Nagar Yuwak Shikshan Sanstha's

Yeshwantrao Chavan College of Engineering

(An Autonomous Institution affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)
Hingna Road, Wanadongri, Nagpur - 441 110
NAAC A++
Ph.: 07104-237919, 234623, 329249, 329250 Fax: 07104-232376, Website: www.ycce.edu

Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

Session 2025-2026

Vision: Dream of where you want.	**Mission:** Means to achieve Vision

Program Educational Objectives of the program (PEO): (broad statements that describe the professional and career accomplishments)

PEO1	**Preparation**	**P: Preparation**	**Pep-CL abbreviation pronounce as Pep-si-lL easy to recall**
PEO2	**Core Competence**	**E: Environment (Learning Environment)**	
PEO3	**Breadth**	**P: Professionalism**	
PEO4	**Professionalism**	**C: Core Competence**	
PEO5	**Learning Environment**	**L: Breadth (Learning in diverse areas)**	

Program Outcomes (PO): (statements that describe what a student should be able to do and know by the end of a program)

Keywords of POs:
Engineering knowledge, Problem analysis, Design/development of solutions, Conduct Investigations of Complex Problems, Engineering Tool Usage, The Engineer and The World, Ethics, Individual and Collaborative Team work, Communication, Project Management and Finance, Life-Long Learning

PSO Keywords: Cutting edge technologies, Research

"I am an engineer, and I know how to apply engineering knowledge to investigate, analyse and design solutions to complex problems using tools for entire world following all ethics in a collaborative way with proper management skills throughout my life." *to contribute to the development of cutting-edge technologies and Research*.

Integrity: I will adhere to the Laboratory Code of Conduct and ethics in its entirety.

Name and Signature of Student and Date
(Signature and Date in Handwritten)
Nikhil Gourkar

Gourkar
15/09/25

Session	**2025-26 (ODD)**	**Course Name**	**Lab : Java Stack**
Semester	**5**	**Course Code**	**CT**
Roll No	**59**	**Name of Student**	**Nikhil Gourkar**

Practical Number	**2**
Course Outcome	Proper Understanding of Basic Java programs covering loops , arrays and conditionals and implementation of programs.
Aim	To implement OOPS methods like polymorphism , method overloading and overriding .
Problem Definition	**Problem Statement 1:** Build a system to manage different types of employees in a company. All employees share some common attributes, but certain types of employees have additional specific information. The common attributes are Name and employeeId and there is a common method DisplayDetails(). There should be a derived class FullTimeEmployee with an additional attribute salary and an overridden DisplayDetails() method that also shows the salary. There should also be a derived class PartTimeEmployee with additional attributes hourlyRate and hoursWorked and an overridden DisplayDetails() method that also shows these additional attributes. The program should display all the information of all types of employees. **Problem Statement 2:** Implement a payment gateway system that allows users to make payments through different payment methods such as credit card, debit card and digital wallet. Each payment method processes the payment differently but they all share a common interface. There should be an abstract class PaymentMethod with an abstract method processPayment() and derived classes CreditCardPayment, DebitCardPayment and DigitalWalletPayment implementing their own version of processPayment() to simulate the logic of each payment type. The main class should have a method makePayment that accepts any payment type and processes it.
Theory (100 words)	**Inheritance:** Allows a class to acquire properties and methods of another class, promoting code reuse. **Method Overriding:** A subclass provides its own implementation of a superclass method, enabling runtime polymorphism. **Abstract Classes:** Classes that cannot be instantiated and may contain abstract methods which must be implemented by subclasses. **Polymorphism:** One reference type can refer to objects of different

classes and call overridden methods dynamically.
Encapsulation: Bundling data and methods in a class and restricting direct access using access modifiers.

Procedure and Execution

(100 Words)

Algorithm:
Employee Management System

1. Start.
2. Create a base class Employee with attributes name and employeeId and method DisplayDetails().
3. Create derived class FullTimeEmployee with attribute salary and override DisplayDetails() to include salary.
4. Create derived class PartTimeEmployee with attributes hourlyRate and hoursWorked and override DisplayDetails() to include them.
5. In the main method, create objects of FullTimeEmployee and PartTimeEmployee.
6. Call DisplayDetails() for each object to display all employee information.
7. End.

