LAB RECORD

23CSE111- Object Oriented Programming

Submitted by

CH.SC.U4CSE24120 -Konda Bhavani

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

CHENNAI

March - 2025

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.U4CSE24120 – Konda Bhavani*** 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

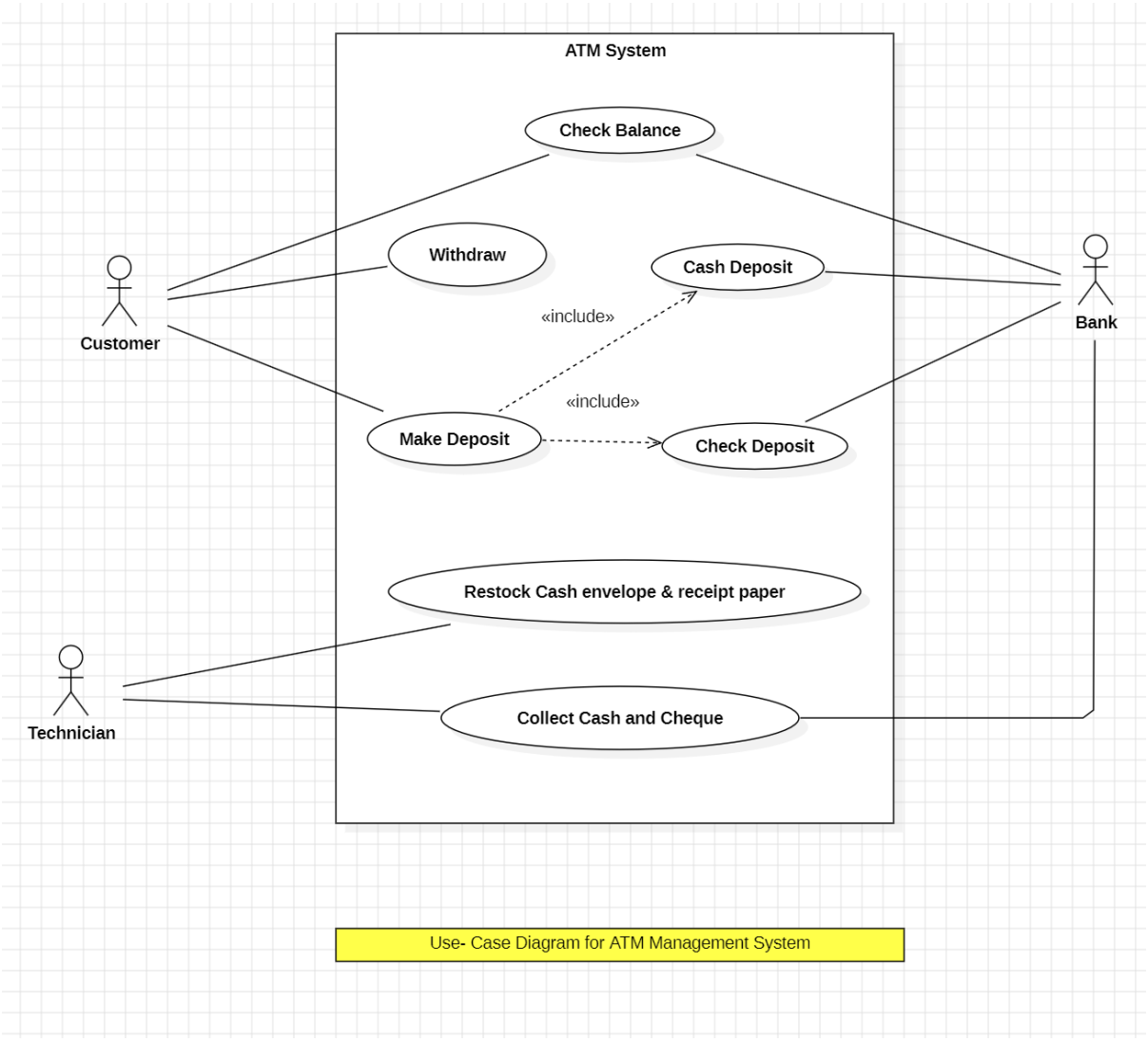
Internal Examiner 1 Internal Examiner 2

INDEX

UML DIAGRAMS

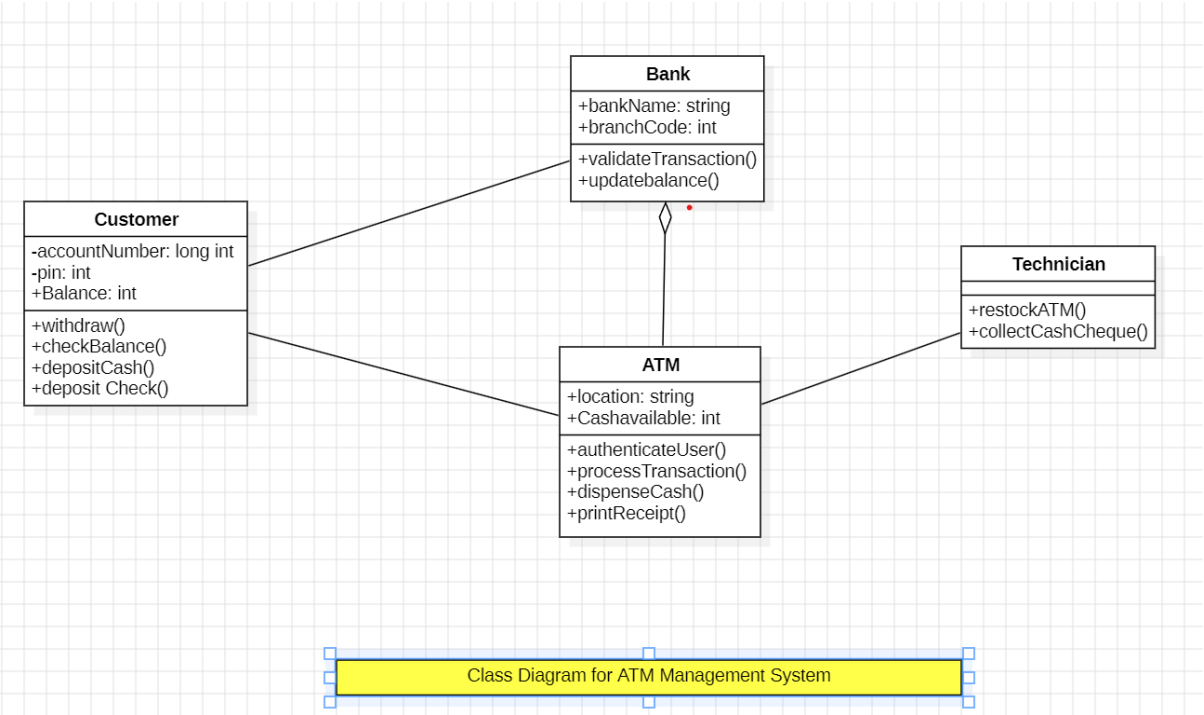
1. ATM MANAGEMENT SYSTEM

1. a) Use Case Diagram:

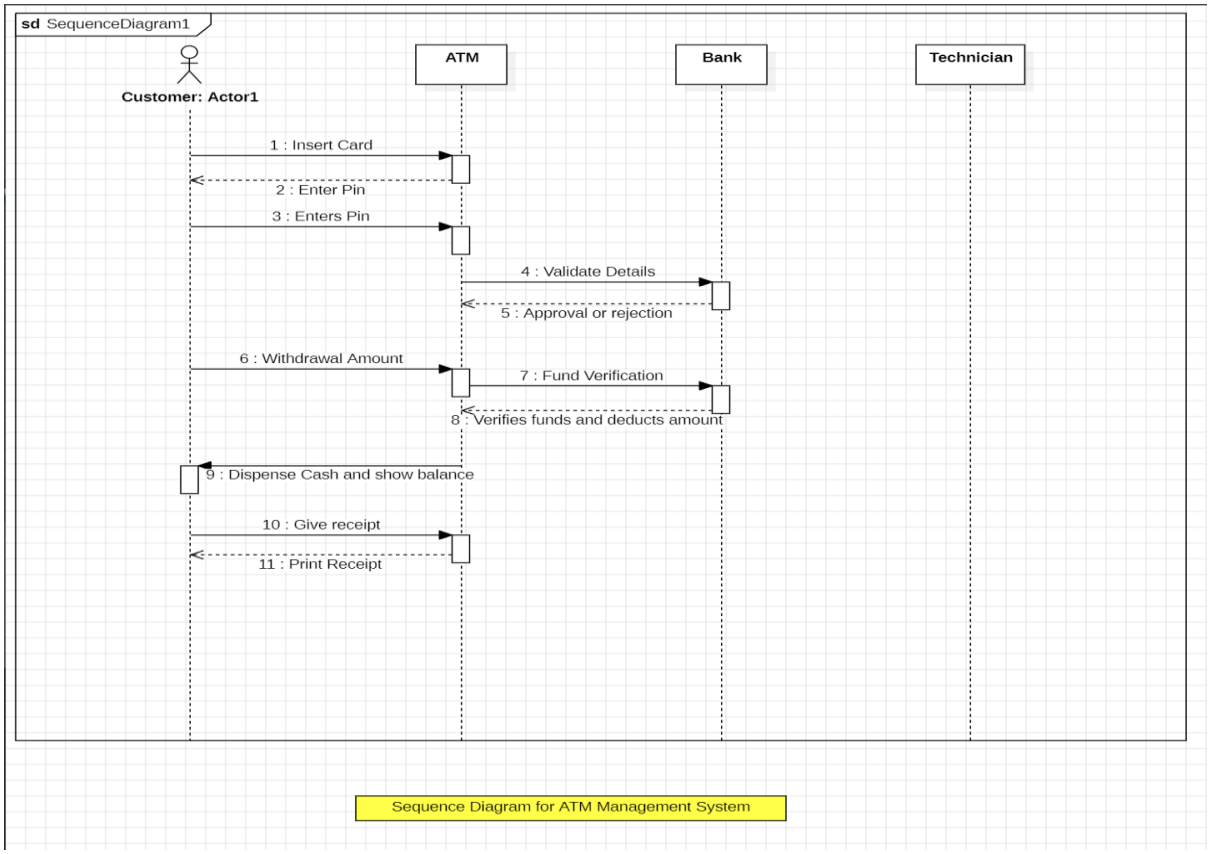


Use- Case Diagram for ATM Management System

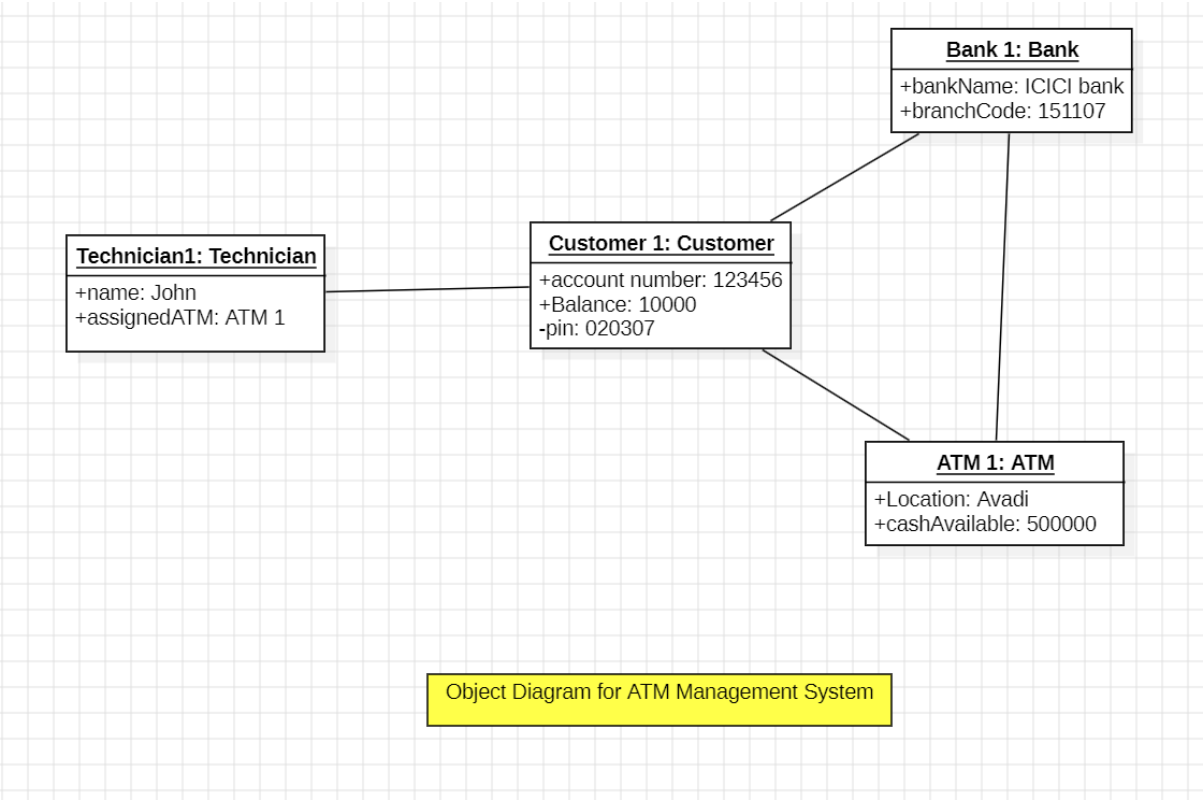
1. b) Class Diagram:



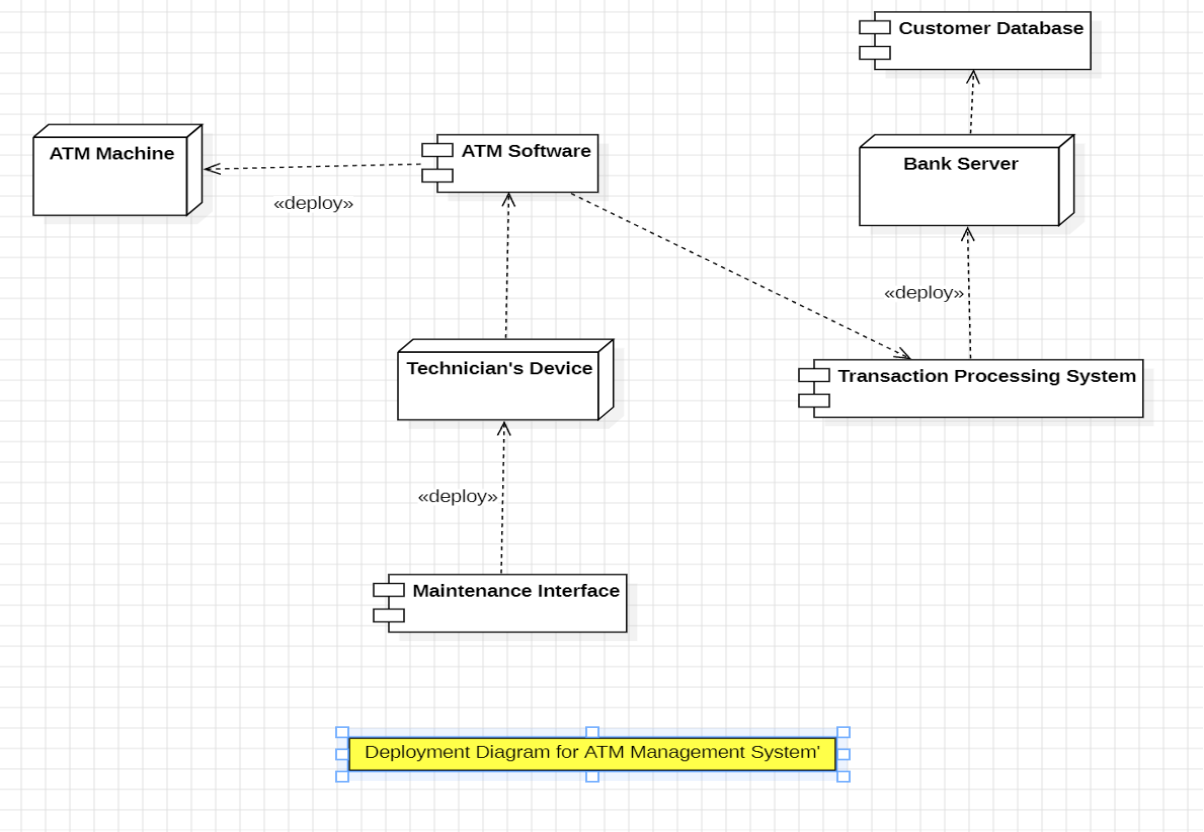
1. c) Sequence Diagram:



1. d) Object Diagram:

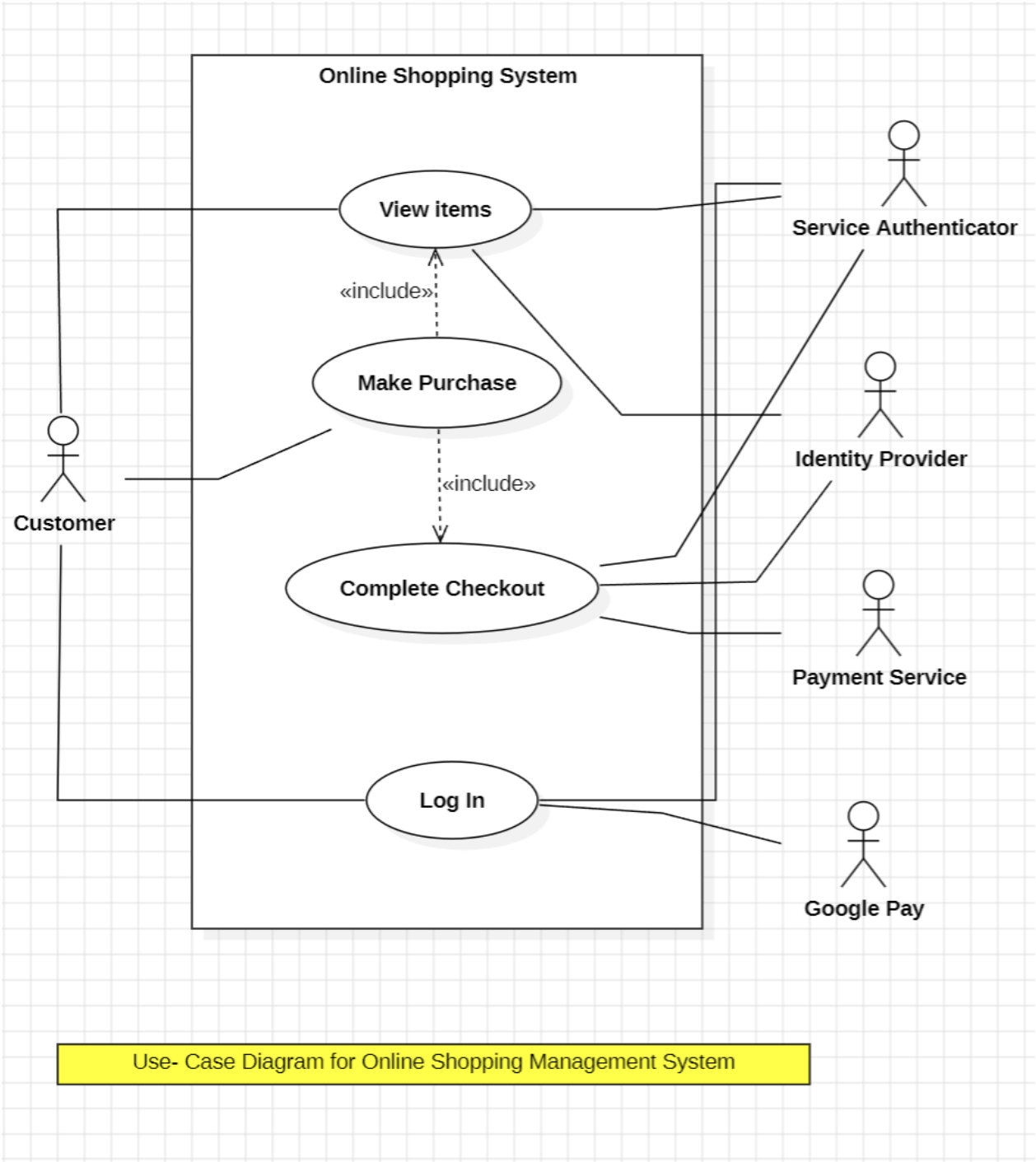


1. e) Deployment Diagram:



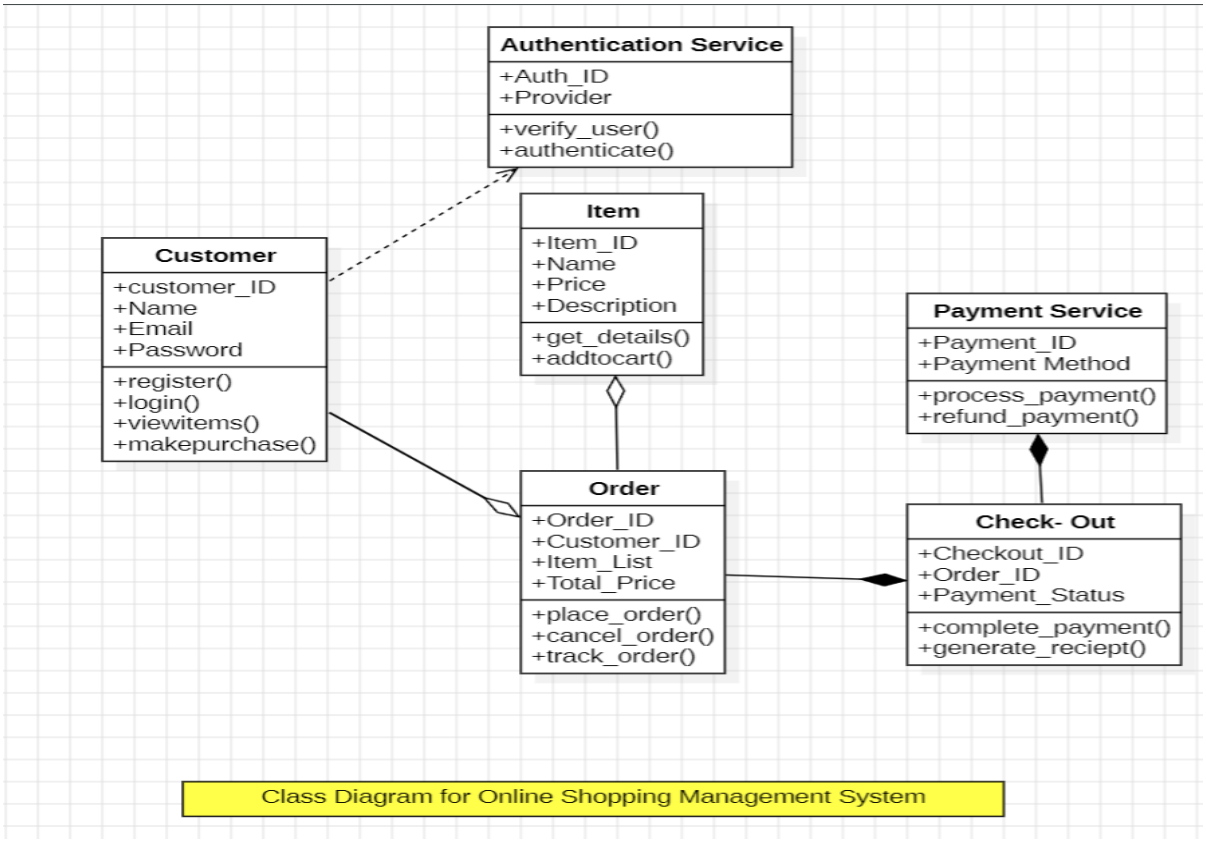
2. ONLINE SHOPPING MANAGEMENT SYSTEM

2.a) Use Case Diagram:



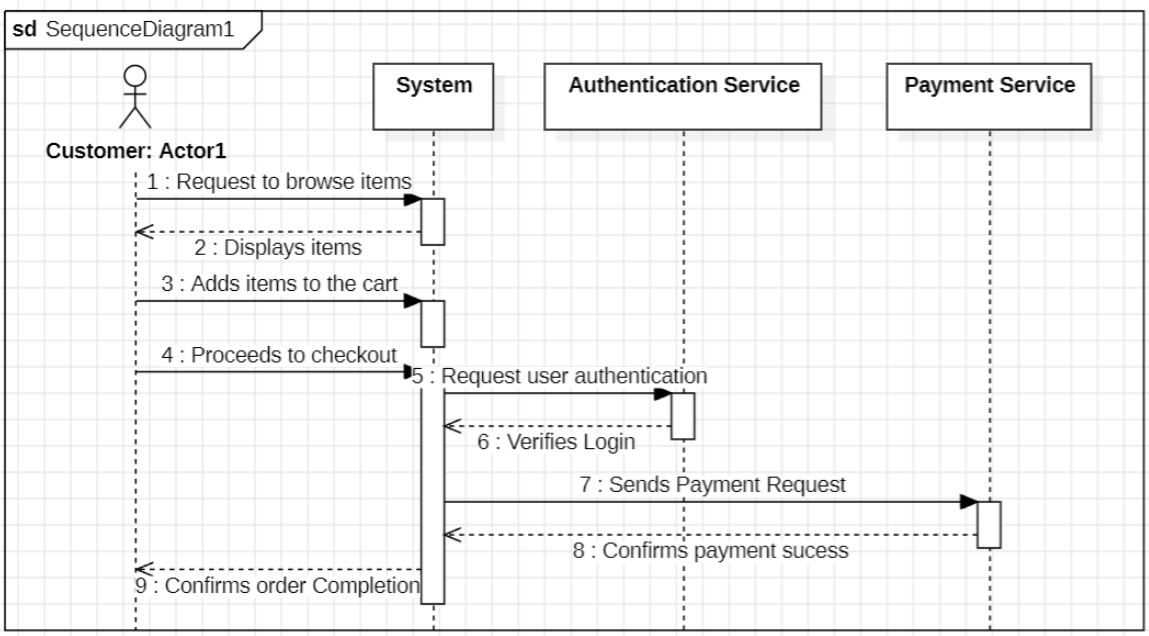
Use- Case Diagram for Online Shopping Management System

2.b) Class Diagram:



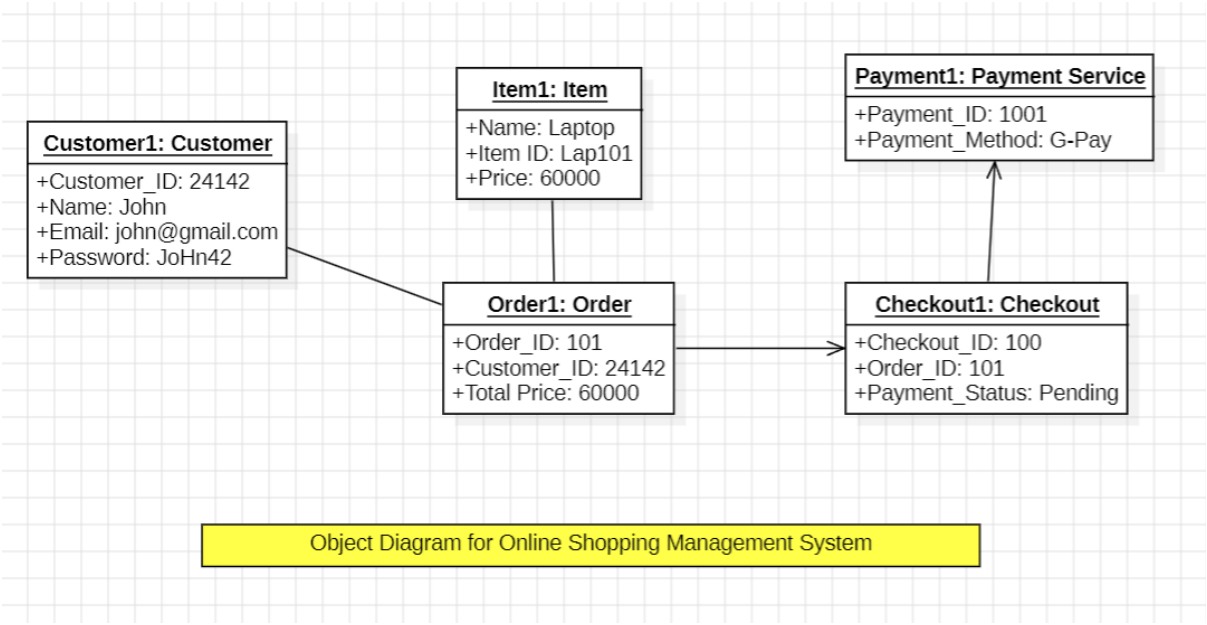
Class Diagram for Online Shopping Management System

2. c) Sequence Diagram:

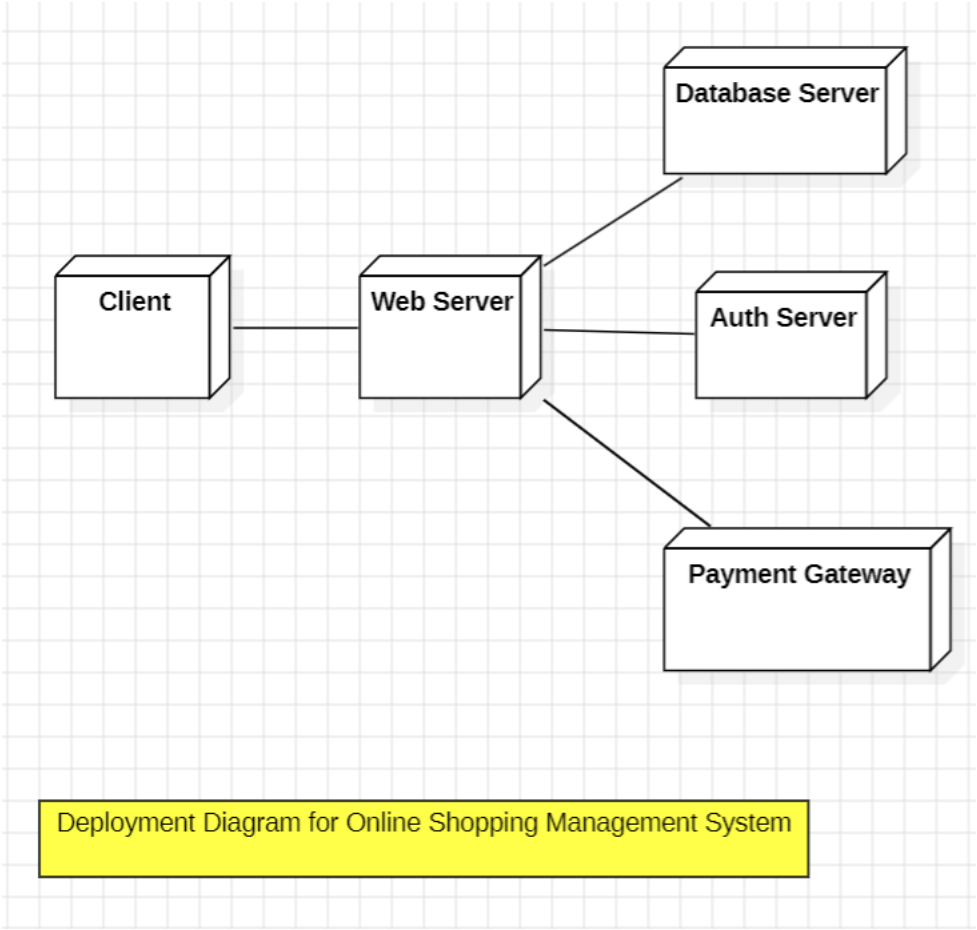


Sequence Diagram for Online Shopping Management System

2.d) Object Diagram:



2.e) Deployment Diagram:



3.Basic Java Programs

2. a) Cash- Withdrawal System:

Code:

```java
import java.util.Scanner;

public class ATMWithdrawal {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter balance: ");
        double balance = scanner.nextDouble();

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

        if (amount > balance) {
            System.out.println("Insufficient balance!");
        } else if (amount <= 0) {
            System.out.println("Invalid amount entered!");
        } else {
            balance -= amount;
            System.out.println("Withdrawal successful! Remaining balance: " + balance);
        }
        scanner.close();
    }
}
```

Output:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>javac ATMWithdrawal.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA>java ATMWithdrawal
Enter balance: 10000
Enter withdrawal amount: 1000
Withdrawal successful! Remaining balance: 9000.0
```

3.b) Odd or Even:

Code:

```java
import java.util.Scanner;

public class EvenOdd {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();

        if (num % 2 == 0) {
            System.out.println(num + " is Even.");
        } else {
            System.out.println(num + " is Odd.");
        }
        scanner.close();
    }
}
```

Output:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>java EvenOdd
Enter a number: 32
32 is Even.
```

3.c) Largest Number:

Code:

```java
import java.util.Scanner;

public class LargestNumber {
    Run | Debug
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print(s:"Enter three numbers: ");
        int a = scanner.nextInt();
        int b = scanner.nextInt();
        int c = scanner.nextInt();

        if (a >= b && a >= c) {
            System.out.println(a + " is the largest.");
        } else if (b >= a && b >= c) {
            System.out.println(b + " is the largest.");
        } else {
            System.out.println(c + " is the largest.");
        }
        scanner.close();
    }
}
```

Output:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>java LargestNumber
Enter three numbers: 32
45
18
45 is the largest.
```

3.d) Leap Year Checker:

Code:

```java
import java.util.Scanner;

public class LeapYearCheck {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a year: ");
        int year = scanner.nextInt();

        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
            System.out.println(year + " is a Leap Year.");
        } else {
            System.out.println(year + " is not a Leap Year.");
        }
        scanner.close();
    }
}
```

Output:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>java LeapYearCheck
Enter a year: 2024
2024 is a Leap Year.
```

3.e) Number Checker:

Code:

```java
import java.util.Scanner;

public class NumberCheck {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();

        if (num > 0) {
            System.out.println("The number is Positive.");
        } else if (num < 0) {
            System.out.println("The number is Negative.");
        } else {
            System.out.println("The number is Zero.");
        }
        scanner.close();
    }
}
```

Output:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>java NumberCheck
Enter a number: -45
The number is Negative.
```

3.f) Quadratic Equation:

Code:

```java
import java.util.Scanner;

public class QuadraticEquation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter coefficients a, b, and c: ");
        double a = scanner.nextDouble();
        double b = scanner.nextDouble();
        double c = scanner.nextDouble();

        double discriminant = b * b - 4 * a * c;

        if (discriminant > 0) {
            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            System.out.println("Roots are real and different: " + root1 + ", " + root2);
        } else if (discriminant == 0) {
            double root = -b / (2 * a);
            System.out.println("Roots are real and equal: " + root);
        } else {
            System.out.println("Roots are imaginary.");
        }
        scanner.close();
    }
}
```

Output:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>javac QuadraticEquation.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA>java QuadraticEquation
Enter coefficients a, b, and c: 1
5
6
Roots are real and different: -2.0, -3.0
```

3.g) Student Grades:

Code:

```
import java.util.Scanner;

public class StudentGrade {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter marks: ");
        int marks = scanner.nextInt();

        if (marks >= 90) {
            System.out.println("Grade: A");
        } else if (marks >= 80) {
            System.out.println("Grade: B");
        } else if (marks >= 70) {
            System.out.println("Grade: C");
        } else if (marks >= 60) {
            System.out.println("Grade: D");
        } else {
            System.out.println("Grade: F (Fail)");
        }
        scanner.close();
    }
}
```

Output:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>javac StudentGrade.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA>java StudentGrade
Enter marks: 97
Grade: A
```

3.h) Triangle Type:

Code:

```java
import java.util.Scanner;

public class TriangleType {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter three sides of the triangle: ");
        int a = scanner.nextInt();
        int b = scanner.nextInt();
        int c = scanner.nextInt();

        if (a == b && b == c) {
            System.out.println("It is an Equilateral Triangle.");
        } else if (a == b || b == c || a == c) {
            System.out.println("It is an Isosceles Triangle.");
        } else {
            System.out.println("It is a Scalene Triangle.");
        }
        scanner.close();
    }
}
```

Output:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>javac TriangleType.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA>java TriangleType
Enter three sides of the triangle: 5
5
5
It is an Equilateral Triangle.
```

3.i) Voting Eligibility:

Code:

```java
import java.util.Scanner;

public class VotingEligibility {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your age: ");
        int age = scanner.nextInt();

        if (age >= 18) {
            System.out.println("You are eligible to vote.");
        } else {
            System.out.println("You are not eligible to vote.");
        }
        scanner.close();
    }
}
```

Output:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>javac VotingEligibility.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA>java VotingEligibility
Enter your age: 21
You are eligible to vote.
```

3.j) Vowels and Consonants Classifier:

Code:

```java
import java.util.Scanner;

public class VowelConsonant {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a character: ");
        char ch = scanner.next().toLowerCase().charAt(0);

        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
            System.out.println(ch + " is a Vowel.");
        } else if ((ch >= 'a' && ch <= 'z')) {
            System.out.println(ch + " is a Consonant.");
        } else {
            System.out.println("Invalid input! Please enter a letter.");
        }
        scanner.close();
    }
}
```

Output:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA>javac VowelConsonant.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA>java VowelConsonant
Enter a character: u
u is a Vowel.
```

4.SINGLE INHERITANCE PROGRAMS

4.a) Parking System:

Code:

```
class ParkingSystem {
    protected int availableSpots;

    public ParkingSystem(int totalSpots) {
        this.availableSpots = totalSpots;
    }

    public boolean parkCar() {
        if (availableSpots > 0) {
            availableSpots--;
            System.out.println("Car parked successfully. Spots left: " + availableSpots);
            return true;
        }
        System.out.println("Parking full. Cannot park car.");
        return false;
    }

    public void exitCar() {
        availableSpots++;
        System.out.println("Car exited. Spots available: " + availableSpots);
    }
}

class PaidParkingSystem extends ParkingSystem {
    private double hourlyRate;
    private double totalRevenue;

    public PaidParkingSystem(int totalSpots, double hourlyRate) {
        super(totalSpots);
        this.hourlyRate = hourlyRate;
        this.totalRevenue = 0;
    }

    public double calculateFee(int hours) {
        return hours * hourlyRate;
    }

    public void makePayment(double amount) {
        totalRevenue += amount;
        System.out.println("Payment of $" + amount + " received.");
    }

    public void displayRevenue() {
        System.out.println("Total revenue: $" + totalRevenue);
    }
}
public class Parking_System {
    public static void main(String[] args) {
        System.out.println("Basic Parking System:");
        ParkingSystem basicParking = new ParkingSystem(3);
        basicParking.parkCar();
        basicParking.parkCar();
        basicParking.parkCar();
        basicParking.parkCar();
        basicParking.exitCar();

        System.out.println("\nPaid Parking System:");
        PaidParkingSystem paidParking = new PaidParkingSystem(2, 2.5);
        paidParking.parkCar();
        paidParking.parkCar();

        double fee = paidParking.calculateFee(3);
        paidParking.makePayment(fee);
        paidParking.exitCar();
        paidParking.displayRevenue();
    }
}
```

Output:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Single Inheritance>javac Parking_System.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Single Inheritance>java Parking_System.java
Basic Parking System:
Car parked successfully. Spots left: 2
Car parked successfully. Spots left: 1
Car parked successfully. Spots left: 0
Parking full. Cannot park car.
Car exited. Spots available: 1

Paid Parking System:
Car parked successfully. Spots left: 1
Car parked successfully. Spots left: 0
Payment of $7.5 received.
Car exited. Spots available: 1
Total revenue: $7.5
```

4.b) Perimeter Calculator

Code:

```java
import java.text.DecimalFormat;
class Shape {
    protected String name;
    protected static int shapeCount = 0;
    public Shape(String name) {
        this.name = name;
        shapeCount++;
    }
    public void displayInfo() {
        System.out.println("\nShape Information:");
        System.out.println("Name: " + name);
    }
    public double calculatePerimeter() {
        System.out.println("Perimeter calculation not defined for generic shape");
        return 0;
    }
    public static void displayTotalShapes() {
        System.out.println("\nTotal shapes created: " + shapeCount);
    }
}
class Rectangle extends Shape {
    private double length;
    private double width;
    private DecimalFormat df = new DecimalFormat("0.00");
    public Rectangle(double length, double width) {
        super("Rectangle");
        this.length = length;
        this.width = width;
    }
    public double calculatePerimeter() {
        double perimeter = 2 * (length + width);
        System.out.println("Perimeter of " + name + ": " + df.format(perimeter));
        return perimeter;
    }
    public boolean isSquare() {
        return length == width;
    }
}
class Circle extends Shape {
    private double radius;
    private DecimalFormat df = new DecimalFormat("0.00");
    public Circle(double radius) {
        super("Circle");
        this.radius = radius;
    }
    public double calculatePerimeter() {
        double perimeter = 2 * Math.PI * radius;
        System.out.println("Circumference of " + name + ": " + df.format(perimeter));
        return perimeter;
    }
    public double calculateArea() {
        return Math.PI * radius * radius;
    }
    public void displayArea() {
        System.out.println("Area of " + name + ": " + df.format(calculateArea()));
    }
}
public class Perimeter_Calculator {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(5.5, 3.2);
        rectangle.displayInfo();
        rectangle.calculatePerimeter();
        System.out.println("Is square? " + rectangle.isSquare());
        Rectangle square = new Rectangle(4.0, 4.0);
        square.displayInfo();
        square.calculatePerimeter();
        System.out.println("Is square? " + square.isSquare());
        Circle circle = new Circle(3.0);
        circle.displayInfo();
        circle.calculatePerimeter();
        circle.displayArea();
        Shape.displayTotalShapes();
    }
}
```

Output:

```
D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Single Inheritance>javac Perimeter_Calculator.java

D:\AVV CHENNAI\Object Oriented Programming\JAVA\Inheritance\Single Inheritance>java Perimeter_Calculator.java

Shape Information:
Name: Rectangle
Perimeter of Rectangle: 17.40
Is square? false

Shape Information:
Name: Rectangle
Perimeter of Rectangle: 16.00
Is square? true

Shape Information:
Name: Circle
Circumference of Circle: 18.85
Area of Circle: 28.27

Total shapes created: 3
```

5. MULTILEVEL INHERITANCE PROGRAMS

5.a) Employee Management

Code:

```
class Employee {
    String name = "John Doe";
    double salary = 50000;

    void showDetails() {
        System.out.println("Employee Name: " + name);
        System.out.println("Salary: $" + salary);
    }
}

class Manager extends Employee {
    String department = "IT";

    void showManagerDetails() {
        System.out.println("Manager Department: " + department);
    }
}

class HR extends Manager {
    void hireEmployee() {
        System.out.println("HR is hiring employees.");
    }
}
public class EmployeeManagement {
    public static void main(String[] args) {
        HR hrPerson = new HR();

        hrPerson.showDetails();
        hrPerson.showManagerDetails();
        hrPerson.hireEmployee();
    }
}
```

Output:

```
D:\OOP>javac EmployeeManagement.java

D:\OOP>java EmployeeManagement
Employee Name: John Doe
Salary: $50000.0
Manager Department: IT
HR is hiring employees.
```

5.b) Banking System

Code:

```java
class BankAccount {
    double balance = 5000;

    void showBalance() {
        System.out.println("Account balance: $" + balance);
    }
}
class SavingsAccount extends BankAccount {
    double interestRate = 5.0;

    void addInterest() {
        balance += balance * (interestRate / 100);
        System.out.println("Interest added. New balance: $" + balance);
    }
}
class FixedDepositAccount extends SavingsAccount {
    void fixedDeposit(double amount) {
        balance += amount;
        System.out.println("Fixed deposit of $" + amount + " added. Updated balance: $" + balance);
    }
}
public class BankingSystem {
    public static void main(String[] args) {
        FixedDepositAccount myAccount = new FixedDepositAccount();

        myAccount.showBalance();
        myAccount.addInterest();
        myAccount.fixedDeposit(2000);
    }
}
```

Output:

```
D:\OOP>javac BankingSystem.java

D:\OOP>java BankingSystem
Account balance: $5000.0
Interest added. New balance: $5250.0
Fixed deposit of $2000.0 added. Updated balance: $7250.0
```

6. HIERARCHICAL INHERITANCE PROGRAMS

6.a) Smart Home System

Code:

```java
class SmartDevice {
    void turnOn() {
        System.out.println("Device is now ON.");
    }
}

class SmartLight extends SmartDevice {
    void adjustBrightness() {
        System.out.println("Brightness adjusted.");
    }
}

class SmartThermostat extends SmartDevice {
    void setTemperature(int temp) {
        System.out.println("Temperature set to " + temp + "°C.");
    }
}

public class SmartHome {
    public static void main(String[] args) {
        SmartLight light = new SmartLight();
        light.turnOn();
        light.adjustBrightness();

        SmartThermostat thermostat = new SmartThermostat();
        thermostat.turnOn();
        thermostat.setTemperature(22);
    }
}
```

Output:

```
D:\OOP>javac SmartHomeSystem.java

D:\OOP>java SmartHomeSystem
Lights are now ON
Door is now UNLOCKED
```

6.b) Transport Booking

Code:

```
class Transport {
    void bookRide() {
        System.out.println("Ride booked successfully.");
    }
}

class Taxi extends Transport {
    void fareEstimate() {
        System.out.println("Estimated fare: $15.");
    }
}

class Bus extends Transport {
    void busSchedule() {
        System.out.println("Next bus arrives in 10 minutes.");
    }
}

public class TransportBooking {
    public static void main(String[] args) {
        Taxi taxi = new Taxi();
        taxi.bookRide();
        taxi.fareEstimate();

        Bus bus = new Bus();
        bus.bookRide();
        bus.busSchedule();
    }
}
```

Output:

```
D:\OOP>javac TransportBooking.java

D:\OOP>java TransportBooking
Ride booked successfully.
Estimated fare: $15.
Ride booked successfully.
Next bus arrives in 10 minutes.
```

7. HYBRID INHERITANCE PROGRAMS

7.a) Animal System

Code:

```java
import java.util.Scanner;

public class AnimalSystem {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.getName();
        dog.getGestationPeriod();
        dog.getBreed();
    }
}
class Animal {
    String name;
    void getName() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter animal name: ");
        name = scanner.nextLine();
    }
}
class Diet {
    String dietType;
    void getDietType() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter diet type: ");
        dietType = scanner.nextLine();
    }
}
class Mammal extends Animal {
    int gestationPeriod;
    void getGestationPeriod() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter gestation period (days): ");
        gestationPeriod = scanner.nextInt();
    }
}
class Dog extends Mammal {
    String breed;
    void getBreed() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter dog breed: ");
        breed = scanner.nextLine();
    }
}
```

Output:

```
D:\OOP>java AnimalSystem
Enter animal name: Lion
Enter gestation period (days): 7
Enter dog breed: jermansheperd
```

7.b) Device System

Code:

```java
import java.util.Scanner;

public class DeviceSystem {
    public static void main(String[] args) {
        Laptop laptop = new Laptop();
        laptop.getBrand();
        laptop.getRAM();
        laptop.getWeight();
    }
}

class Device {
    String brand;
    void getBrand() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter device brand: ");
        brand = scanner.nextLine();
    }
}

class Processor {
    String type;
    void getType() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter processor type: ");
        type = scanner.nextLine();
    }
}

class Computer extends Device {
    int ram;
    void getRAM() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter RAM size (GB): ");
        ram = scanner.nextInt();
    }
}

class Laptop extends Computer {
    double weight;
    void getWeight() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter laptop weight (kg): ");
        weight = scanner.nextDouble();
    }
}
```

Output:

```
D:\OOP>javac DeviceSystem.java

D:\OOP>java DeviceSystem
Enter device brand: IQ00
Enter RAM size (GB): 250
Enter laptop weight (kg): 1.5
```

8. CONSTRUCTOR PROGRAMS

8.a) Pizza

Code:

```
public class Pizza {
    private final String size;

    private boolean cheese = false;
    private boolean pepperoni = false;
    private boolean mushrooms = false;

    public static class Builder {
        private final String size;

        private boolean cheese = false;
        private boolean pepperoni = false;
        private boolean mushrooms = false;

        public Builder(String size) {
            this.size = size;
        }

        public Builder addCheese() {
            this.cheese = true;
            return this;
        }

        public Builder addPepperoni() {
            this.pepperoni = true;
            return this;
        }

        public Builder addMushrooms() {
            this.mushrooms = true;
            return this;
        }

        public Pizza build() {
            return new Pizza(this);
        }
    }

    private Pizza(Builder builder) {
        this.size = builder.size;
        this.cheese = builder.cheese;
        this.pepperoni = builder.pepperoni;
        this.mushrooms = builder.mushrooms;
    }

    @Override
    public String toString() {
        return "Pizza: " + size +
               (cheese ? " +cheese" : "") +
               (pepperoni ? " +pepperoni" : "") +
               (mushrooms ? " +mushrooms" : "");
    }

    Run | Debug
    public static void main(String[] args) {
        Pizza myPizza = new Pizza.Builder(size:"Large")
                            .addCheese()
                            .addPepperoni()
                            .build();

        System.out.println(myPizza);
    }
}
```

Output:

```
D:\OOP>javac Pizza.java

D:\OOP>java Pizza
Pizza: Large +cheese +pepperoni
```

9. CONSTRUCTOR OVERLOADING PROGRAMS

9.a) Bank Account

Code:

```java
class BankAccount {
    private String name;
    private double balance;
    private String accountType;

    public BankAccount(String name, double deposit) {
        this.name = name;
        this.balance = deposit;
        this.accountType = "Savings";
    }

    public BankAccount(String name, double deposit, String accountType) {
        this.name = name;
        this.balance = deposit;
        this.accountType = accountType;
    }

    public BankAccount() {
        this.name = "Anonymous";
        this.balance = 0.0;
        this.accountType = "Checking";
    }

    public void displayAccountDetails() {
        System.out.println("Account Holder: " + name);
        System.out.println("Balance: $" + balance);
        System.out.println("Account Type: " + accountType);
    }

    public static void main(String[] args) {
        BankAccount account1 = new BankAccount("Alice", 1000);
        account1.displayAccountDetails();

        BankAccount account2 = new BankAccount("Bob", 5000, "Checking");
        account2.displayAccountDetails();

        BankAccount account3 = new BankAccount();
        account3.displayAccountDetails();
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Documents>javac BankAccount.java

C:\Users\ch.sc.u4cse24120\Documents>java BankAccount
Account Holder: Alice
Balance: $1000.0
Account Type: Savings
Account Holder: Bob
Balance: $5000.0
Account Type: Checking
Account Holder: Anonymous
Balance: $0.0
Account Type: Checking
```

10. METHOD OVERLOADING PROGRAMS

10.a) Ecommerce App

Code:

```
class ProductSearch {
    void search(String name) {
        System.out.println("Searching for product: " + name);
    }

    void search(String name, String category) {
        System.out.println("Searching for " + name + " in category: " + category);
    }

    void search(int productId) {
        System.out.println("Searching for product with ID: " + productId);
    }
}

public class EcommerceApp {
    public static void main(String[] args) {
        ProductSearch search = new ProductSearch();

        search.search("Laptop");
        search.search("Laptop", "Electronics");
        search.search(101);
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Documents>javac EcommerceApp.java

C:\Users\ch.sc.u4cse24120\Documents>java EcommerceApp
Searching for product: Laptop
Searching for Laptop in category: Electronics
Searching for product with ID: 101
```

10.b) Hospital System

Code:

```
class Hospital {
    void registerPatient(String name, int age) {
        System.out.println("Registering patient: " + name + ", Age: " + age);
    }

    void registerPatient(String name, int age, String disease) {
        System.out.println("Registering patient: " + name + ", Age: " + age + ", Disease: " + disease);
    }

    void registerPatient(String name, int age, String disease, String emergencyLevel) {
        System.out.println("Registering patient: " + name + ", Age: " + age + ", Disease: " + disease + ", Emergency Level: " + emergencyLevel);
    }
}

public class HospitalSystem {
    public static void main(String[] args) {
        Hospital hospital = new Hospital();

        hospital.registerPatient("Alice", 30);
        hospital.registerPatient("Bob", 45, "Diabetes");
        hospital.registerPatient("Charlie", 60, "Heart Attack", "Critical");
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Documents>javac HospitalSystem.java

C:\Users\ch.sc.u4cse24120\Documents>java HospitalSystem
Registering patient: Alice, Age: 30
Registering patient: Bob, Age: 45, Disease: Diabetes
Registering patient: Charlie, Age: 60, Disease: Heart Attack, Emergency Level: Critical
```

11. METHOD OVERRIDING PROGRAMS

11.a) Vehicle Info

Code:

```java
class Vehicle {
    private String make;
    private String model;
    private int year;
    public Vehicle(String make, String model, int year) {
        this.make = make;
        this.model = model;
        this.year = year;
    }
    public void start() {
        System.out.println("Starting the vehicle...");
    }
    public void displayInfo() {
        System.out.println("Make: " + make + ", Model: " + model + ", Year: " + year);
    }
    public final void stop() {
        System.out.println("Stopping the vehicle...");
    }
}
class ElectricCar extends Vehicle {
    private int batteryCapacity;
    public ElectricCar(String make, String model, int year, int batteryCapacity) {
        super(make, model, year);
        this.batteryCapacity = batteryCapacity;
    }
    public void start() {
        System.out.println("Starting electric motor silently...");
        checkBattery();
    }
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Battery Capacity: " + batteryCapacity + " kWh");
    }
    private void checkBattery() {
        System.out.println("Battery level: 95%");
    }
}
class Motorcycle extends Vehicle {
    private boolean hasSideCar;
    public Motorcycle(String make, String model, int year, boolean hasSideCar) {
        super(make, model, year);
        this.hasSideCar = hasSideCar;
    }
    public void start() {
        System.out.println("Kickstarting the motorcycle...");
    }
    public void displayInfo() {
        super.displayInfo();
        System.out.println("Has Sidecar: " + (hasSideCar ? "Yes" : "No"));
    }
}
public class VehicleInfo {
    public static void main(String[] args) {
        Vehicle genericVehicle = new Vehicle("Generic", "Model", 2020);
        ElectricCar tesla = new ElectricCar("Tesla", "Model S", 2023, 100);
        Motorcycle harley = new Motorcycle("Harley-Davidson", "Sportster", 2022, false);
        System.out.println("--- Starting Vehicles ---");
        genericVehicle.start();
        tesla.start();
        harley.start();
        System.out.println("\n--- Vehicle Information ---");
        genericVehicle.displayInfo();
        tesla.displayInfo();
        harley.displayInfo();
        System.out.println("\n--- Stopping Vehicles ---");
        genericVehicle.stop();
        tesla.stop();
        harley.stop();
        System.out.println("\n--- Polymorphism Demonstration ---");
        Vehicle[] vehicles = {genericVehicle, tesla, harley};
        for (Vehicle v : vehicles) {
            v.start();
        }
    }
}
```

Output:

```
D:\OOP>javac VehicleInfo.java

D:\OOP>java VehicleInfo
--- Starting Vehicles ---
Starting the vehicle...
Starting electric motor silently...
Battery level: 95%
Kickstarting the motorcycle...

