71-72
	17.d) SearchInFile	73-74

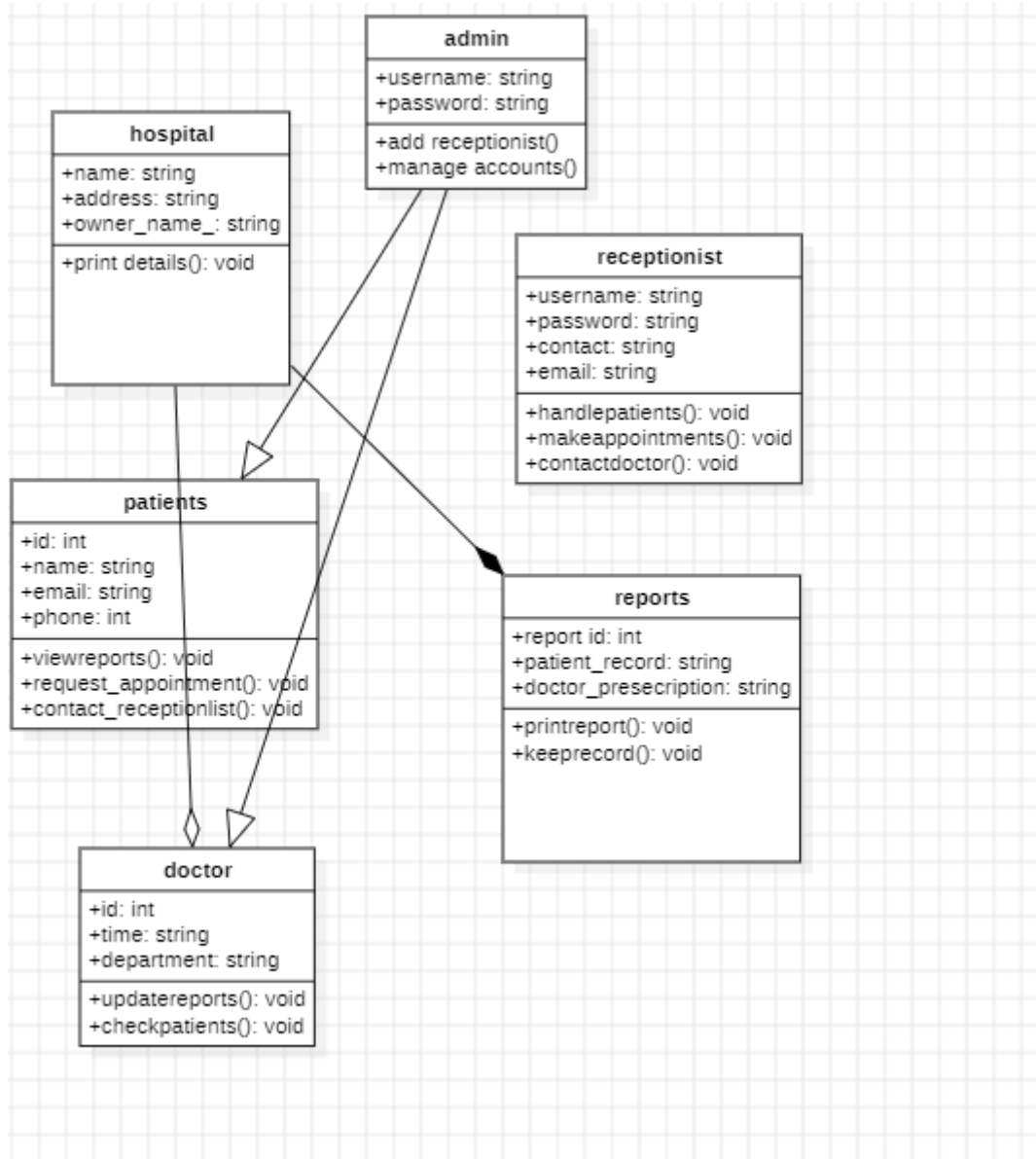
## UML DIAGRAMS

### 1) HOSPITAL MANAGEMENT SYSTEM

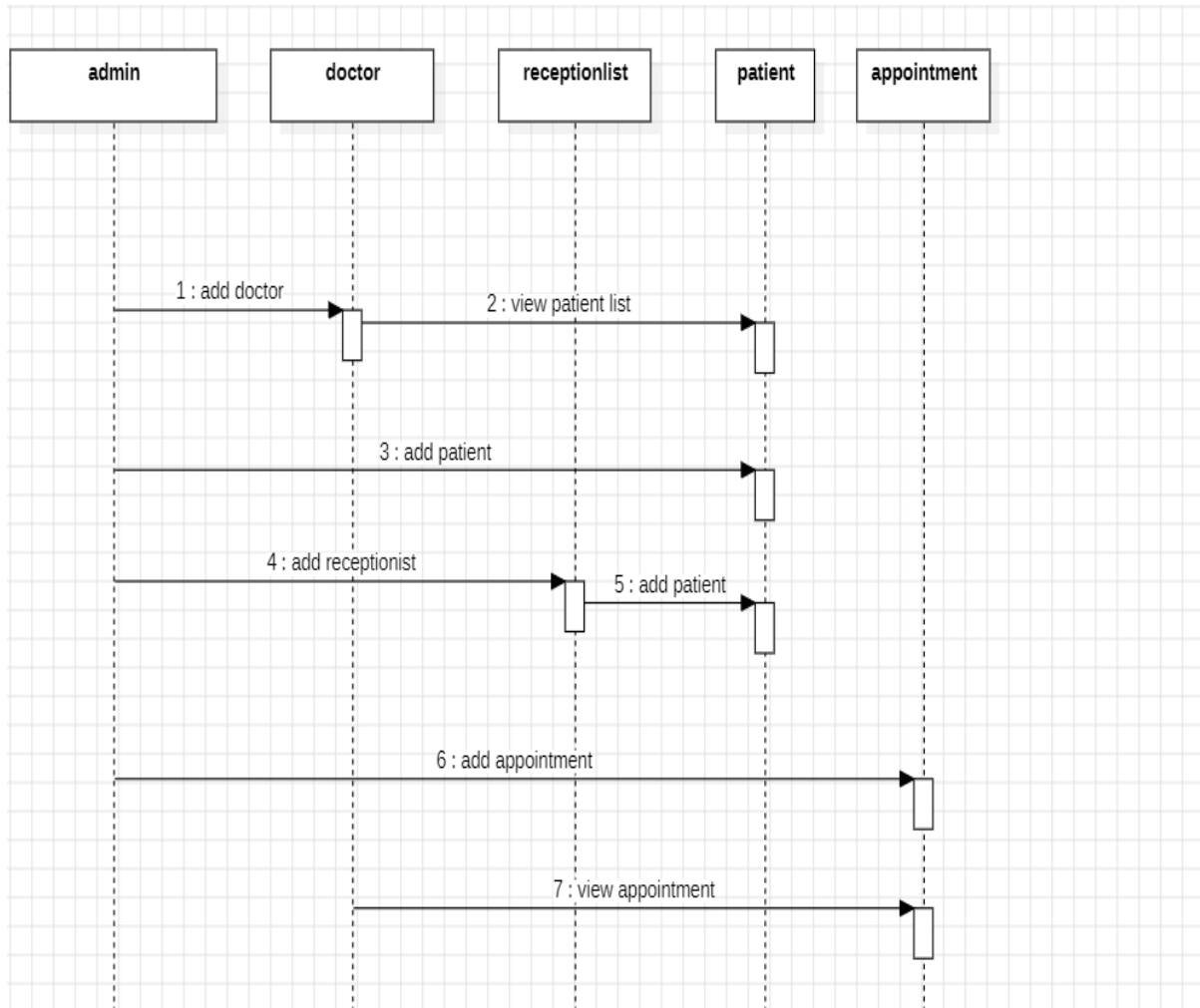
#### 1a) USE CASE DIAGARAM:



## 1B) class diagram:

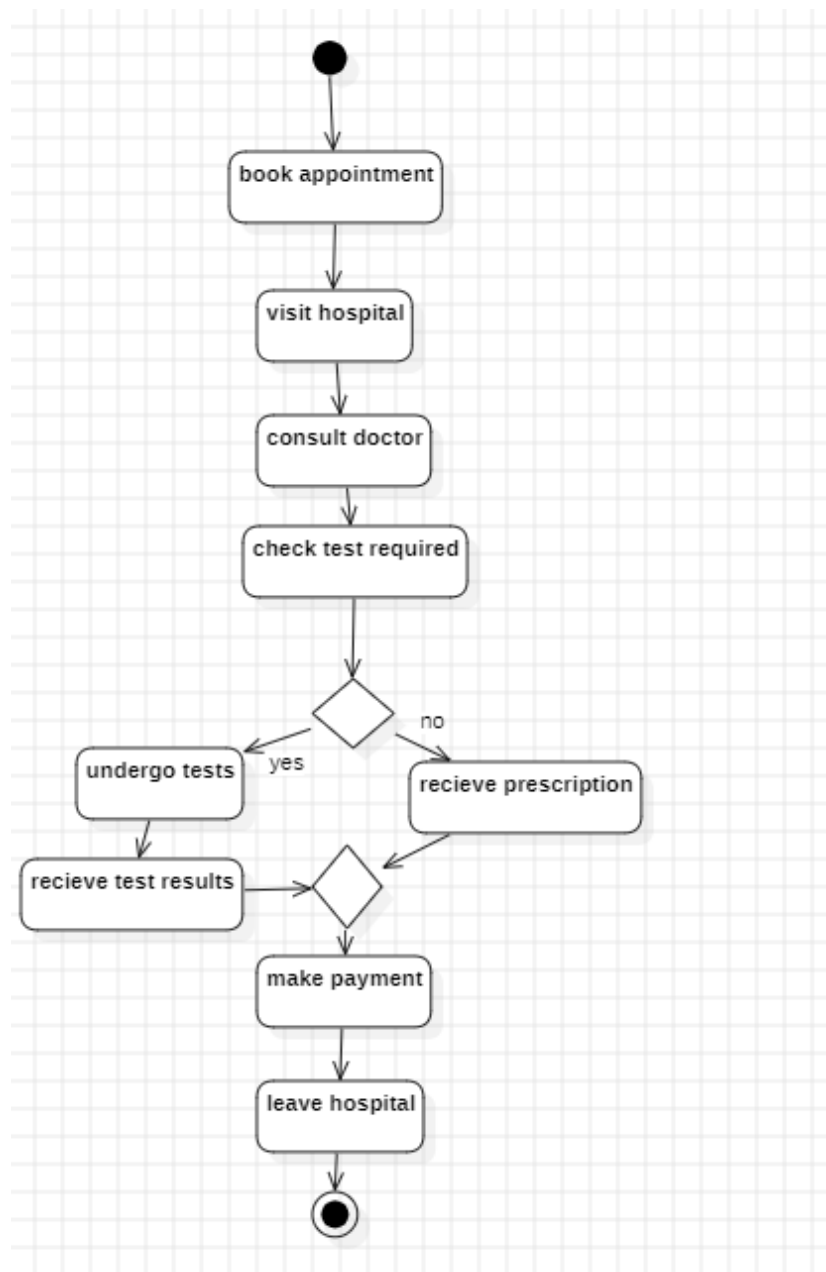


### 1C) sequence diagram:

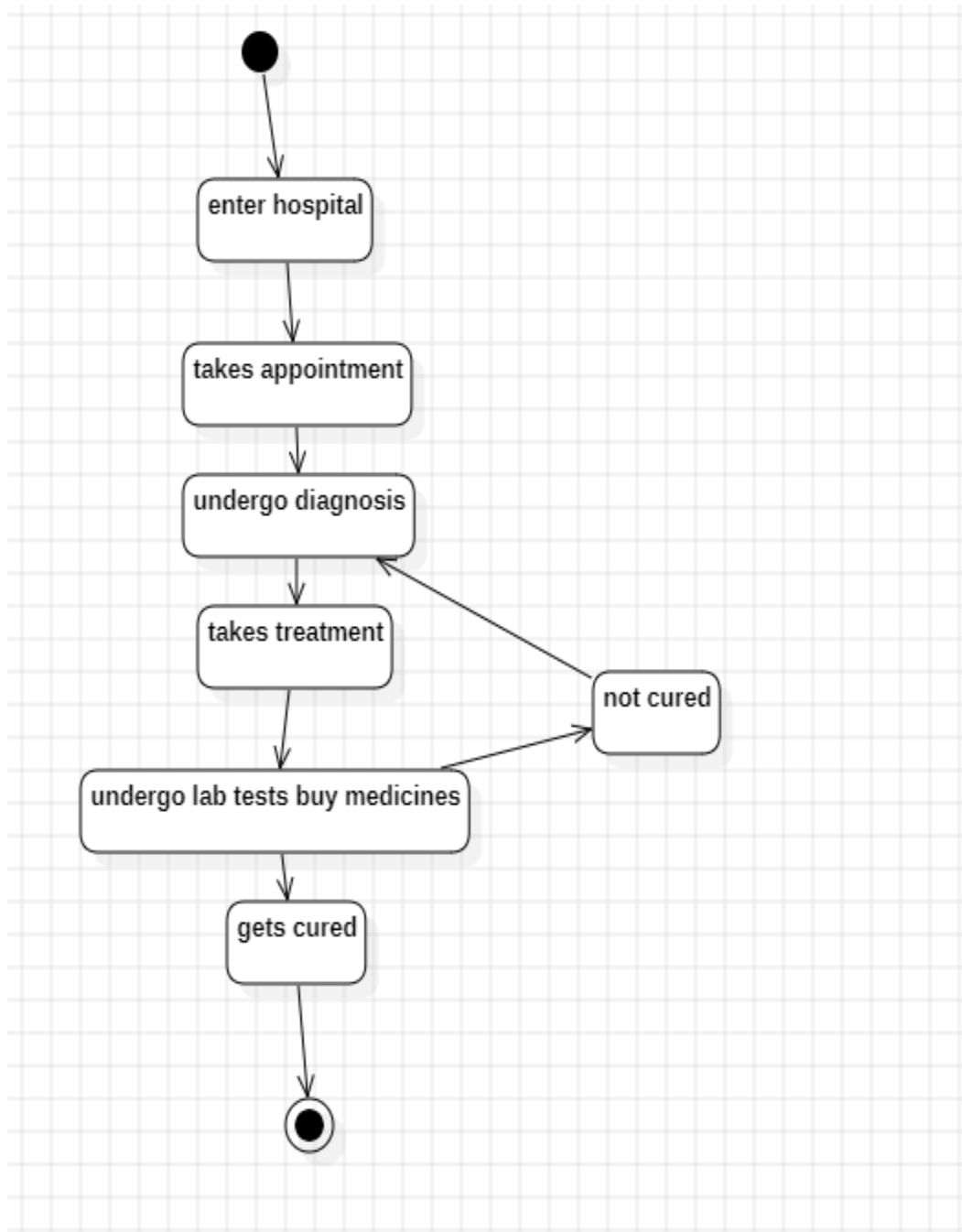




## 1D) activity diagram

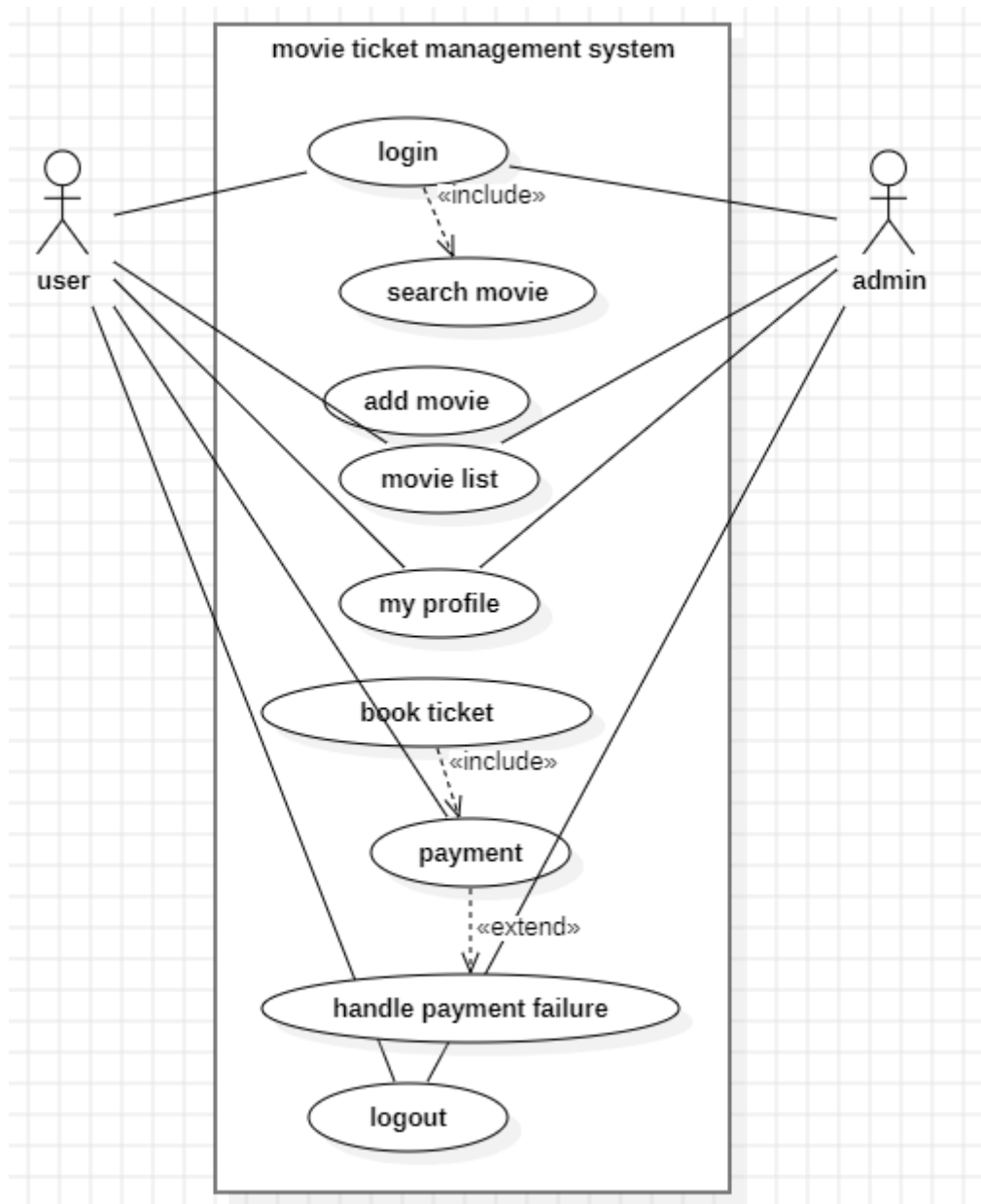