Payment Gateway System

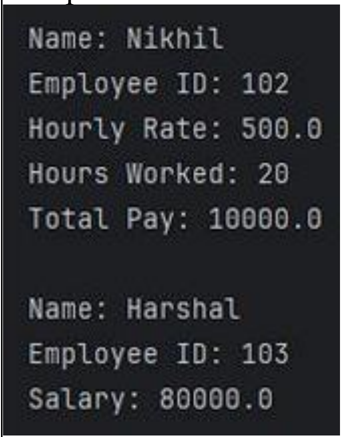
1. Start.
2. Create an abstract class PaymentMethod with an abstract method processPayment().
3. Create derived classes CreditCardPayment, DebitCardPayment, and DigitalWalletPayment, and implement processPayment() in each.
4. In the main method, create objects of each payment type.
5. Write a method makePayment() that accepts a PaymentMethod object and calls processPayment().
6. Use makePayment() to process payments for different methods.
7. End.

Code:

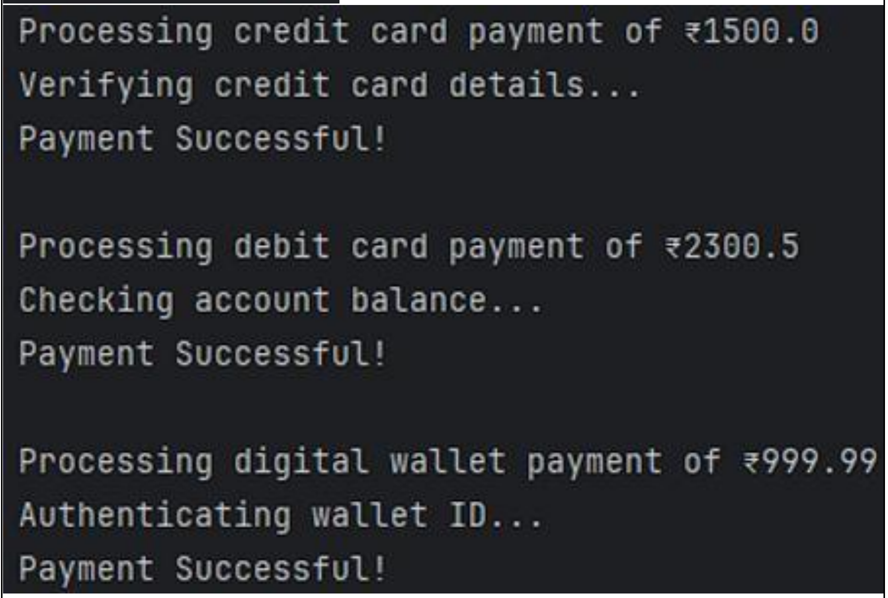
```
1) abstract class Employee {
   protected String name;
   protected int employeeId;

   public Employee(String name, int employeeId) {
      this.name = name;
      this.employeeId = employeeId;
   }

   public void displayDetails() {
      System.out.println("Name: " + name);
      System.out.println("Employee ID: " + employeeId);
   }
}

class FTE extends Employee {
   private double salary;

   public FTE(String name, int employeeId, double salary) {
        super(name, employeeId);
        this.salary = salary;
    }

    @Override
    public void displayDetails() {
        System.out.println("--- Full-Time Employee ---");
        super.displayDetails(); // call base class method
        System.out.println("Salary: " + salary);
        System.out.println();
    }
}

class PTE extends Employee {
    private double hourlyRate;
    private int hoursWorked;

    public PTE(String name, int employeeId, double hourlyRate, int
hoursWorked) {
        super(name, employeeId);
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
    }

    @Override
    public void displayDetails() {
        System.out.println("--- Part-Time Employee ---");
        super.displayDetails();
        System.out.println("Hourly Rate: " + hourlyRate);
        System.out.println("Hours Worked: " + hoursWorked);
        System.out.println("Total Pay: " + (hourlyRate *
hoursWorked));
        System.out.println();
    }
}

public class EmployeeDemo {
    public static void main(String[] args) {
        Employee[] employees = {
                new FTE("Mithilesh", 101, 60000),
                new PTE("Nikhil", 102, 500, 20),
                new FTE("Harshal", 103, 80000),
                new PTE("Vedant", 104, 450, 15)
        };

        for (Employee emp : employees) {
            emp.displayDetails();
        }
    }
}
2) abstract class PaymentMethod {
    public abstract void processPayment(double amount);
}
```

```
class CCP extends PaymentMethod {
    @Override
    public void processPayment(double amount) {
        System.out.println("Processing credit card payment of ₹" +
amount);
        System.out.println("Verifying credit card details...");
        System.out.println("Payment Successful!\n");
    }
}

class DCP extends PaymentMethod {
    @Override
    public void processPayment(double amount) {
        System.out.println("Processing debit card payment of ₹" +
amount);
        System.out.println("Checking account balance...");
        System.out.println("Payment Successful!\n");
    }
}

class DWP extends PaymentMethod {
    @Override
    public void processPayment(double amount) {
        System.out.println("Processing digital wallet payment of ₹" +
amount);
        System.out.println("Authenticating wallet ID...");
        System.out.println("Payment Successful!\n");
    }
}

public class PaymentDemo {
    // Method accepts any payment type and processes it
(polymorphism)
    public static void makePayment(PaymentMethod method, double
amount) {
        method.processPayment(amount);
    }

    public static void main(String[] args) {
        PaymentMethod credit = new CCP();
        PaymentMethod debit = new DCP();
        PaymentMethod wallet = new DWP();

        makePayment(credit, 1500.00);
        makePayment(debit, 2300.50);
        makePayment(wallet, 999.99);
    }
}
```

