

**Java OOP Introduction: Classes and Objects - Lab Practice Java Best Programming Practice**

1. Use Variables including for Fixed, User Inputs, and Results

2. Use Methods instead of writing code in the main() function

3. Proper naming conventions for all variables and methods

4. Proper Program Name and Class Name

5. Handle Checked and Unchecked Exceptions wherever possible

6. Proper Method Name which indicates action taking inputs and providing result

**Sample Program: Create a program to manage student information using classes and objects**

**Task**: Take user input for student details and demonstrate object creation and method usage a. Take user input for Student Name, Roll Number, and Marks in 3 subjects b. Write a class Student with proper encapsulation (private variables, public methods) c. Create methods to calculate total, average, and grade d. Call the methods in main and display the formatted result

// Program to manage student information using OOP concepts import java.util.Scanner;

class Student {

// Private instance variables (Encapsulation)

private String studentName;

private int rollNumber;

private double physics, chemistry, mathematics;

// Static variable to count total students

private static int totalStudents = 0;

// Constructor to initialize student object

public Student(String name, int roll, double phy, double chem, double math) {

this.studentName = name;

this.rollNumber = roll;

this.physics = phy;

this.chemistry = chem;

1



this.mathematics = math;

totalStudents++; // Increment static counter

}

// Method to calculate total marks

public double calculateTotal() {

return physics + chemistry + mathematics;

}

// Method to calculate average marks

public double calculateAverage() {

return calculateTotal() / 3.0;

}

// Method to determine grade based on average

public char calculateGrade() {

double avg = calculateAverage();

if (avg >= 90) return 'A';

else if (avg >= 80) return 'B';

else if (avg >= 70) return 'C';

else if (avg >= 60) return 'D';

else return 'F';

}

// Method to display student information

public void displayStudentInfo() {

System.out.println("Name: " + studentName);

System.out.println("Roll: " + rollNumber);

System.out.println("Total: " + calculateTotal()); System.out.println("Average: " + String.format("%.2f", calculateAverage()));

System.out.println("Grade: " + calculateGrade()); }

// Static method to get total student count

public static int getTotalStudents() {

return totalStudents;

}

}

2



**Lab Practice Problems (Any Four)**

**1. Write a program to create a Bank Account management system without using built-in collection classes**

**Hint** =>

a. Create a BankAccount class with private instance variables: accountNumber (String), accountHolderName (String), balance (double), and a static variable totalAccounts (int)

b. Create a constructor that takes account holder name and initial deposit, automatically generates account number using a static counter

c. Create instance methods: deposit(double amount), withdraw(double amount), checkBalance() with proper validation for negative amounts and insufficient funds d. Create static methods: getTotalAccounts(), generateAccountNumber() that returns a unique account number like "ACC001", "ACC002"

e. Create a method displayAccountInfo() to show all account details in a formatted manner

f. In the main method, create an array of BankAccount objects, demonstrate creating multiple accounts, performing transactions, and showing the difference between static and instance variables

SOLUTION:

class BankAccount {

private String accountNumber;

private String accountHolderName;

private double balance;

private static int totalAccounts = 0;

public BankAccount(String accountHolderName, double initialDeposit) {

if (initialDeposit < 0) {

System.out.println("Initial deposit cannot be negative. Setting to 0.");

initialDeposit = 0;

}

this.accountHolderName = accountHolderName;

this.balance = initialDeposit;

this.accountNumber = generateAccountNumber();

}

private static String generateAccountNumber() {

totalAccounts++;

return String.format("ACC%03d", totalAccounts);

}

public void deposit(double amount) {

if (amount <= 0) {

System.out.println("Deposit amount must be greater than 0.");

return;

}

balance += amount;

System.out.println("Rs." + amount + " deposited successfully.");

}

public void withdraw(double amount) {

if (amount <= 0) {

System.out.println("Withdrawal amount must be greater than 0.");

return;

}

if (amount > balance) {

System.out.println("Insufficient balance. Withdrawal failed!");

return;

}

balance -= amount;

System.out.println("Rs." + amount + " withdrawn successfully.");

}

public void checkBalance() {

System.out.println("Current Balance: Rs." + balance);

}

public void displayAccountInfo() {

System.out.println("\n--- Account Details ---");