**1E) state diagram:**

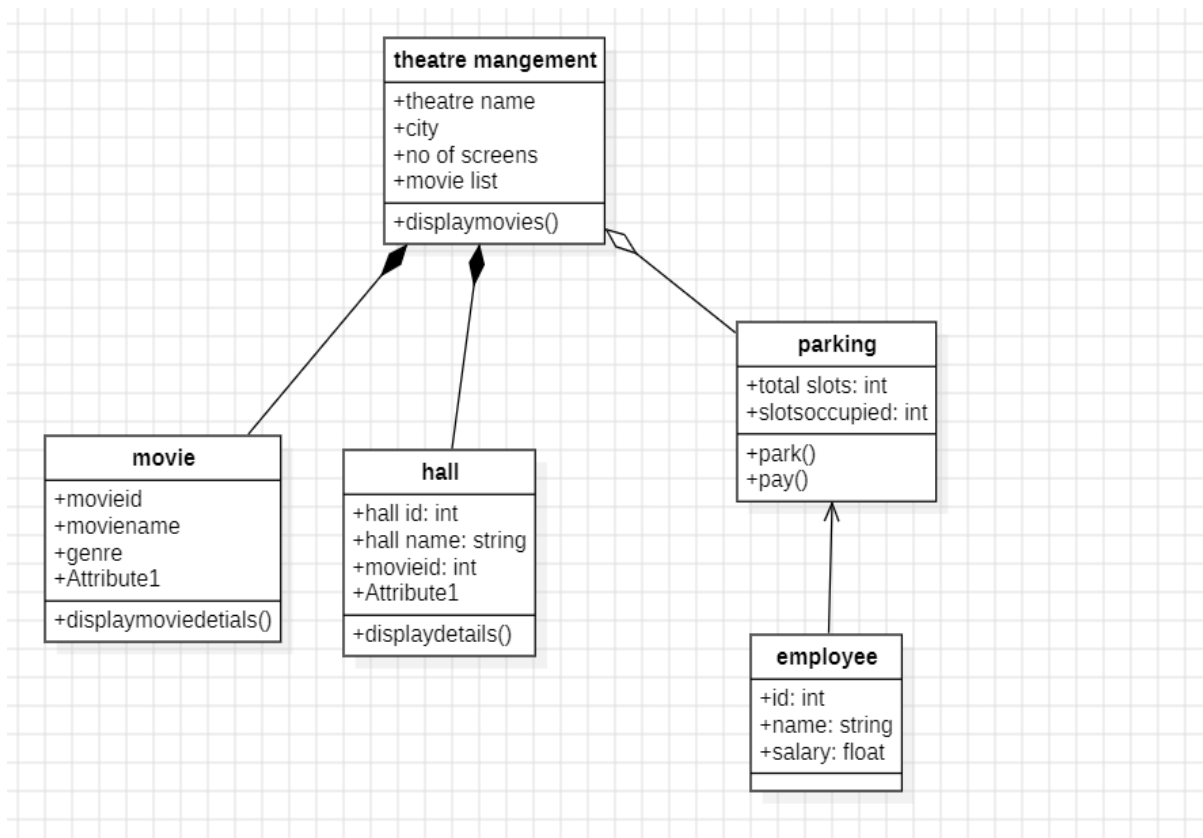


## 2) movie management system

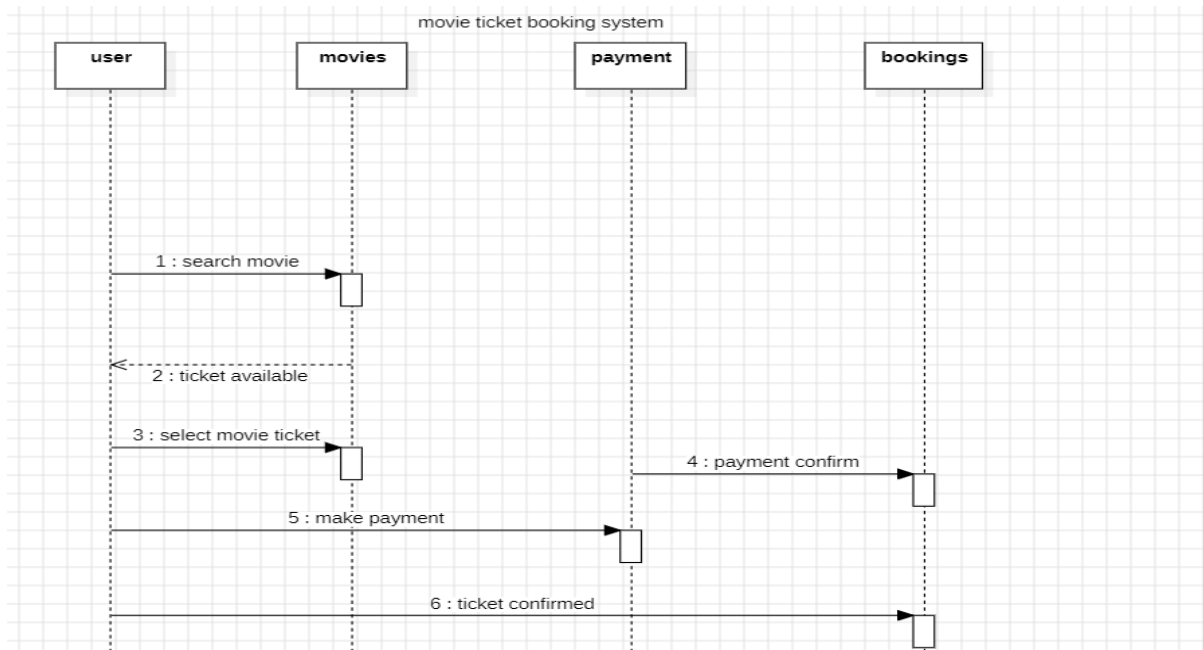
### 2A) use case diagram:



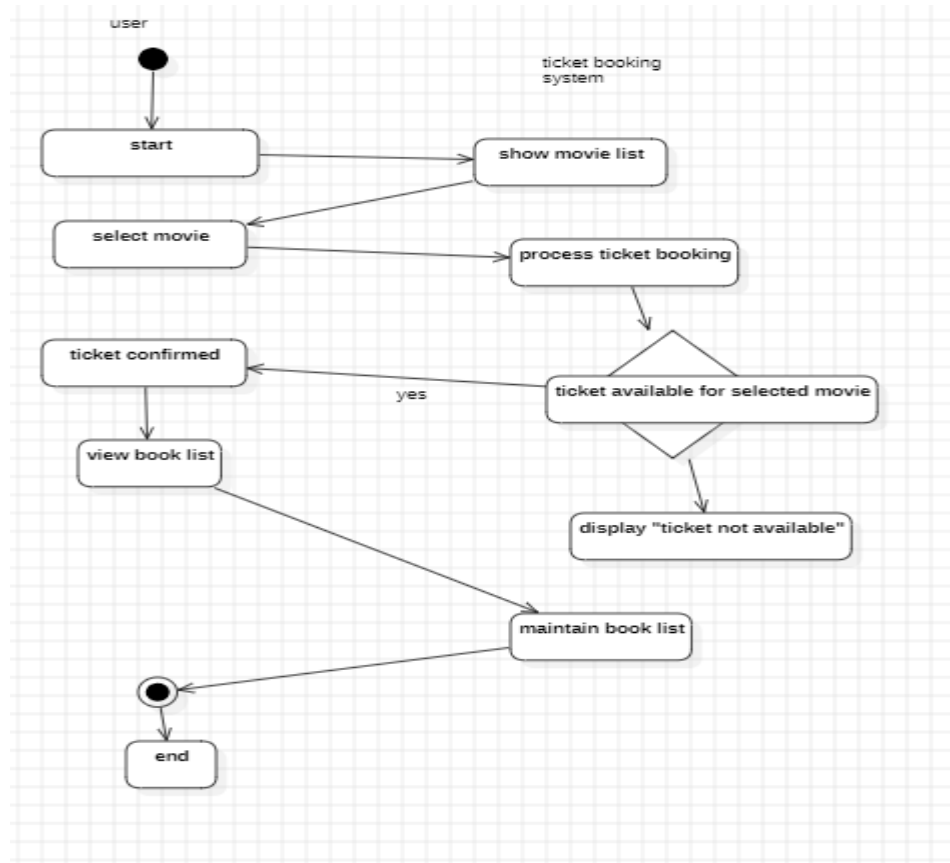
## 2B) class diagram:



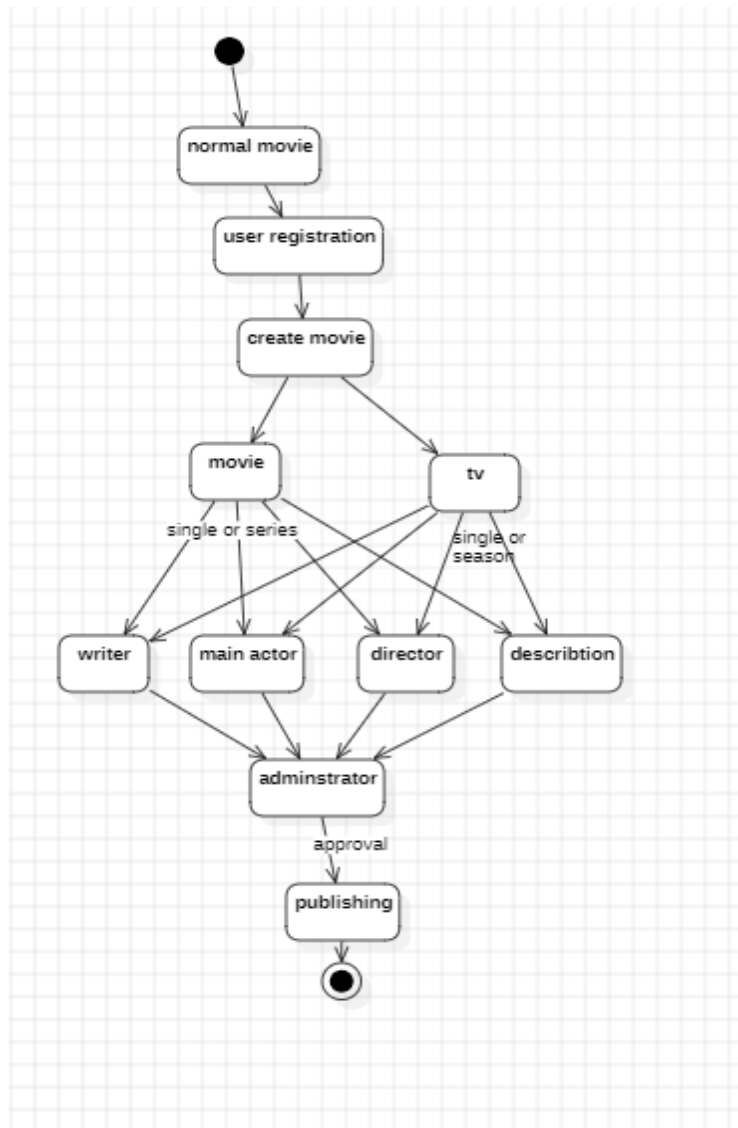
## 2C) sequence diagram:



## 2D) activity diagram:



## 2E) state diagram:



### 3) Basic java programs:

#### 3a) arm strong number:

```
public class ArmstrongNumber {  
    public static void main(String[] args) {  
        int num = 153, original = num, sum = 0;  
        while (num > 0) {  
            int digit = num % 10;  
            sum += digit * digit * digit;  
            num /= 10;  
        }  
        if (sum == original) System.out.println(original + " is an Armstrong  
number.");  
        else System.out.println(original + " is not an Armstrong number.");  
    }  
}
```

Output:

```
C:\Users\medar\OneDrive\Desktop\Basic Jva programmes>D:  
  
D:\>javac ArmstrongNumber.java  
  
D:\>java ArmstrongNumber.java  
153 is an Armstrong number.  
  
D:\>|
```



### 3b) Automorphic Number

```
public class AutomorphicNumber {  
    public static void main(String[] args) {  
        int num = 25, square = num * num;  
        String numStr = String.valueOf(num);  
        String squareStr = String.valueOf(square);  
        if (squareStr.endsWith(numStr)) System.out.println(num + " is an  
Automorphic Number.");  
        else System.out.println(num + " is not an Automorphic Number.");  
    }  
}
```

Output:

```
D:\>javac AutomorphicNumber.java  
  
D:\>java AutomorphicNumber.java  
25 is an Automorphic Number.
```

### 3c) Count Digits

```
public class CountDigits {  
    public static void main(String[] args) {  
        int num = 123456, count = 0;  
        while (num > 0) {  
            count++;  
            num /= 10;  
        }  
        System.out.println("Number of digits: " + count);  
    }  
}
```

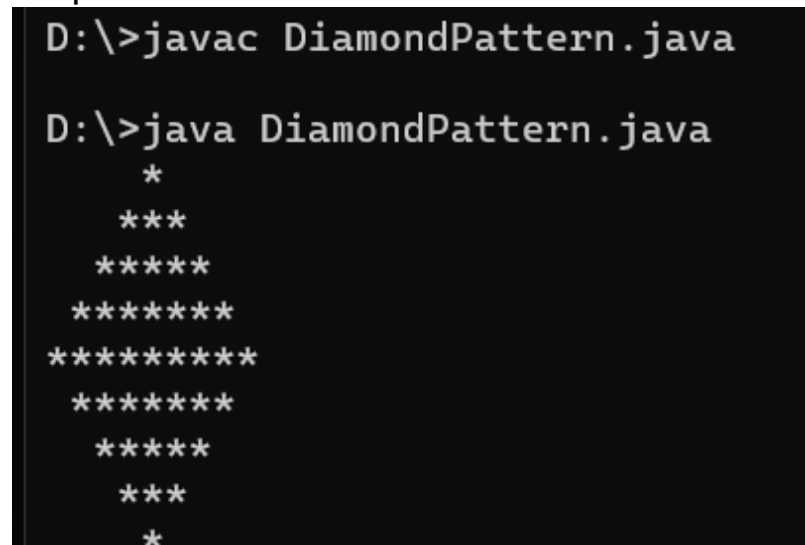
Output:

```
D:\>javac CountDigits.java  
  
D:\>java CountDigits.java  
Number of digits: 6
```

### 3d) Diamond Pattern

```
public class DiamondPattern {  
    public static void main(String[] args) {  
        int n = 5;  
        for (int i = 1; i <= n; i++) {  
            for (int j = n; j > i; j--) System.out.print(" ");  
            for (int k = 1; k <= (2 * i - 1); k++) System.out.print("*");  
            System.out.println();  
        }  
        for (int i = n - 1; i >= 1; i--) {  
            for (int j = n; j > i; j--) System.out.print(" ");  
            for (int k = 1; k <= (2 * i - 1); k++) System.out.print("*");  
            System.out.println();  
        }  
    }  
}
```

Output:

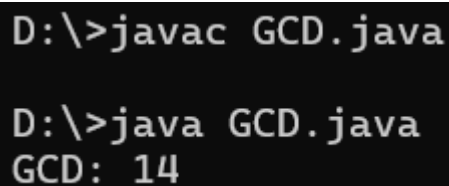


```
D:\>javac DiamondPattern.java  
  
D:\>java DiamondPattern.java  
    *  
   ***  
  *****  
 *****  
*****  
 *****  
  *****  
   ***  
    *
```

### 3e)GCD

```
public class GCD {  
    public static void main(String[] args) {  
        int a = 56, b = 98;  
        while (b != 0) {  
            int temp = b;  
            b = a % b;  
            a = temp;  
        }  
        System.out.println("GCD: " + a);  
    }  
}
```

Output:



```
D:\>javac GCD.java  
  
D:\>java GCD.java  
GCD: 14
```

### 3f) number to words

```
public class NumberToWords {  
    public static void main(String[] args) {  
        int num = 7;  
        String[] words = {"Zero", "One", "Two", "Three", "Four", "Five", "Six",  
"Seven", "Eight", "Nine"};  
        if (num >= 0 && num <= 9) System.out.println("Number in words: " +  
words[num]);  
        else System.out.println("Number out of range");  
    }  
}
```

Output:

```
D:\>javac NumberToWords.java
```

```
D:\>java NumberToWords.java  
Number in words: Seven
```

3g) perfect number

```
public class PerfectNumber {  
    public static void main(String[] args) {  
        int num = 28, sum = 0;  
        for (int i = 1; i < num; i++) {  
            if (num % i == 0) sum += i;  
        }  
        if (sum == num) System.out.println(num + " is a Perfect Number.");  
        else System.out.println(num + " is not a Perfect Number.");  
    }  
}
```