Output:

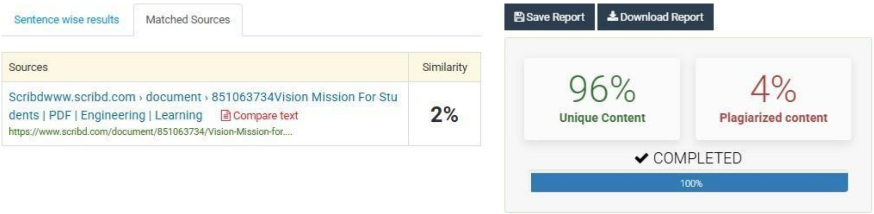


```
Name: Nikhil
Employee ID: 102
Hourly Rate: 500.0
Hours Worked: 20
Total Pay: 10000.0

Name: Harshal
Employee ID: 103
Salary: 80000.0
```



```
Processing credit card payment of ₹1500.0
Verifying credit card details...
Payment Successful!

Processing debit card payment of ₹2300.5
Checking account balance...
Payment Successful!

Processing digital wallet payment of ₹999.99
Authenticating wallet ID...
Payment Successful!
```

Output Analysis	**1. Employee** When the program runs, it creates objects of FullTimeEmployee and PartTimeEmployee and stores them in an array of type Employee. The displayDetails() method is called for each object. **2. Payment** When the program runs, it calls the makePayment() method with different payment type objects (CreditCardPayment, DebitCardPayment, DigitalWalletPayment). Because of **polymorphism**, the method processPayment() of the actual object type is executed.
Link of student Github profile where lab assignment has	https://github.com/Nikhil07Gourkar/Java-Lab

been uploaded	
Conclusion	The programs successfully demonstrate the concepts of inheritance, method overriding, abstract classes, and polymorphism in Java. They show how different types of objects can be handled uniformly while retaining their specific behaviors, making the system flexible, reusable, and easy to extend
Plag Report (Similarity index < 12%)	 Sentence wise results Matched Sources Save Report Download Report Sources — Similarity Scribdwww.scribd.com › document › 851063734Vision Mission For Students | PDF | Engineering | Learning Compare text https://www.scribd.com/document/851063734/Vision-Mission-for.... — 2% 96% Unique Content 4% Plagiarized content ✓ COMPLETED 100%
Date	15/09/25