System.out.println("Account Number : " + accountNumber);

System.out.println("Account Holder : " + accountHolderName);

System.out.println("Balance : Rs." + balance);

System.out.println("------------------------");

}

public static int getTotalAccounts() {

return totalAccounts;

}

}

public class BankManagement {

public static void main(String[] args) {

BankAccount[] accounts = new BankAccount[3];

accounts[0] = new BankAccount("Aayush Rai", 5000);

accounts[1] = new BankAccount("Rohan Sharma", 10000);

accounts[2] = new BankAccount("Priya Singh", 2000);

accounts[0].deposit(2000);

accounts[0].withdraw(1000);

accounts[0].checkBalance();

accounts[1].withdraw(12000);

accounts[2].deposit(3000);

for (int i = 0; i < accounts.length; i++) {

accounts[i].displayAccountInfo();

}

System.out.println("\nTotal Bank Accounts Created: " + BankAccount.getTotalAccounts());

}

}

OUTPUT:

Rs.2000.0 deposited successfully.

Rs.1000.0 withdrawn successfully.

Current Balance: Rs.6000.0

Insufficient balance. Withdrawal failed!

Rs.3000.0 deposited successfully.

--- Account Details ---

Account Number : ACC001

Account Holder : Aayush Rai

Balance : Rs.6000.0

------------------------

--- Account Details ---

Account Number : ACC002

Account Holder : Rohan Sharma

Balance : Rs.10000.0

------------------------

--- Account Details ---

Account Number : ACC003

Account Holder : Priya Singh

Balance : Rs.5000.0

------------------------

Total Bank Accounts Created: 3

**2.** Writ**e a program to create a Library Book management system, demonstrating object relationships**

**Hint** =>

a. Create a Book class with private variables: bookId (String), title (String), author (String), isAvailable (boolean), and static variables totalBooks (int), availableBooks (int)

b. Create a constructor for Book class and methods: issueBook(), returnBook(), displayBookInfo()

c. Create a Member class with private variables: memberId (String), memberName (String), booksIssued (String array to store book IDs), bookCount (int to track number of books issued)

d. Create methods in Member class: borrowBook(Book book) which checks if book is available and updates both book and member status, returnBook(String bookId, Book[] books) to return a specific book

e. Create static methods in both classes to generate unique IDs and track statistics

f. In main, create arrays of Book and Member objects, demonstrate borrowing and returning books, showing how objects interact with each other

SOLUTION:

class Book {

private String bookId, title, author;

private boolean isAvailable;

private static int totalBooks = 0, availableBooks = 0;

public Book(String bookId, String title, String author) {

this.bookId = bookId;

this.title = title;

this.author = author;

this.isAvailable = true;

totalBooks++;

availableBooks++;

}

public String getBookId() {

return bookId;

}

public boolean isAvailable() {

return isAvailable;

}

public void issueBook() {

if (isAvailable) {

isAvailable = false;

availableBooks--;

System.out.println(title + " issued.");

} else System.out.println(title + " is not available.");

}

public void returnBook() {

if (!isAvailable) {

isAvailable = true;

availableBooks++;

System.out.println(title + " returned.");

}

}

public void displayBookInfo() {

System.out.println(bookId + " | " + title + " | " + author + " | " + (isAvailable ? "Available" : "Issued"));

}

public static int getTotalBooks() {

return totalBooks;

}

public static int getAvailableBooks() {

return availableBooks;

}

}

class Member {

private String memberId, memberName;

private String[] booksIssued;

private int bookCount;

public Member(String memberId, String memberName, int maxBooks) {

this.memberId = memberId;

this.memberName = memberName;

this.booksIssued = new String[maxBooks];

this.bookCount = 0;

}

public void borrowBook(Book book) {

if (book.isAvailable() && bookCount < booksIssued.length) {

book.issueBook();

booksIssued[bookCount++] = book.getBookId();

} else {

System.out.println(memberName + " cannot borrow " + book.getBookId());

}

}

public void returnBook(String bookId, Book[] books) {

for (int i = 0; i < bookCount; i++) {

if (booksIssued[i].equals(bookId)) {

for (Book b : books) {

if (b.getBookId().equals(bookId)) {

b.returnBook();

booksIssued[i] = booksIssued[--bookCount];

booksIssued[bookCount] = null;

return;