Output:

```
D:\>javac perfectNumber.java
```

```
D:\>java perfectNumber.java  
28 is a Perfect Number.
```

3h) reverse number

```
public class ReverseNumber {  
    public static void main(String[] args) {  
        int num = 12345, reversed = 0;  
        while (num != 0) {  
            int digit = num % 10;  
            reversed = reversed * 10 + digit;  
            num /= 10;  
        }  
        System.out.println("Reversed Number: " + reversed);  
    }  
}
```

Output:

```
D:\>javac ReverseNumber.java
```

```
D:\>java ReverseNumber.java  
Reversed Number: 54321
```

3i) reverse string

```
public class ReverseString {  
    public static void main(String[] args) {  
        String str = "Hello", reversed = "";  
        for (int i = str.length() - 1; i >= 0; i--) {  
            reversed += str.charAt(i);  
        }  
        System.out.println("Reversed String: " + reversed);  
    }  
}
```

Output:

```
D:\>javac ReverseString.java
```

```
D:\>java ReverseString.java  
Reversed String: olleH
```

3j) sum even odd digits

```
public class SumEvenOddDigits {  
    public static void main(String[] args) {  
        int num = 123456, evenSum = 0, oddSum = 0;  
        while (num > 0) {  
            int digit = num % 10;  
            if (digit % 2 == 0) evenSum += digit;  
            else oddSum += digit;  
            num /= 10;  
        }  
        System.out.println("Sum of even digits: " + evenSum);  
        System.out.println("Sum of odd digits: " + oddSum);  
    }  
}
```

Output:

```
D:\>java SumEvenOddDigits.java  
Sum of even digits: 12  
Sum of odd digits: 9
```

# Inheritance

## single inheritance

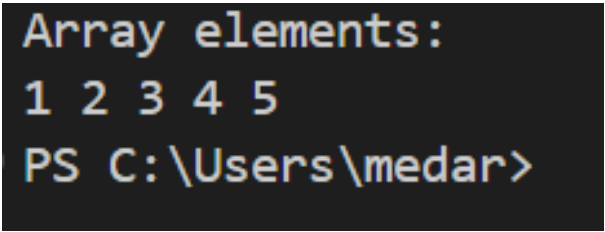
4a)

### Code:

```
class Animal {  
    void makeSound() {  
        System.out.println("Animal makes a sound");  
    }  
}
```

```
class Dog extends Animal {  
    void bark() {  
        System.out.println("Dog barks");  
    }  
  
    public static void main(String[] args) {  
        Dog myDog = new Dog();  
        myDog.makeSound();  
        myDog.bark();  
    }  
}
```

### Output:



```
Array elements:  
1 2 3 4 5  
PS C:\Users\medar>
```

4b)      single inheritance

**Code:**

```
class Person {  
    String name;  
  
    void setName(String n) {  
        name = n;  
    }  
  
    void greet() {  
        System.out.println("Hello, my name is " + name);  
    }  
}  
  
class Student extends Person {  
    void study() {  
        System.out.println(name + " is studying.");  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Student s = new Student();  
        s.setName("Alice");  
        s.greet();  
        s.study();  
    }  
}
```

**Output:**



```
Hello, my name is Alice  
Alice is studying.
```

## **5a) MULTILEVEL INHERITANCE**

**Code:**

```
class Animal {  
    void eat() {  
        System.out.println("This animal eats food.");  
    }  
}  
  
class Mammal extends Animal {  
    void walk() {  
        System.out.println("This mammal walks on land.");  
    }  
}  
  
class Dog extends Mammal {  
    void bark() {  
        System.out.println("Dog barks.");  
    }  
  
    public static void main(String[] args) {  
        Dog myDog = new Dog();  
        myDog.eat();  
        myDog.walk();  
        myDog.bark();  
    }  
}
```

**Output:**

```
Animal makes a sound  
Dog barks  
PS C:\Users\medar>
```

**5b)**

**Code:**

```
class Vehicle {  
    void start() {  
        System.out.println("Vehicle is starting...");  
    }  
}  
  
class Car extends Vehicle {  
    void drive() {  
        System.out.println("Car is driving...");  
    }  
}  
  
class SportsCar extends Car {  
    void turboBoost() {  
        System.out.println("SportsCar activates turbo boost!");  
    }  
}  
  
public class Main1 {  
    public static void main(String[] args) {  
        SportsCar sc = new SportsCar();  
    }  
}
```

```
        sc.start();  
        sc.drive();  
        sc.turboBoost();  
    }  
}
```

**Output:**

```
Vehicle is starting...  
Car is driving...  
SportsCar activates turbo boost!
```

**6a)**

**Hierarchical Inheritance**

```
class Vehicle {  
    void start() {  
        System.out.println("Vehicle is starting...");  
    }  
}
```

```
class Car extends Vehicle {  
    void drive() {  
        System.out.println("Car is driving...");  
    }  
}
```

```
class Bike extends Vehicle {  
    void ride() {  
        System.out.println("Bike is riding...");  
    }  
}
```

```
public static void main(String[] args) {  
    Car myCar = new Car();  
    myCar.start();  
    myCar.drive();  
  
    Bike myBike = new Bike();  
    myBike.start();  
    myBike.ride();  
}
```

**Output:**

```
Vehicle is starting...
Car is driving...
Vehicle is starting...
Bike is riding...
PS C:\Users\medar>
```

**6b)**

```
class Shape {
    void draw() {
        System.out.println("Drawing a shape");
    }
}
```

```
class Circle extends Shape {
    void area() {
        System.out.println("Area =  $\pi r^2$ ");
    }
}
```

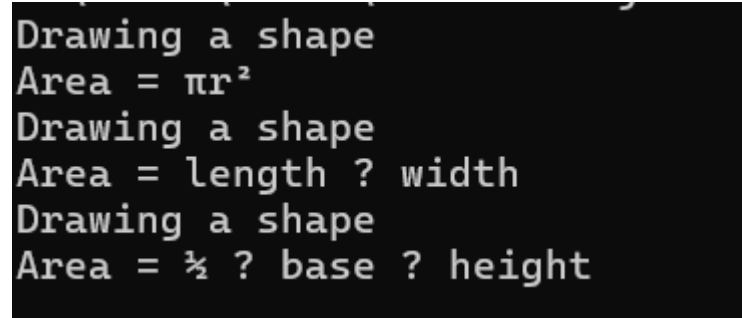
```
class Rectangle extends Shape {
    void area() {
        System.out.println("Area = length x width");
    }
}
```

```
class Triangle extends Shape {
    void area() {
        System.out.println("Area =  $\frac{1}{2} \times \text{base} \times \text{height}$ ");
    }
}
```

```
}
```

```
public class Main2 {  
    public static void main(String[] args) {  
        Circle circle = new Circle();  
        circle.draw();  
        circle.area();  
  
        Rectangle rect = new Rectangle();  
        rect.draw();  
        rect.area();  
  
        Triangle tri = new Triangle();  
        tri.draw();  
        tri.area();  
    }  
}
```

Output:



```
Drawing a shape  
Area =  $\pi r^2$   
Drawing a shape  
Area = length * width  
Drawing a shape  
Area =  $\frac{1}{2}$  * base * height
```

**7a)**

## HYBRID INHERITANCE

**Code:**

```
interface Chef {
```

```

    void prepareFood(String foodItem);
}

interface Waiter {
    void serveFood(String foodItem);
}

class Restaurant implements Chef, Waiter {
    @Override
    public void prepareFood(String foodItem) {
        System.out.println("Chef is preparing " + foodItem);
    }

    @Override
    public void serveFood(String foodItem) {
        System.out.println("Waiter is serving " + foodItem);
    }

    public void manageOrder(String foodItem) {
        prepareFood(foodItem);
        serveFood(foodItem);
    }
}

public class HybridInheritance_ChefWaiter {
    public static void main(String[] args) {
        Restaurant restaurant = new Restaurant();
        restaurant.manageOrder("Pasta");
    }
}

```

**Output:**

```

Chef is preparing Pasta
Waiter is serving Pasta

```

**7b)**

**Code:**

```

class Appliance {
    void consumeElectricity() {

```



```
        System.out.println("Appliance consumes electricity.");
    }
}
```

```
class WashingMachine extends Appliance {
    void washClothes() {
        System.out.println("Washing Machine is washing clothes.");
    }
}
```

```
class Refrigerator extends Appliance {
    void keepFoodFresh() {
        System.out.println("Refrigerator keeps food fresh.");
    }
}
```

```
public class HierarchicalInheritance1 {
    public static void main(String[] args) {
        WashingMachine wm = new WashingMachine();
        wm.consumeElectricity();
        wm.washClothes();

        Refrigerator fridge = new Refrigerator();
        fridge.consumeElectricity();
        fridge.keepFoodFresh();
    }
}
```

```
}
```

Output:

```
Appliance consumes electricity.  
Washing Machine is washing clothes.  
Appliance consumes electricity.  
Refrigerator keeps food fresh.
```

8a)