--- Vehicle Information ---
Make: Generic, Model: Model, Year: 2020
Make: Tesla, Model: Model S, Year: 2023
Battery Capacity: 100 kWh
Make: Harley-Davidson, Model: Sportster, Year: 2022
Has Sidecar: No

--- Stopping Vehicles ---
Stopping the vehicle...
Stopping the vehicle...
Stopping the vehicle...

--- Polymorphism Demonstration ---
Starting the vehicle...
Starting electric motor silently...
Battery level: 95%
Kickstarting the motorcycle...
```

11.b) University Grading System

Code:

```java
class Course {
    protected String courseName;
    protected int creditHours;
    public Course(String name, int credits) {
        this.courseName = name;
        this.creditHours = credits;
    }
    public String calculateGrade(double score) {
        if (score >= 90) return "A";
        else if (score >= 80) return "B";
        else if (score >= 70) return "C";
        else if (score >= 60) return "D";
        else return "F";
    }
    public void displayCourseInfo() {
        System.out.println(courseName + " (" + creditHours + " credits)");
    }
}
class LabCourse extends Course {
    public LabCourse(String name, int credits) {
        super(name, credits);
    }
    public String calculateGrade(double score) {
        String grade = super.calculateGrade(score);
        if (grade.equals("A") || grade.equals("B")) {
            return grade + "+";
        }
        return grade;
    }
}
class PassFailCourse extends Course {
    public PassFailCourse(String name, int credits) {
        super(name, credits);
    }
    public String calculateGrade(double score) {
        return (score >= 70) ? "Pass" : "Fail";
    }
    public void displayCourseInfo() {
        super.displayCourseInfo();
        System.out.println("Grading: Pass/Fail");
    }
}
public class UniversityGradingSystem {
    public static void main(String[] args) {
        Course math = new Course("Calculus", 4);
        Course physicsLab = new LabCourse("Physics Lab", 2);
        Course ethics = new PassFailCourse("Ethics", 3);
        System.out.println("=== Course Grades ===");
        math.displayCourseInfo();
        System.out.println("Grade: " + math.calculateGrade(85));
        physicsLab.displayCourseInfo();
        System.out.println("Grade: " + physicsLab.calculateGrade(92));

        ethics.displayCourseInfo();
        System.out.println("Grade: " + ethics.calculateGrade(68));
    }
}
```

Output:

```
D:\OOP>javac UniversityGradingSystem.java

D:\OOP>java UniversityGradingSystem
=== Course Grades ===
Calculus (4 credits)
Grade: B
Physics Lab (2 credits)
Grade: A+
Ethics (3 credits)
Grading: Pass/Fail
Grade: Fail
```

12. INTERFACE PROGRAMS

12.a) Employee Attendance System

Code:

```
interface Attendance {
    void markAttendance();
}

class BiometricAttendance implements Attendance {
    public void markAttendance() {
        System.out.println("Attendance marked using fingerprint scanner.");
    }
}

class RFIDAttendance implements Attendance {
    public void markAttendance() {
        System.out.println("Attendance marked using RFID card.");
    }
}

public class EmployeeAttendanceSystem {
    public static void main(String[] args) {
        Attendance bio = new BiometricAttendance();
        Attendance rfid = new RFIDAttendance();

        bio.markAttendance();
        rfid.markAttendance();
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Desktop\OOPs>java EmployeeAttendanceSystem
Attendance marked using fingerprint scanner.
Attendance marked using RFID card.
```

12.b) Shopping Cart System

Code:

```
interface ShoppingCart {
    void addItem(String item);
}

class GroceryCart implements ShoppingCart {
    public void addItem(String item) {
        System.out.println("Grocery item added: " + item);
    }
}

class ElectronicsCart implements ShoppingCart {
    public void addItem(String item) {
        System.out.println("Electronic item added: " + item);
    }
}

public class ShoppingCartSystem {
    public static void main(String[] args) {
        ShoppingCart grocery = new GroceryCart();
        ShoppingCart electronics = new ElectronicsCart();

        grocery.addItem("Milk");
        electronics.addItem("Laptop");
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Desktop\OOPs>javac ShoppingCartSystem.java

C:\Users\ch.sc.u4cse24120\Desktop\OOPs>java ShoppingCartSystem
Grocery item added: Milk
Electronic item added: Laptop
```

12.c) Restaurant Ordering System

Code:

```
interface Order {
    void placeOrder(String item);
}

class OnlineOrder implements Order {
    public void placeOrder(String item) {
        System.out.println("Online order placed for: " + item);
    }
}

class OfflineOrder implements Order {
    public void placeOrder(String item) {
        System.out.println("Offline order placed for: " + item);
    }
}

public class RestaurantOrderingSystem {
    public static void main(String[] args) {
        Order online = new OnlineOrder();
        Order offline = new OfflineOrder();

        online.placeOrder("Pizza");
        offline.placeOrder("Burger");
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Desktop\OOPs>javac RestaurantOrderingSystem.java

C:\Users\ch.sc.u4cse24120\Desktop\OOPs>java RestaurantOrderingSystem
Online order placed for: Pizza
Offline order placed for: Burger
```

12.d) Weather Forecast System

Code:

```
interface WeatherService {
    void getWeatherReport();
}

class APIWeatherService implements WeatherService {
    public void getWeatherReport() {
        System.out.println("Fetching weather report from API...");
    }
}

class SatelliteWeatherService implements WeatherService {
    public void getWeatherReport() {
        System.out.println("Fetching weather report from satellite...");
    }
}

public class WeatherForecastSystem {
    public static void main(String[] args) {
        WeatherService apiService = new APIWeatherService();
        WeatherService satelliteService = new SatelliteWeatherService();

        apiService.getWeatherReport();
        satelliteService.getWeatherReport();
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Desktop\OOPs>javac WeatherForecastSystem.java

C:\Users\ch.sc.u4cse24120\Desktop\OOPs>java WeatherForecastSystem
Fetching weather report from API...
Fetching weather report from satellite...
```

13. ABSTRACT CLASS PROGRAMS

13.a) Database System

Code:

```java
abstract class Database {
    protected boolean isConnected = false;
    public final void executeQuery(String query) {
        connect();
        if (isConnected) {
            String result = processQuery(query);
            System.out.println("Result: " + result);
            disconnect();
        }
    }
    protected abstract void connect();
    protected abstract String processQuery(String query);
    protected void disconnect() {
        isConnected = false;
        System.out.println("Disconnected from database");
    }
}
class MockUsersDatabase extends Database {
    private String[] users = {"Alice", "Bob", "Charlie"};
    protected void connect() {
        System.out.println("Connecting to Users database...");
        isConnected = true;
    }
    protected String processQuery(String query) {
        if (query.equals("GET_ALL_USERS")) {
            return String.join(", ", users);
        }
        return "Invalid query";
    }
}
class MockProductsDatabase extends Database {
    private String[] products = {"Laptop", "Phone", "Tablet"};
    protected void connect() {
        System.out.println("Connecting to Products database...");
        isConnected = true;
    }
    protected String processQuery(String query) {
        if (query.equals("GET_ALL_PRODUCTS")) {
            return String.join(" | ", products);
        }
        return "Invalid query";
    }
}
public class DatabaseSystem {
    public static void main(String[] args) {
        Database usersDB = new MockUsersDatabase();
        Database productsDB = new MockProductsDatabase();

        System.out.println("=== Users Database ===");
        usersDB.executeQuery("GET_ALL_USERS");

        System.out.println("\n=== Products Database ===");
        productsDB.executeQuery("GET_ALL_PRODUCTS");

        System.out.println("\n=== Invalid Query Test ===");
        productsDB.executeQuery("BAD_QUERY");
    }
}
```

Output:

```
D:\OOP>javac DatabaseSystem.java

D:\OOP>java DatabaseSystem
=== Users Database ===
Connecting to Users database...
Result: Alice, Bob, Charlie
Disconnected from database

=== Products Database ===
Connecting to Products database...
Result: Laptop | Phone | Tablet
Disconnected from database

=== Invalid Query Test ===
Connecting to Products database...
Result: Invalid query
Disconnected from database
```

13.b) Payment System

Code:

```java
abstract class PaymentProcessor {
    abstract void processPayment(double amount);
}

class CreditCardPayment extends PaymentProcessor {
    @Override
    void processPayment(double amount) {
        System.out.println("Processing credit card payment: $" + amount);
    }
}

class PayPalPayment extends PaymentProcessor {
    @Override
    void processPayment(double amount) {
        System.out.println("Processing PayPal payment: $" + amount);
    }
}

public class PaymentSystem {
    public static void main(String[] args) {
        PaymentProcessor card = new CreditCardPayment();
        PaymentProcessor paypal = new PayPalPayment();

        card.processPayment(100.50);  // Output: Processing credit card payment: $100.5
        paypal.processPayment(50.25); // Output: Processing PayPal payment: $50.25
    }
}
```

Output:

```
D:\OOP>javac  PaymentSystem.java

D:\OOP>java  PaymentSystem
Processing credit card payment: $100.5
Processing PayPal payment: $50.25
```

13.c) Simple Bank System

Code:

```java
abstract class BankAccount {
    double balance;
    BankAccount(double initialBalance) {
        this.balance = initialBalance;
    }
    abstract void deposit(double amount);
    abstract void withdraw(double amount);
    void showBalance() {
        System.out.println("Balance: $" + balance);
    }
}
class SavingsAccount extends BankAccount {
    SavingsAccount(double initialBalance) {
        super(initialBalance);
    }
    void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: $" + amount);
    }
    void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrew: $" + amount);
        } else {
            System.out.println("Not enough money");
        }
    }
}
class CheckingAccount extends BankAccount {
    CheckingAccount(double initialBalance) {
        super(initialBalance);
    }
    void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: $" + amount);
    }
    void withdraw(double amount) {
        balance -= amount;
        System.out.println("Withdrew: $" + amount);
    }
}
public class SimpleBank {
    public static void main(String[] args) {
        BankAccount savings = new SavingsAccount(100);
        BankAccount checking = new CheckingAccount(200);
        savings.deposit(50);
        savings.withdraw(30);
        savings.showBalance();
        checking.deposit(100);
        checking.withdraw(250); // Can go negative
        checking.showBalance();
    }
}
```

Output:

```
D:\OOP>javac SimpleBank.java

D:\OOP>java SimpleBank
Deposited: $50.0
Withdrew: $30.0
Balance: $120.0
Deposited: $100.0
Withdrew: $250.0
Balance: $50.0
```

13.d) Simple Game System

Code:

```java
abstract class Character {
    String name;
    int health;
    Character(String name, int health) {
        this.name = name;
        this.health = health;
    }
    abstract void attack();
    void takeDamage(int damage) {
        health -= damage;
        System.out.println(name + " took " + damage + " damage!");
    }
}
class Warrior extends Character {
    Warrior(String name) {
        super(name, 100);
    }

    @Override
    void attack() {
        System.out.println(name + " swings a sword!");
    }
}
class Mage extends Character {
    Mage(String name) {
        super(name, 80);
    }
    void attack() {
        System.out.println(name + " casts a spell!");
    }
}
public class SimpleGame {
    public static void main(String[] args) {
        Character hero = new Warrior("Conan");
        Character enemy = new Mage("Dark Mage");
        hero.attack();  // "Conan swings a sword!"
        enemy.attack(); // "Dark Mage casts a spell!"
        hero.takeDamage(20);
        enemy.takeDamage(30);
    }
}
```

Output:

```
D:\OOP>javac SimpleGame.java

D:\OOP>java SimpleGame
Conan swings a sword!
Dark Mage casts a spell!
Conan took 20 damage!
Dark Mage took 30 damage!
```

14. ENCAPSULATION PROGRAMS

14.a) Inventory System

Code:

```java
class Inventory {
    private String itemName;
    private int stockQuantity;
    public Inventory(String itemName, int stockQuantity) {
        this.itemName = itemName;
        this.stockQuantity = stockQuantity;
    }
    public String getItemName() {
        return itemName;
    }
    public int getStockQuantity() {
        return stockQuantity;
    }
    public void addStock(int quantity) {
        if (quantity > 0) {
            stockQuantity += quantity;
            System.out.println(quantity + " items added to inventory.");
        } else {
            System.out.println("Invalid stock quantity.");
        }
    }
    public void removeStock(int quantity) {
        if (quantity > 0 && quantity <= stockQuantity) {
            stockQuantity -= quantity;
            System.out.println(quantity + " items removed from inventory.");
        } else {
            System.out.println("Not enough stock available.");
        }
    }
}
public class InventorySystem {
    public static void main(String[] args) {
        Inventory item = new Inventory("Laptop", 20);

        System.out.println("Item: " + item.getItemName());
        System.out.println("Stock Quantity: " + item.getStockQuantity());
        item.addStock(10);
        item.removeStock(5);
        System.out.println("Updated Stock: " + item.getStockQuantity());
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Desktop\OOP>javac InventorySystem.java

C:\Users\ch.sc.u4cse24120\Desktop\OOP>java InventorySystem
Item: Laptop
Stock Quantity: 20
10 items added to inventory.
5 items removed from inventory.
Updated Stock: 25
```

14.b) Smart Home System

Code:

```java
class SmartHome {
    private boolean lightsOn;
    private boolean doorLocked;

    public SmartHome() {
        this.lightsOn = false;
        this.doorLocked = true;
    }

    public boolean isLightsOn() {
        return lightsOn;
    }

    public boolean isDoorLocked() {
        return doorLocked;
    }

    public void toggleLights() {
        lightsOn = !lightsOn;
        System.out.println("Lights are now " + (lightsOn ? "ON" : "OFF"));
    }

    public void lockDoor(boolean lock) {
        doorLocked = lock;
        System.out.println("Door is now " + (doorLocked ? "LOCKED" : "UNLOCKED"));
    }
}

public class SmartHomeSystem {
    public static void main(String[] args) {
        SmartHome home = new SmartHome();

        home.toggleLights();
        home.lockDoor(false);
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Desktop\OOP>javac SmartHomeSystem.java

C:\Users\ch.sc.u4cse24120\Desktop\OOP>java SmartHomeSystem
Lights are now ON
Door is now UNLOCKED
```

14.c) Ticket Booking System

Code:

```java
class TicketBooking {
    private String movieName;
    private int availableSeats;

    public TicketBooking(String movieName, int availableSeats) {
        this.movieName = movieName;
        this.availableSeats = availableSeats;
    }

    public String getMovieName() {
        return movieName;
    }

    public int getAvailableSeats() {
        return availableSeats;
    }

    public void bookTickets(int seats) {
        if (seats > 0 && seats <= availableSeats) {
            availableSeats -= seats;
            System.out.println(seats + " ticket(s) booked successfully.");
        } else {
            System.out.println("Not enough seats available.");
        }
    }
}

public class TicketBookingSystem {
    public static void main(String[] args) {
        TicketBooking booking = new TicketBooking("Avengers: Endgame", 50);

        System.out.println("Movie: " + booking.getMovieName());
        System.out.println("Available Seats: " + booking.getAvailableSeats());

        booking.bookTickets(5);
        System.out.println("Updated Available Seats: " + booking.getAvailableSeats());
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Desktop\OOP>javac TicketBookingSystem.java

C:\Users\ch.sc.u4cse24120\Desktop\OOP>java TicketBookingSystem
Movie: Avengers: Endgame
Available Seats: 50
5 ticket(s) booked successfully.
Updated Available Seats: 45
```

14.d) Video Streaming

Code:

```java
class Video {
    private String title;
    private boolean isPremium;

    public Video(String title, boolean isPremium) {
        this.title = title;
        this.isPremium = isPremium;
    }

    public String getTitle() {
        return title;
    }

    public boolean isPremium() {
        return isPremium;
    }

    public void watchVideo(boolean hasSubscription) {
        if (!isPremium || hasSubscription) {
            System.out.println("Playing: " + title);
        } else {
            System.out.println("This is a premium video. Please subscribe to watch.");
        }
    }
}

public class VideoStreaming {
    public static void main(String[] args) {
        Video video1 = new Video("Tech Trends 2025", false);
        Video video2 = new Video("Exclusive Movie Premiere", true);

        System.out.println("Video: " + video1.getTitle());
        video1.watchVideo(false);

        System.out.println("Video: " + video2.getTitle());
        video2.watchVideo(false);
        video2.watchVideo(true);
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Desktop\OOP>javac VideoStreaming.java

C:\Users\ch.sc.u4cse24120\Desktop\OOP>java VideoStreaming
Video: Tech Trends 2025
Playing: Tech Trends 2025
Video: Exclusive Movie Premiere
This is a premium video. Please subscribe to watch.
Playing: Exclusive Movie Premiere
```

15) Packages Programs

15.a) File Stats Calculator

File Operations Package:

```java
package file.utils;

import java.io.*;
import java.nio.file.*;
import java.util.*;

public class FileOperations {

    // Basic file operations
    public static void createFile(String filePath) throws IOException {
        File file = new File(filePath);
        if (file.createNewFile()) {
            System.out.println("File created: " + file.getName());
        } else {
            System.out.println("File already exists.");
        }
    }

    public static void writeToFile(String filePath, String content) throws IOException {
        Files.write(Paths.get(filePath), content.getBytes(), StandardOpenOption.CREATE);
    }

    public static String readFile(String filePath) throws IOException {
        return new String(Files.readAllBytes(Paths.get(filePath)));
    }

    // Intermediate file operations
    public static void appendToFile(String filePath, String content) throws IOException {
        Files.write(Paths.get(filePath), content.getBytes(), StandardOpenOption.APPEND);
    }

    public static int countLines(String filePath) throws IOException {
        try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
            int lines = 0;
            while (reader.readLine() != null) lines++;
            return lines;
        }
    }

    public static List<String> findFilesWithExtension(String directory, String extension) throws IOException {
        List<String> result = new ArrayList<>();
        try (DirectoryStream<Path> stream = Files.newDirectoryStream(Paths.get(directory), "*." + extension)) {
            for (Path path : stream) {
                if (!Files.isDirectory(path)) {
                    result.add(path.getFileName().toString());
                }
            }
        }
        return result;
    }

    public static void copyFile(String sourcePath, String destPath) throws IOException {
        Files.copy(Paths.get(sourcePath), Paths.get(destPath), StandardCopyOption.REPLACE_EXISTING);
    }
}
```

String Operations Package:

```java
package string.utils;

import java.util.Arrays;


public class StringOperations {

    public static String reverse(String input) {
        return new StringBuilder(input).reverse().toString();
    }

    public static int countVowels(String input) {
        int count = 0;
        String vowels = "aeiouAEIOU";
        for (char c : input.toCharArray()) {
            if (vowels.indexOf(c) != -1) {
                count++;
            }
        }
        return count;
    }

    // Intermediate string operations
    public static boolean isPalindrome(String input) {
        String clean = input.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();
        return clean.equals(reverse(clean));
    }

    public static String[] splitIntoWords(String input) {
        return input.trim().split("\\s+");
    }

    public static String removeDuplicates(String input) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < input.length(); i++) {
            if (result.indexOf(String.valueOf(input.charAt(i))) == -1) {
                result.append(input.charAt(i));
            }
        }
        return result.toString();
    }

    public static String capitalizeWords(String input) {
        String[] words = splitIntoWords(input);
        for (int i = 0; i < words.length; i++) {
            if (words[i].length() > 0) {
                words[i] = words[i].substring(0, 1).toUpperCase() +
                           words[i].substring(1).toLowerCase();
            }
        }
        return String.join(" ", words);
    }
}
```

Program Code:

```java
import file.utils.FileOperations;
import string.utils.StringOperations;
import java.io.IOException;

public class FileStatsCalculator {
    public static void main(String[] args) {
        String filePath = "sample.txt";
        try {
            // Create and write to file
            FileOperations.createFile(filePath);
            FileOperations.writeToFile(filePath, "This is a sample text file.\n");
            FileOperations.appendToFile(filePath, "It contains multiple lines.\n");
            FileOperations.appendToFile(filePath, "For demonstration purposes.\n");

            // Read and analyze file
            String content = FileOperations.readFile(filePath);
            System.out.println("File content:\n" + content);

            System.out.println("\nFile statistics:");
            System.out.println("Line count: " + FileOperations.countLines(filePath));
            System.out.println("Vowel count: " + StringOperations.countVowels(content));
            System.out.println("Character count: " + content.length());
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
D:\OOP\UserDefined>javac -d . StringOperations.java

D:\OOP\UserDefined>javac -d . FileOperations.java

D:\OOP\UserDefined>javac FileStatsCalculator.java

D:\OOP\UserDefined>java FileStatsCalculator
File already exists.
File content:
This is a sample text file.
It contains multiple lines.
For demonstration purposes.
It contains multiple lines.
For demonstration purposes.