}

}

}

}

System.out.println(memberName + " hasn't issued " + bookId);

}

public void displayMemberInfo() {

System.out.print(memberId + " | " + memberName + " | Books: ");

if (bookCount == 0) System.out.println("None");

else {

for (int i = 0; i < bookCount; i++) System.out.print(booksIssued[i] + " ");

System.out.println();

}

}

}

public class LibraryManagement {

public static void main(String[] args) {

Book[] books = {

new Book("B001", "Java Basics", "James"),

new Book("B002", "DSA", "Karumanchi"),

new Book("B003", "OS", "Galvin")

};

Member[] members = {

new Member("M001", "Aayush Rai", 2),

new Member("M002", "Rohan Sharma", 2)

};

members[0].borrowBook(books[0]);

members[0].borrowBook(books[1]);

members[1].borrowBook(books[1]);

members[1].borrowBook(books[2]);

members[0].returnBook("B001", books);

members[1].returnBook("B002", books);

System.out.println("\n--- Books ---");

for (Book b : books) b.displayBookInfo();

System.out.println("\n--- Members ---");

for (Member m : members) m.displayMemberInfo();

System.out.println("\nTotal Books: " + Book.getTotalBooks());

System.out.println("Available Books: " + Book.getAvailableBooks());

}

}

OUTPUT:

Java Basics issued.

DSA issued.

Rohan Sharma cannot borrow B002

OS issued.

Java Basics returned.

Rohan Sharma hasn't issued B002

--- Books ---

B001 | Java Basics | James | Available

B002 | DSA | Karumanchi | Issued

B003 | OS | Galvin | Issued

--- Members ---

M001 | Aayush Rai | Books: B002

M002 | Rohan Sharma | Books: B003

Total Books: 3

Available Books: 1

3



**3. Write a program to create an Employee payroll system with different employee types using method overloading**

**Hint** =>

a. Create an Employee class with private variables: empId (String), empName (String), department (String), baseSalary (double), empType (String), and static variable totalEmployees (int)

b. Create multiple constructors for different employee types: constructor for full-time employees, constructor for part-time employees, constructor for contract employees

c. Create overloaded methods calculateSalary(): one for full-time (base salary + bonus), one for part-time (hourly rate × hours), one for contract (fixed amount)

d. Create overloaded methods calculateTax() with different tax rates for different employee types

e. Create methods generatePaySlip() to display employee details with calculated salary and tax, displayEmployeeInfo() for formatted output

f. Create static methods to track total employees and generate company-wide payroll reports

g. In main, create different types of employee objects, demonstrate method overloading by calling the same method names with different parameters

SOLUTION:

class Employee {

private String empId, empName, department, empType;

private double baseSalary;

private static int totalEmployees = 0;

public Employee(String empId, String empName, String department, double baseSalary) {

this.empId = empId;

this.empName = empName;

this.department = department;

this.baseSalary = baseSalary;

this.empType = "Full-Time";

totalEmployees++;

}

public Employee(String empId, String empName, String department, double hourlyRate, int hoursWorked) {

this.empId = empId;

this.empName = empName;

this.department = department;

this.baseSalary = hourlyRate \* hoursWorked;

this.empType = "Part-Time";

totalEmployees++;

}

public Employee(String empId, String empName, String department, double contractAmount, boolean isContract) {

this.empId = empId;

this.empName = empName;

this.department = department;

this.baseSalary = contractAmount;

this.empType = "Contract";

totalEmployees++;

}

public double calculateSalary(double bonus) {

return baseSalary + bonus; // Full-time

}

public double calculateSalary(int hoursWorked, double hourlyRate) {

return hourlyRate \* hoursWorked; // Part-time

}

public double calculateSalary() {

return baseSalary; // Contract

}

public double calculateTax(double salary, double taxRate) {

return salary \* taxRate / 100;

}

public void generatePaySlip() {

double salary;

double tax;

if (empType.equals("Full-Time")) {

salary = calculateSalary(5000);

tax = calculateTax(salary, 10);

} else if (empType.equals("Part-Time")) {

salary = baseSalary;

tax = calculateTax(salary, 5);