### **Polymorphism constructor**

#### **Code:**

```
class Patient {  
    String name;  
    int age;  
  
    Patient() {  
        this.name = "Unknown";  
        this.age = 0;  
    }  
  
    Patient(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
  
    void showInfo() {  
        System.out.println("Patient Name: " + name + ", Age: " + age);  
    }  
}
```

```

public class PatientCostructor{
    public static void main(String[] args) {
        java.util.Scanner sc = new java.util.Scanner(System.in);
        System.out.print("Enter patient name: ");
        String patientName = sc.nextLine();
        System.out.print("Enter patient age: ");
        int patientAge = sc.nextInt();
        Patient p1 = new Patient();
        Patient p2 = new Patient(patientName, patientAge);
        p1.showInfo();
        p2.showInfo();
        sc.close();
    }
}

```

**Output:**

```

Enter patient name: dheeraj
Enter patient age: 18
Patient Name: Unknown, Age: 0
Patient Name: dheeraj, Age: 18

```

9a)

**POLYMORPHISM constructor overloading**

**Code:**

```

class Car {
    String brand;
    int speed;
}

```

```
Car() {  
    brand = "Unknown";  
    speed = 0;  
    System.out.println("Car brand not specified.");  
}  
  
Car(String b) {  
    brand = b;  
    speed = 0;  
    System.out.println("Car Brand: " + brand);  
}  
  
Car(String b, int s) {  
    brand = b;  
    speed = s;  
    System.out.println("Car Brand: " + brand + ", Speed: " + speed + " km/h");  
}  
}  
  
public class Main3 {  
    public static void main(String[] args) {  
        Car car1 = new Car();  
        Car car2 = new Car("Toyota");  
  
        Car car3 = new Car("BMW", 200);  
    }  
}
```

**Output:**

```
Car brand not specified.  
Car Brand: Toyota  
Car Brand: BMW, Speed: 200 km/h
```

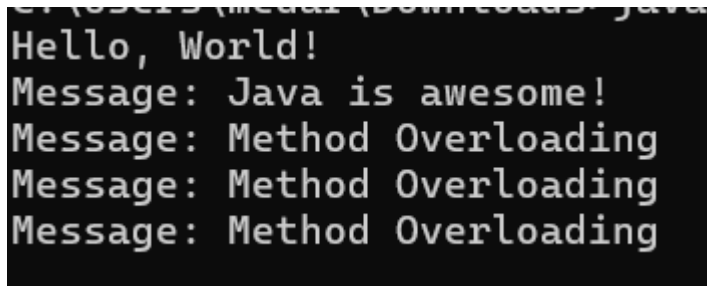
10a)

**METHOD OVERLOADING PROGRAMS****Code:**

```
class DisplayMessage {  
    void display() {  
        System.out.println("Hello, World!");  
    }  
  
    void display(String message) {  
        System.out.println("Message: " + message);  
    }  
  
    void display(String message, int count) {  
        for (int i = 0; i < count; i++) {  
            System.out.println("Message: " + message);  
        }  
    }  
}  
  
public class Main4 {  
    public static void main(String[] args) {  
        DisplayMessage dm = new DisplayMessage();  
    }  
}
```

```
dm.display();  
dm.display("Java is awesome!");  
dm.display("Method Overloading", 3);  
}  
}
```

**Output:**



```
Hello, World!  
Message: Java is awesome!  
Message: Method Overloading  
Message: Method Overloading  
Message: Method Overloading
```

**10B)**

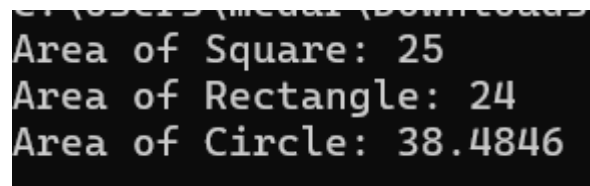
**Code:**

```
class AreaCalculator {  
    int calculateArea(int side) {  
        return side * side;  
    }  
  
    int calculateArea(int length, int width) {  
        return length * width;  
    }  
  
    double calculateArea(double radius) {  
        return 3.1416 * radius * radius;  
    }  
}
```

```
}
```

```
public class Main5 {  
    public static void main(String[] args) {  
        AreaCalculator ac = new AreaCalculator();  
  
        System.out.println("Area of Square: " + ac.calculateArea(5));  
        System.out.println("Area of Rectangle: " + ac.calculateArea(4, 6));  
        System.out.println("Area of Circle: " + ac.calculateArea(3.5));  
    }  
}
```

**Output:**

A screenshot of a terminal window showing the output of the Java program. The text is white on a black background. It displays three lines of output: "Area of Square: 25", "Area of Rectangle: 24", and "Area of Circle: 38.4846".

```
Area of Square: 25  
Area of Rectangle: 24  
Area of Circle: 38.4846
```

11a) **METHOD OVERRIDING PROGRAMS**

**Code:**

```
class Bank {  
    double getRate() {  
        return 5.0;  
    }  
}  
  
class SBI extends Bank {
```

```
@Override  
double getRate() {  
    return 6.5;  
}  
}
```

```
class HDFC extends Bank {  
    @Override  
    double getRate() {  
        return 7.0;  
    }  
}
```

```
class ICICI extends Bank {  
    @Override  
    double getRate() {  
        return 7.5;  
    }  
}
```

```
public class Main6 {  
    public static void main(String[] args) {  
        Bank bank1 = new SBI();  
        System.out.println("SBI Interest Rate: " + bank1.getRate() + "%");  
  
        Bank bank2 = new HDFC();  
        System.out.println("HDFC Interest Rate: " + bank2.getRate() + "%");  
    }  
}
```



```
Bank bank3 = new ICICI();  
System.out.println("ICICI Interest Rate: " + bank3.getRate() + "%");  
}  
}
```

Output:

```
SBI Interest Rate: 6.5%  
HDFC Interest Rate: 7.0%  
ICICI Interest Rate: 7.5%
```

### 11b)

#### Code:

```
class Animal {  
    void sound() {  
        System.out.println("Animals make sounds.");  
    }  
}
```

```
class Dog extends Animal {  
    @Override  
    void sound() {
```

```
        System.out.println("Dog barks.");
    }
}
```

```
class Cat extends Animal {
    @Override
    void sound() {
        System.out.println("Cat meows.");
    }
}
```

```
public class Main7 {
    public static void main(String[] args) {
        Animal myAnimal = new Animal();
        myAnimal.sound();

        Animal myDog = new Dog();
        myDog.sound();

        Animal myCat = new Cat();
        myCat.sound();
    }
}
```

**Code:**

```
Animals make sounds.
Dog barks.
Cat meows.
```

**12a)**

## **ABSTRACTION**

### **INTERFACE PROGRAMS**

#### **Code:**

```
interface Vehicle {  
    void start();  
}
```

```
class Car implements Vehicle {  
    public void start() {  
        System.out.println("Car is starting...");  
    }  
}
```

```
public class Main8 {  
    public static void main(String[] args) {  
        Car c = new Car();  
        c.start();  
    }  
}
```

#### **Output:**

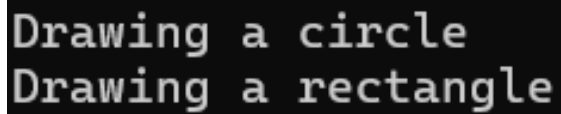
```
Car is starting...
```

## 12b)

### Code:

```
interface Drawable {  
    void draw();  
}  
  
class Circle implements Drawable {  
    public void draw() {  
        System.out.println("Drawing a circle");  
    }  
}  
  
class Rectangle implements Drawable {  
    public void draw() {  
        System.out.println("Drawing a rectangle");  
    }  
}  
  
public class Main9 {  
    public static void main(String[] args) {  
        Drawable d1 = new Circle();  
        Drawable d2 = new Rectangle();  
        d1.draw();  
        d2.draw();  
    }  
}
```

### Output:



Drawing a circle  
Drawing a rectangle

**12c)**

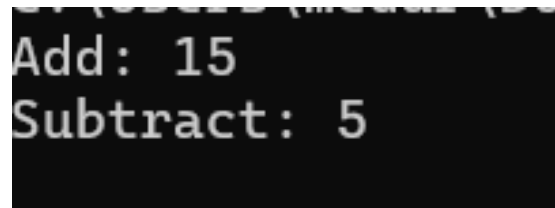
**Code:**

```
interface Calculator {  
    int add(int a, int b);  
    int subtract(int a, int b);  
}
```

```
class SimpleCalculator implements Calculator {  
    public int add(int a, int b) {  
        return a + b;  
    }  
    public int subtract(int a, int b) {  
        return a - b;  
    }  
}
```

```
public class Main10 {  
    public static void main(String[] args) {  
        SimpleCalculator calc = new SimpleCalculator();  
        System.out.println("Add: " + calc.add(10, 5));  
        System.out.println("Subtract: " + calc.subtract(10, 5));  
    }  
}
```

Output:

A screenshot of a terminal window with a black background and white text. The text displays 'Add: 15' on the first line and 'Subtract: 5' on the second line.