File statistics:
Line count: 5
Vowel count: 44
Character count: 140
```

15.b) Math String Demo

Math Operations Package:

```java
package math.utils;

public class MathOperations {

    // Basic arithmetic operations
    public static int add(int a, int b) {
        return a + b;
    }

    public static int subtract(int a, int b) {
        return a - b;
    }

    public static int multiply(int a, int b) {
        return a * b;
    }

    public static double divide(int a, int b) {
        if (b == 0) {
            throw new IllegalArgumentException("Cannot divide by zero");
        }
        return (double) a / b;
    }

    // Intermediate operations
    public static int factorial(int n) {
        if (n < 0) {
            throw new IllegalArgumentException("Factorial is not defined for negative numbers");
        }
        int result = 1;
        for (int i = 2; i <= n; i++) {
            result *= i;
        }
        return result;
    }

    public static boolean isPrime(int number) {
        if (number <= 1) return false;
        if (number == 2) return true;
        if (number % 2 == 0) return false;

        for (int i = 3; i * i <= number; i += 2) {
            if (number % i == 0) {
                return false;
            }
        }
        return true;
    }

    public static int gcd(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }
}
```

String Operations Package:

```java
package string.utils;

import java.util.Arrays;


public class StringOperations {

    public static String reverse(String input) {
        return new StringBuilder(input).reverse().toString();
    }

    public static int countVowels(String input) {
        int count = 0;
        String vowels = "aeiouAEIOU";
        for (char c : input.toCharArray()) {
            if (vowels.indexOf(c) != -1) {
                count++;
            }
        }
        return count;
    }

    // Intermediate string operations
    public static boolean isPalindrome(String input) {
        String clean = input.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();
        return clean.equals(reverse(clean));
    }

    public static String[] splitIntoWords(String input) {
        return input.trim().split("\\s+");
    }

    public static String removeDuplicates(String input) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < input.length(); i++) {
            if (result.indexOf(String.valueOf(input.charAt(i))) == -1) {
                result.append(input.charAt(i));
            }
        }
        return result.toString();
    }

    public static String capitalizeWords(String input) {
        String[] words = splitIntoWords(input);
        for (int i = 0; i < words.length; i++) {
            if (words[i].length() > 0) {
                words[i] = words[i].substring(0, 1).toUpperCase() +
                           words[i].substring(1).toLowerCase();
            }
        }
        return String.join(" ", words);
    }
}
```

Program Code:

```java
import math.utils.MathOperations;
import string.utils.StringOperations;

public class MathStringDemo {
    public static void main(String[] args) {
        // Math operations
        int a = 12, b = 18;
        System.out.println("GCD of " + a + " and " + b + ": " + MathOperations.gcd(a, b));
        System.out.println("Factorial of 5: " + MathOperations.factorial(5));

        // String operations
        String text = "Java Programming";
        System.out.println("\nOriginal string: " + text);
        System.out.println("Reversed: " + StringOperations.reverse(text));
        System.out.println("Vowel count: " + StringOperations.countVowels(text));
        System.out.println("Capitalized words: " + StringOperations.capitalizeWords("hello world"));
    }
}
```

Output:

```
D:\OOP\UserDefined>javac -d . MathOperations.java

D:\OOP\UserDefined>javac -d . StringOperations.java

D:\OOP\UserDefined>javac MathStringDemo.java

D:\OOP\UserDefined>java MathStringDemo
GCD of 12 and 18: 6
Factorial of 5: 120

Original string: Java Programming
Reversed: gnimmargorP avaJ
Vowel count: 5
Capitalized words: Hello World
```

15.c) Collection Example

Code:

```java
import java.util.*;
import java.time.*;              // Added package for date/time operations
import java.util.stream.*;       // Added package for stream operations

public class CollectionExample {
    public static void main(String[] args) {
        // List example
        List<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Orange");
        System.out.println("Fruits: " + fruits);

        // Map example
        Map<String, Integer> inventory = new HashMap<>();
        inventory.put("Apples", 50);
        inventory.put("Oranges", 75);
        System.out.println("Inventory: " + inventory);

        // Sort list
        Collections.sort(fruits);
        System.out.println("Sorted fruits: " + fruits);

        // Using java.time package (LocalDate)
        LocalDate today = LocalDate.now();
        System.out.println("\nToday's date: " + today);
        System.out.println("Day of week: " + today.getDayOfWeek());

        // Using java.util.stream package
        System.out.println("\nStream operations:");
        List<String> filteredFruits = fruits.stream()
            .filter(f -> f.startsWith("A"))
            .collect(Collectors.toList());
        System.out.println("Fruits starting with 'A': " + filteredFruits);

        long count = fruits.stream()
            .filter(f -> f.length() > 5)
            .count();
        System.out.println("Number of fruits with more than 5 letters: " + count);
    }
}
```

Output:

```
D:\OOP>javac CollectionExample.java

D:\OOP>java CollectionExample
Fruits: [Apple, Banana, Orange]
Inventory: {Apples=50, Oranges=75}
Sorted fruits: [Apple, Banana, Orange]
```

15.d) File Example

Code:

```java
import java.io.*;
import java.nio.file.*;       // For Files and Path classes
import java.nio.file.attribute.FileTime;
import java.util.*;           // For Scanner and List
import java.time.*;           // For date/time operations

public class FileExample {
    public static void main(String[] args) {
        try {
            Path filePath = Paths.get("example.txt");

            FileWriter writer = new FileWriter("example.txt");
            writer.write("Hello, Java File I/O!\n");
            writer.close();
            Files.write(filePath, "Additional line using NIO\n".getBytes(), StandardOpenOption.APPEND);
            List<String> lines = Arrays.asList("Third line from List", "Fourth line from List");
            Files.write(filePath, lines, StandardOpenOption.APPEND);
            System.out.println("\nFile content:");
            BufferedReader reader = new BufferedReader(new FileReader("example.txt"));
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println("1. " + line);
            }
            reader.close();
            System.out.println("\nReading with Files.readAllLines:");
            List<String> fileContent = Files.readAllLines(filePath);
            for (String content : fileContent) {
                System.out.println("2. " + content);
            }
            System.out.println("\nReading with Scanner:");
            Scanner scanner = new Scanner(filePath);
            while (scanner.hasNextLine()) {
                System.out.println("3. " + scanner.nextLine());
            }
            scanner.close();
            FileTime lastModified = Files.getLastModifiedTime(filePath);
            System.out.println("\nFile last modified: " + lastModified.toInstant());
            System.out.println("File size: " + Files.size(filePath) + " bytes");
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
D:\OOP>javac FileExample.java

D:\OOP>java FileExample
File content: Hello, Java File I/O!
```

16. EXCEPTION HANDLING PROGRAMS

16.a) Multiple Catch Example

Code:

```java
public class MultipleCatchExample {
    public static void main(String[] args) {
        try {
            int[] numbers = {1, 2, 3};
            System.out.println(numbers[5]); // Causes ArrayIndexOutOf-BoundsException
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic error occurred.");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Array index out of bounds.");
        }
    }
}
```

Output:

```
D:\OOP>javac MultipleCatchExample.java

D:\OOP>java MultipleCatchExample
Error: Array index out of bounds.
```

16.b) Rethrow Example

Code:

```java
public class RethrowExample {
    public static void riskyMethod() throws Exception {
        throw new Exception("Something went wrong!");
    }

    public static void main(String[] args) {
        try {
            riskyMethod();
        } catch (Exception e) {
            System.out.println("Caught an error: " + e.getMessage());
            throw new RuntimeException("Re-throwing the exception!", e);
        }
    }
}
```

Output:

```
D:\OOP>javac RethrowExample.java

D:\OOP>java RethrowExample
Caught an error: Something went wrong!
Exception in thread "main" java.lang.RuntimeException: Re-throwing the exception!
        at RethrowExample.main(RethrowExample.java:11)
Caused by: java.lang.Exception: Something went wrong!
        at RethrowExample.riskyMethod(RethrowExample.java:3)
        at RethrowExample.main(RethrowExample.java:8)
```

16.c) Division Example

Code:

```java
public class DivisionExample {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero!");
        }
    }
}
```

Output:

```
D:\OOP>javac DivisionExample.java

D:\OOP>java DivisionExample
Cannot divide by zero!
```

16.d) String Index

Code:

```java
public class StringIndexOutOfBoundsExample {
    public static void main(String[] args) {
        try {
            String str = "Hello";
            System.out.println(str.charAt(10));
        } catch (StringIndexOutOfBoundsException e) {
            System.out.println("String index is out of bounds!");
        }
    }
}

```

Output:

```
D:\OOP>javac StringIndexOutOfBoundsExample.java

D:\OOP>java StringIndexOutOfBoundsExample
String index is out of bounds!
```

17. FILE HANDLING PROGRAMS

17.a) Check Hidden File

Code:

```java
import java.io.File;

public class CheckHiddenFile {
    public static void main(String[] args) {
        File file = new File("test.txt");
        if (file.isHidden()) {
            System.out.println("The file is hidden.");
        } else {
            System.out.println("The file is not hidden.");
        }
    }
}

```

Output:

```
C:\Users\ch.sc.u4cse24120\Desktop\OOPs>javac CheckHiddenFile.java

C:\Users\ch.sc.u4cse24120\Desktop\OOPs>java CheckHiddenFile
The file is not hidden.
```

17.b) Check Empty File

Code:

```java
import java.io.File;

public class CheckEmptyFile {
    public static void main(String[] args) {
        File file = new File("test.txt");
        if (file.length() == 0) {
            System.out.println("The file is empty.");
        } else {
            System.out.println("The file is not empty.");
        }
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Desktop\OOPs>javac CheckEmptyFile.java

C:\Users\ch.sc.u4cse24120\Desktop\OOPs>java CheckEmptyFile
The file is not empty.
```

17.c) Write File NIO

Code:

```java
import java.nio.file.*;
import java.io.IOException;
import java.util.Arrays;

public class WriteFileNIO {
    public static void main(String[] args) {
        try {
            Files.write(Paths.get("example.txt"), Arrays.asList("Hello", "Java NIO File Handling"), StandardOpenOption.APPEND);
            System.out.println("Data written using NIO.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Desktop\OOPs>javac WriteFileNIO.java

C:\Users\ch.sc.u4cse24120\Desktop\OOPs>java WriteFileNIO
Data written using NIO.
```

```
OOPs
Hello
Java NIO File Handling
```

17.d) File Operations

Code:

```java
import java.io.File;

public class FileOperations {
    public static void main(String[] args) {
        File file = new File("example.txt");

        if (file.exists()) {
            System.out.println("File exists. Size: " + file.length() + " bytes.");

            if (file.delete()) {
                System.out.println("File deleted successfully.");
            } else {
                System.out.println("Failed to delete the file.");
            }
        } else {
            System.out.println("File does not exist.");
        }
    }
}
```

Output:

```
C:\Users\ch.sc.u4cse24120\Desktop\OOPs>javac FileOperations.java

C:\Users\ch.sc.u4cse24120\Desktop\OOPs>java FileOperations
File exists. Size: 4 bytes.
File deleted successfully.
```