} else {

salary = calculateSalary();

tax = calculateTax(salary, 8);

}

System.out.println("\n--- Pay Slip ---");

System.out.println("ID: " + empId + " | Name: " + empName + " | Dept: " + department);

System.out.println("Type: " + empType);

System.out.println("Salary: Rs." + salary);

System.out.println("Tax: Rs." + tax);

System.out.println("Net Pay: Rs." + (salary - tax));

}

public void displayEmployeeInfo() {

System.out.println(empId + " | " + empName + " | " + department + " | " + empType + " | Rs." + baseSalary);

}

public static int getTotalEmployees() {

return totalEmployees;

}

}

public class EmployeePayroll {

public static void main(String[] args) {

Employee e1 = new Employee("E001", "Aayush Rai", "IT", 40000);

Employee e2 = new Employee("E002", "Rohan Sharma", "HR", 500, 40);

Employee e3 = new Employee("E003", "Priya Singh", "Finance", 60000, true);

System.out.println("--- Employee Details ---");

e1.displayEmployeeInfo();

e2.displayEmployeeInfo();

e3.displayEmployeeInfo();

e1.generatePaySlip();

e2.generatePaySlip();

e3.generatePaySlip();

System.out.println("\nTotal Employees: " + Employee.getTotalEmployees());

}

}

OUTPUT:

--- Employee Details ---

E001 | Aayush Rai | IT | Full-Time | Rs.40000.0

E002 | Rohan Sharma | HR | Part-Time | Rs.20000.0

E003 | Priya Singh | Finance | Contract | Rs.60000.0

--- Pay Slip ---

ID: E001 | Name: Aayush Rai | Dept: IT

Type: Full-Time

Salary: Rs.45000.0

Tax: Rs.4500.0

Net Pay: Rs.40500.0

--- Pay Slip ---

ID: E002 | Name: Rohan Sharma | Dept: HR

Type: Part-Time

Salary: Rs.20000.0

Tax: Rs.1000.0

Net Pay: Rs.19000.0

--- Pay Slip ---

ID: E003 | Name: Priya Singh | Dept: Finance

Type: Contract

Salary: Rs.60000.0

Tax: Rs.4800.0

Net Pay: Rs.55200.0

Total Employees: 3

**4. Write a program to create a Vehicle rental system demonstrating static and instance members**

**Hint** =>

a. Create a Vehicle class with private instance variables: vehicleId (String), brand (String), model (String), rentPerDay (double), isAvailable (boolean)

4



b. Create static variables: totalVehicles (int), totalRevenue (double), companyName (String), rentalDays (int to track total rental days)

c. Create a constructor and instance methods: rentVehicle(int days) which calculates rent and updates availability, returnVehicle() to make vehicle available again

d. Create static methods: setCompanyName(String name), getTotalRevenue(), getAverageRentPerDay(), displayCompanyStats()

e. Create a method calculateRent(int days) that returns rental amount and updates static revenue counter

f. Create a displayVehicleInfo() method showing all vehicle details including rental history

g. In main, demonstrate the difference between static and instance members by creating multiple vehicle objects, show how static variables are shared across all objects while instance variables are unique to each object

SOLUTION:

class Vehicle {

private String vehicleId, brand, model;

private double rentPerDay;

private boolean isAvailable;

private static int totalVehicles = 0;

private static double totalRevenue = 0;

private static int rentalDays = 0;

private static String companyName = "Not Set";

public Vehicle(String vehicleId, String brand, String model, double rentPerDay) {

this.vehicleId = vehicleId;

this.brand = brand;

this.model = model;

this.rentPerDay = rentPerDay;

this.isAvailable = true;

totalVehicles++;

}

public double calculateRent(int days) {

double amount = rentPerDay \* days;

totalRevenue += amount;

rentalDays += days;

return amount;

}

public void rentVehicle(int days) {

if (isAvailable) {

double amount = calculateRent(days);

isAvailable = false;

System.out.println(brand + " " + model + " rented for " + days + " days. Rent: Rs." + amount);

} else {

System.out.println(brand + " " + model + " is already rented.");

}

}

public void returnVehicle() {

if (!isAvailable) {

isAvailable = true;

System.out.println(brand + " " + model + " returned and is now available.");

} else {

System.out.println(brand + " " + model + " was not rented.");