**12d)**

**Code:**

```
interface Animal {  
    void eat();  
}
```

```
interface Bird extends Animal {  
    void fly();  
}
```

```
class Eagle implements Bird {  
    public void eat() {  
        System.out.println("Eagle eats meat.");  
    }  
  
    public void fly() {  
        System.out.println("Eagle flies high.");  
    }  
}
```

```
public class Main11 {  
    public static void main(String[] args) {  
        Eagle eagle = new Eagle();  
    }  
}
```

```
eagle.eat();  
eagle.fly();  
}  
}
```

**Output:**

```
Eagle eats meat.  
Eagle flies high.
```

**13a) Abstraction**

**ABSTRACT CLASS PROGRAMS**

**Code:**

```
abstract class Bank {  
    abstract double getRateOfInterest();  
}  
  
class SBI extends Bank {  
    double getRateOfInterest() {  
        return 6.5;  
    }  
}  
  
public class Main12 {  
    public static void main(String[] args) {  
        Bank b = new SBI();  
        System.out.println("Interest Rate: " + b.getRateOfInterest());  
    }  
}
```

**Output:**

```
Interest Rate: 6.5
```

**13b)**

**Code:**

```
abstract class Transport {  
    abstract void move();  
}
```

```
class Bus extends Transport {  
    void move() {  
        System.out.println("Bus moves on road");  
    }  
}
```

```
public class Main13 {  
    public static void main(String[] args) {  
        Transport t = new Bus();  
        t.move();  
    }  
}
```

**Output:**

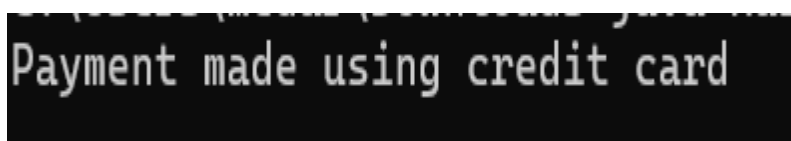
```
Bus moves on road
```

**13c)**



**Code:**

```
abstract class Payment {  
    abstract void makePayment();  
}  
  
class CreditCard extends Payment {  
    void makePayment() {  
        System.out.println("Payment made using credit card");  
    }  
}  
  
public class Main14 {  
    public static void main(String[] args) {  
        Payment p = new CreditCard();  
        p.makePayment();  
    }  
}
```

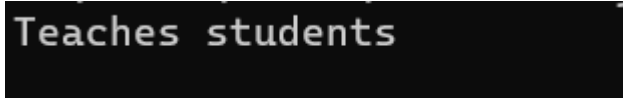
**Output:**A screenshot of a terminal window with a black background and white text. The text displayed is "Payment made using credit card".**13d)****Code:**

```
abstract class Person {  
    abstract void role();  
}  
  
class Teacher extends Person {
```

```
void role() {  
    System.out.println("Teaches students");  
}  
}
```

```
public class Main15 {  
    public static void main(String[] args) {  
        Person p = new Teacher();  
        p.role();  
    }  
}
```

**Output:**



```
Teaches students
```

**14)**

**ENCAPSULATION PROGRAMS**

**14a)**

```
class BankAccount {  
    private double balance;  
  
    public double getBalance() {  
        return balance;  
    }  
  
    public void deposit(double amount) {
```

```
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: " + amount);
        }
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient funds or invalid amount");
        }
    }
}

public class Main16 {
    public static void main(String[] args) {
        BankAccount acc = new BankAccount();
        acc.deposit(500);
        acc.withdraw(200);
        System.out.println("Balance: " + acc.getBalance());
    }
}
```

**Output:**

```
Deposited: 500.0  
Withdrawn: 200.0  
Balance: 300.0
```

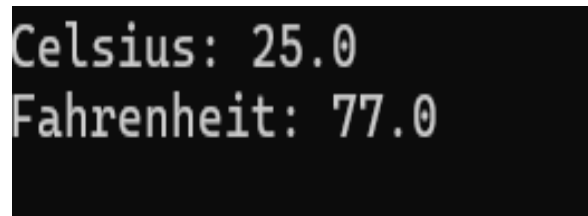
## 14b)

### Code:

```
class Temperature {  
    private double celsius;  
  
    public void setCelsius(double celsius) {  
        if (celsius >= -273.15) {  
            this.celsius = celsius;  
        } else {  
            System.out.println("Invalid temperature");  
        }  
    }  
  
    public double getCelsius() {  
        return celsius;  
    }  
  
    public double getFahrenheit() {  
        return (celsius * 9/5) + 32;  
    }  
}  
  
public class Main17 {  
    public static void main(String[] args) {
```

```
Temperature t = new Temperature();  
t.setCelsius(25);  
System.out.println("Celsius: " + t.getCelsius());  
System.out.println("Fahrenheit: " + t.getFahrenheit());  
}  
}
```

**Output:**



```
Celsius: 25.0  
Fahrenheit: 77.0
```

**14c)**

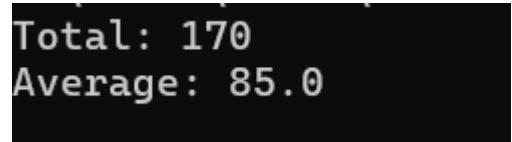
**Code:**

```
class Marks {  
    private int math;  
    private int science;  
  
    public void setMarks(int math, int science) {  
        this.math = math;  
        this.science = science;  
    }  
  
    public int getTotal() {  
        return math + science;  
    }  
  
    public double getAverage() {
```

```
        return (math + science) / 2.0;
    }
}

public class Main18 {
    public static void main(String[] args) {
        Marks m = new Marks();
        m.setMarks(80, 90);
        System.out.println("Total: " + m.getTotal());
        System.out.println("Average: " + m.getAverage());
    }
}
```

**Output:**

A screenshot of a terminal window with a black background and white text. It displays the output of the program: "Total: 170" on the first line and "Average: 85.0" on the second line.

```
Total: 170
Average: 85.0
```

**14d)**

**Code:**

```
class User {
    private String username;
    private String password;

    public void register(String user, String pass) {
        username = user;
        password = pass;
    }

    public boolean login(String user, String pass) {
        return username.equals(user) && password.equals(pass);
    }
}
```

```

    }
}

public class Main19 {

    public static void main(String[] args) {

        User u = new User();

        u.register("admin", "admin123");

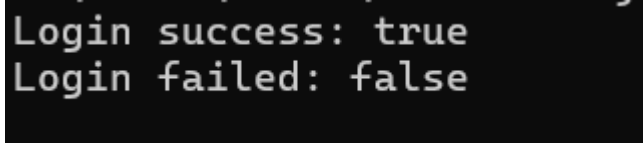

        System.out.println("Login success: " + u.login("admin", "admin123"));

        System.out.println("Login failed: " + u.login("admin", "wrongpass"));

    }
}

```

### **Output:**



```

Login success: true
Login failed: false

```

## **15a) PACKAGES PROGRAMS**

### User Defined Packages

#### **Code:**

```

package contactmanagement;


public class ContactManager {

    private Contact contact;


    public void addContact(String name, String phoneNumber) {

        contact = new Contact(name, phoneNumber);

    }
}

```

```
public void displayContact() {  
    if (contact != null) {  
        System.out.println(contact);  
    } else {  
        System.out.println("No contact found.");  
    }  
}  
}
```

```
package contactmanagement;
```

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

```
public class ContactMain {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        ContactManager contactManager = new ContactManager();  
  
        System.out.print("Enter contact name: ");  
        String name = scanner.nextLine();  
  
        System.out.print("Enter contact phone number: ");  
        String phoneNumber = scanner.nextLine();  
  
        contactManager.addContact(name, phoneNumber);  
        System.out.println("Contact added successfully!");  
    }  
}
```



```
        System.out.println("Displaying contact:");
        contactManager.displayContact();

        scanner.close();
    }
}
```

```
package contactmanagement;
```

```
public class Contact {
    private String name;
    private String phoneNumber;

    public Contact(String name, String phoneNumber) {
        this.name = name;
        this.phoneNumber = phoneNumber;
    }

    public String getName() {
        return name;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }
}
```

```
@Override  
public String toString() {  
    return "Contact [Name: " + name + ", Phone Number: " + phoneNumber + "];"  
}  
}
```

Output:

```
D:\>java contactmanagement.ContactMain  
Enter contact name: dheeraj  
Enter contact phone number: 94907  
Contact added successfully!  
Displaying contact:  
Contact [Name: dheeraj, Phone Number: 94907]
```

### 15b)

User Defined Packages

#### Code:

```
package taskmanagement;
```

```
public class TaskManager {
```

```
    private Task task;
```

```
    public void addTask(String title, String description) {
```

```
        task = new Task(title, description);
```

```
    }
```

```
public void displayTask() {  
    if (task != null) {  
        System.out.println(task);  
    } else {  
        System.out.println("No task found.");  
    }  
}  
}
```

```
package taskmanagement;
```

```
public class Task {  
    private String title;  
    private String description;  
  
    public Task(String title, String description) {  
        this.title = title;  
        this.description = description;  
    }  
}
```

```
public String getTitle() {  
    return title;  
}
```

```
public String getDescription() {  
    return description;  
}
```

```
@Override
```

```
    public String toString() {  
        return "Task [Title: " + title + ", Description: " + description + "];"  
    }  
}
```

```
package taskmanagement;
```