}

}

public void displayVehicleInfo() {

System.out.println(vehicleId + " | " + brand + " " + model + " | Rs." + rentPerDay + "/day | " +

(isAvailable ? "Available" : "Rented"));

}

public static void setCompanyName(String name) {

companyName = name;

}

public static double getTotalRevenue() {

return totalRevenue;

}

public static double getAverageRentPerDay() {

return rentalDays == 0 ? 0 : totalRevenue / rentalDays;

}

public static void displayCompanyStats() {

System.out.println("\n--- " + companyName + " Rental Stats ---");

System.out.println("Total Vehicles: " + totalVehicles);

System.out.println("Total Revenue: Rs." + totalRevenue);

System.out.println("Total Rental Days: " + rentalDays);

System.out.println("Average Rent/Day: Rs." + getAverageRentPerDay());

}

}

public class VehicleRentalSystem {

public static void main(String[] args) {

Vehicle.setCompanyName("ZoomRide Rentals");

Vehicle v1 = new Vehicle("V001", "Toyota", "Innova", 1500);

Vehicle v2 = new Vehicle("V002", "Honda", "City", 1200);

Vehicle v3 = new Vehicle("V003", "Hyundai", "Creta", 1000);

v1.rentVehicle(3);

v2.rentVehicle(2);

v1.returnVehicle();

v3.rentVehicle(4);

System.out.println("\n--- Vehicle Info ---");

v1.displayVehicleInfo();

v2.displayVehicleInfo();

v3.displayVehicleInfo();

Vehicle.displayCompanyStats();

}

}

OUTPUT:

Toyota Innova rented for 3 days. Rent: Rs.4500.0

Honda City rented for 2 days. Rent: Rs.2400.0

Toyota Innova returned and is now available.

Hyundai Creta rented for 4 days. Rent: Rs.4000.0

--- Vehicle Info ---

V001 | Toyota Innova | Rs.1500.0/day | Available

V002 | Honda City | Rs.1200.0/day | Rented

V003 | Hyundai Creta | Rs.1000.0/day | Rented

--- ZoomRide Rentals Rental Stats ---

Total Vehicles: 3

Total Revenue: Rs.10900.0

Total Rental Days: 9

Average Rent/Day: Rs.1211.11111111111

**5. Write a program to create a Product Inventory system with supplier management using arrays**

**Hint** =>

a. Create a Product class with private variables: productId (String), productName (String), price (double), quantity (int), supplierName (String), category (String)

b. Create static variables: totalProducts (int), totalInventoryValue (double), lowStockCount (int), and a static array to track categories

c. Create instance methods: addStock(int quantity), reduceStock(int quantity) with validation, isLowStock() that returns true if quantity < 10

d. Create methods: calculateProductValue() (price × quantity), updatePrice(double newPrice), displayProductInfo()

5



e. Create static methods: calculateTotalInventoryValue(Product[] products), findLowStockProducts(Product[] products), generateInventoryReport(Product[] products)

f. Create a method searchProduct(Product[] products, String productName) that returns the product if found

g. In main, create an array of Product objects, demonstrate inventory operations, stock management, and generate reports showing the use of both static and instance methods

**6. Write a program to create a School Management system with multiple interacting classes**

**Hint** =>

a. Create a Student class with private variables: studentId (String), studentName (String), grade (int), marks (double array for 5 subjects), totalMarks (double), percentage (double)

b. Create a Teacher class with private variables: teacherId (String), teacherName (String), subject (String), studentsHandled (int), and static variable totalTeachers (int)

c. Create a Subject class with private variables: subjectCode (String), subjectName (String), maxMarks (int), passMarks (int)

d. In Student class, create methods: calculateTotal(), calculatePercentage(), displayResult(), isPass() that checks if student passed in all subjects

e. In Teacher class, create methods: assignGrades(Student student, Subject subject, double marks), displayTeacherInfo(), and static methods to track teacher statistics

f. Create static methods in Student class: getTopStudent(Student[] students), getClassAverage(Student[] students), getPassPercentage(Student[] students)

g. In main, create arrays of Student, Teacher, and Subject objects, demonstrate how teachers assign grades to students, generate class reports, and show school-wide statistics using static methods

6

7