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

```
public class TaskMain {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        TaskManager taskManager = new TaskManager();  
  
        System.out.print("Enter task title: ");  
        String title = scanner.nextLine();  
  
        System.out.print("Enter task description: ");  
        String description = scanner.nextLine();  
  
        taskManager.addTask(title, description);  
        System.out.println("Task added successfully!");  
  
        System.out.println("Displaying task:");  
        taskManager.displayTask();  
    }  
}
```

```
        scanner.close();
    }
}
```

Output:

```
D:\>java taskmanagement.TaskMain
Enter task title: java oops manual
Enter task description: lab manual
Task added successfully!
Displaying task:
Task [Title: java oops manual, Description: lab manual]
```

### 15c)

#### Code:

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class MultiPackageExample {
    public static void main(String[] args) {
        ArrayList<String> logs = new ArrayList<>();
        logs.add("User logged in");
        logs.add("User viewed profile");
        logs.add("User logged out");

        LocalDateTime now = LocalDateTime.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
```

```

try {
    FileWriter writer = new FileWriter("log.txt", true);

    writer.write("----- Log Entry: " + now.format(formatter) + "-----\n");
    for (String log : logs) {
        writer.write(log + "\n");
    }
    writer.write("-----\n\n");

    writer.close();
    System.out.println("Logs written to file successfully.");
} catch (IOException e) {
    System.out.println("Error writing to file: " + e.getMessage());
}
}

```

## Output:

```

PS C:\Users\medar\Downloads> javac MultiPackageExample.java
PS C:\Users\medar\Downloads> java MultiPackageExample
Logs written to file successfully.

```

## 15d)

### Code:

```

import java.util.ArrayList;
import java.util.Collections;
import java.util.Random;

```

```
import java.lang.System;
import java.text.DecimalFormat;

public class RandomSortSystemInfo {
    public static void main(String[] args) {

        Random rand = new Random();
        ArrayList<Double> numbers = new ArrayList<>();

        for (int i = 0; i < 5; i++) {
            numbers.add(rand.nextDouble() * 100);
        }

        Collections.sort(numbers);

        DecimalFormat df = new DecimalFormat("#.##");

        System.out.println("Sorted Random Numbers:");
        for (double num : numbers) {
            System.out.println(df.format(num));
        }

        System.out.println("\nSystem Information:");
        System.out.println("Java Version: " + System.getProperty("java.version"));
        System.out.println("User Name: " + System.getProperty("user.name"));
        System.out.println("Operating System: " + System.getProperty("os.name"));
    }
}
```

```
}  
}
```

### Output:

```
PS C:\Users\medar\Downloads> javac RandomSortSystemInfo.java  
PS C:\Users\medar\Downloads> java RandomSortSystemInfo  
Sorted Random Numbers:  
5.1  
5.85  
68.47  
73.27  
99.36
```



**16)**

## **EXCEPTION HANDLING PROGRAMS**

**16a)**

```
import java.io.FileWriter;
```

```
import java.io.IOException;
```

```
public class CreateFile {
```

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

```
        try {
```

```
            FileWriter writer = new FileWriter("data.txt");
```

```
            writer.write("Hello, this is a new file.\n");
```

```
            writer.write("This is the second line.");
```

```
            writer.close();
```

```
            System.out.println("File created and data written successfully.");
```

```
        } catch (IOException e) {
```

```
            System.out.println("An error occurred.");
```

```
        }
```

```
    }
```

```
}
```

**Output:**

File created and data written successfully.

### 16b)

#### Code:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class ReadFile {
    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(new FileReader("data.txt"));
            String line;
            System.out.println("Reading file contents:");
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
            reader.close();
        } catch (IOException e) {
            System.out.println("An error occurred.");
        }
    }
}
```

```
}
```

**Output:**

```
Reading file contents:  
Hello, this is a new file.  
This is the second line.
```

**16c)**

**Code:**

```
import java.io.File;  
  
public class DeleteFile {  
    public static void main(String[] args) {  
        File file = new File("data.txt");  
  
        if (file.delete()) {  
            System.out.println("File deleted successfully.");  
        } else {  
            System.out.println("File deletion failed.");  
        }  
    }  
}
```

Output:

```
File deleted successfully.
```

**16d)**

**Code:**

```
import java.io.*;
```

```
public class UpdateFile {  
    public static void main(String[] args) {  
        String oldContent = "";  
        String newContent = "";  
  
        try {  
            BufferedReader reader = new BufferedReader(new FileReader("data.txt"));  
            String line;  
  
            while ((line = reader.readLine()) != null) {  
                oldContent += line + "\n";  
            }  
            reader.close();  
  
            newContent = oldContent.replaceAll("Hello", "Hi");  
  
            FileWriter writer = new FileWriter("data.txt");  
            writer.write(newContent);  
            writer.close();  
  
            System.out.println("File updated successfully.");  
        } catch (IOException e) {  
            System.out.println("An error occurred.");  
        }  
    }  
}
```

**Output:**

```
C:\Users\medai\Downloads>java UpdateFile.java  
An error occurred: data.txt (The system cannot find the file specified)
```

17)

**FILE HANDLING PROGRAMS**

17a)

Code:

```
import java.io.File;
```

```
public class FileMetadata {  
    public static void main(String[] args) {  
        File file = new File("data.txt");  
  
        if (file.exists()) {  
            System.out.println("File Name: " + file.getName());  
            System.out.println("Absolute Path: " + file.getAbsolutePath());  
            System.out.println("Writable: " + file.canWrite());  
            System.out.println("Readable: " + file.canRead());  
            System.out.println("File Size (bytes): " + file.length());  
            System.out.println("Last Modified: " + file.lastModified());  
        } else {  
            System.out.println("File does not exist.");  
        }  
    }  
}
```

Output:

```
PS C:\Users\medar> cd Downloads
PS C:\Users\medar\Downloads> javac FileMetadata.java
PS C:\Users\medar\Downloads> java FileMetadata
File Name: data.txt
Absolute Path: C:\Users\medar\Downloads\data.txt
Writable: true
Readable: true
File Size (bytes): 0
Last Modified: 1743857774110
```

17b)

Code:

```
import java.io.*;
```

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

```
public class UpdateLineInFile {
    public static void main(String[] args) {
        String filePath = "data.txt";
        List<String> lines = new ArrayList<>();

        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            String line;

            while ((line = reader.readLine()) != null) {
                if (line.contains("oldword")) {
                    line = line.replace("oldword", "newword");
                }
                lines.add(line);
            }
        } catch (IOException e) {
            System.out.println("Error reading the file.");
        }
    }
}
```

```

try (BufferedWriter writer = new BufferedWriter(new FileWriter(filePath))) {
    for (String l : lines) {
        writer.write(l);
        writer.newLine();
    }
    System.out.println("File updated successfully.");
} catch (IOException e) {
    System.out.println("Error writing the file.");
}
}
}

```

Output:

```

Last Modified: 1743857774110
PS C:\Users\medar\Downloads> javac UpdateLineInFile.java
PS C:\Users\medar\Downloads> java UpdateLineInFile
File updated successfully.

```

17c)

Code:

```
import java.io.*;
```

```

public class CopyFile {
    public static void main(String[] args) {
        try (FileInputStream in = new FileInputStream("source.txt");
            FileOutputStream out = new FileOutputStream("destination.txt")) {

```

```

        int content;

        while ((content = in.read()) != -1) {
            out.write(content);
        }

        System.out.println("File copied successfully.");
    } catch (IOException e) {
        System.out.println("Error during copy: " + e.getMessage());
    }
}
}

```

Output:

```

PS C:\Users\medar\Downloads> javac CopyFile.java
PS C:\Users\medar\Downloads> java CopyFile
File copied successfully.

```

## 17d)

### Code:

```

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class SearchInFile {
    public static void main(String[] args) {
        String wordToFind = "Java";
        boolean found = false;
    }
}

```



```
try (BufferedReader reader = new BufferedReader(new
FileReader("sample.txt"))) {
    String line;
    int lineNumber = 1;

    while ((line = reader.readLine()) != null) {
        if (line.contains(wordToFind)) {
            System.out.println("Found \"" + wordToFind + "\" on line " + lineNumber
+ ": " + line);
            found = true;
        }
        lineNumber++;
    }

    if (!found) {
        System.out.println("Word not found.");
    }
} catch (IOException e) {
    System.out.println("Error reading the file.");
}
}
```

Output:

```
PS C:\Users\medar\Downloads> javac SearchInFile.java
PS C:\Users\medar\Downloads> java SearchInFile
Found "Java" on line 3: Java
```