

�� **PRACTICE PROBLEM 1: (Any 4)**

**Understanding Classes and Objects - Real World Analogy**

**Task**: Create a program that demonstrates the concept of classes and objects using a real-world analogy.

public class Car {

// TODO: Define instance variables (attributes):

// - brand (String)

// - model (String)

// - year (int)

// - color (String)

// - isRunning (boolean)

// TODO: Create a constructor that initializes all attributes

// TODO: Create instance methods:

// - startEngine() - sets isRunning to true, prints message // - stopEngine() - sets isRunning to false, prints message // - displayInfo() - prints all car information

// - getAge() - returns current year minus car year

public static void main(String[] args) {

// TODO: Create 3 different Car objects with different attributes // TODO: Demonstrate calling methods on each object

// TODO: Show how each object maintains its own state

// TODO: Explain in comments: How is this similar to real-world cars?

}

}



�� **PRACTICE PROBLEM 2:**

**Class Definition and Object Creation**

**Task**: Create a Student class that demonstrates proper class structure and object instantiation.

public class Student {

// TODO: Define private instance variables:

// - studentId (String)

// - name (String)

// - grade (double)

// - course (String)

// TODO: Create a default constructor (no parameters)

// TODO: Create a parameterized constructor that accepts all attributes // TODO: Create getter and setter methods for all attributes

// TODO: Create a method calculateLetterGrade() that returns: // A (90-100), B (80-89), C (70-79), D (60-69), F (below 60)

// TODO: Create a method displayStudent() that shows all information

public static void main(String[] args) {

// TODO: Create one student using default constructor, then set values

// TODO: Create another student using parameterized constructor // TODO: Demonstrate all getter/setter methods

// TODO: Show both students' information and letter grades }

}



�� **PRACTICE PROBLEM 3:**

**Instance vs Static (Class) Members**

**Task**: Create a program that clearly demonstrates the difference between instance and static members.

public class BankAccount {

// TODO: Create static variables:

// - bankName (String) - same for all accounts

// - totalAccounts (int) - count of all accounts created // - interestRate (double) - same rate for all accounts

// TODO: Create instance variables:

// - accountNumber (String) - unique for each account

// - accountHolder (String) - unique for each account

// - balance (double) - unique for each account

// TODO: Create constructor that:

// - Initializes instance variables

// - Increments totalAccounts counter

// TODO: Create static methods:

// - setBankName(String name)

// - setInterestRate(double rate)

// - getTotalAccounts() - returns count

// - displayBankInfo() - shows bank name and total accounts

// TODO: Create instance methods:

// - deposit(double amount)

// - withdraw(double amount)

// - calculateInterest() - uses static interestRate

// - displayAccountInfo()

public static void main(String[] args) {

// TODO: Set bank name and interest rate using static methods // TODO: Create multiple BankAccount objects

// TODO: Show that static members are shared across all objects // TODO: Show that instance members are unique to each object



// TODO: Demonstrate calling static methods with and without objects

}

}

�� **PRACTICE PROBLEM 4:**

**OOP Benefits - Reusability and Extensibility Task**: Create a base class and demonstrate how OOP promotes code reusability.

public class Vehicle {

// TODO: Create protected instance variables:

// - make (String)

// - model (String)

// - year (int)

// - fuelLevel (double)

// TODO: Create constructor

// TODO: Create common methods:

// - startVehicle()

// - stopVehicle()

// - refuel(double amount)

// - displayVehicleInfo()

public static void main(String[] args) {

// TODO: Create different types of vehicles (Car, Truck, Motorcycle)

// TODO: Show how the same Vehicle class can be reused // TODO: Create an array of Vehicle objects

// TODO: Demonstrate polymorphic behavior

// TODO: In comments, explain:

// - How does this show reusability?

// - How could this be extended for new vehicle types?



// - What are the benefits over writing separate classes?

}

}

�� **PRACTICE PROBLEM 5:**

**Multiple Classes and Object Interaction**

**Task**: Create a program with multiple interacting classes to demonstrate object-oriented design.

// TODO: Create a Book class with:

// - title, author, isbn, price, quantity

// - constructor and methods for book operations

// TODO: Create a Library class with:

// - libraryName, Book array, totalBooks count

// - methods to add books, search books, display inventory

public class LibrarySystem {

// TODO: Create multiple Book objects

// TODO: Create a Library object

// TODO: Demonstrate:

// - Adding books to library

// - Searching for books by title or author

// - Displaying library inventory

// - Calculating total value of all books

// TODO: Show object interaction:

// - Library contains Book objects

// - Methods that work with multiple objects

public static void main(String[] args) {

// TODO: Implement a simple menu system:

// 1. Add Book

// 2. Search Book



// 3. Display All Books

// 4. Calculate Total Value

// 5. Exit

// Your implementation here

}

}

�� **PRACTICE PROBLEM 6:**

**Complete OOP Application (15 minutes) Task**: Create a comprehensive application that combines all OOP concepts learned. import java.util.Scanner;

public class EmployeeManagementSystem {

// TODO: Create an Employee class with:

// - Static variable: companyName, totalEmployees

// - Instance variables: empId, name, department, salary // - Constructors (default and parameterized)

// - Static methods: setCompanyName(), getTotalEmployees() // - Instance methods: calculateAnnualSalary(), displayEmployee(), updateSalary()

// TODO: Create a Department class with:

// - deptName, Employee array, employeeCount

// - methods to add employees, find highest paid, calculate total payroll

// TODO: Main application features:

// 1. Set company name

// 2. Create multiple departments

// 3. Add employees to departments



// 4. Display department-wise employee information // 5. Find highest paid employee across all departments // 6. Calculate company-wide payroll

// 7. Show total number of employees

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("=== EMPLOYEE MANAGEMENT SYSTEM ===");

// TODO: Implement interactive menu system:

// 1. Add Employee

// 2. Display All Employees

// 3. Search Employee

// 4. Department Statistics

// 5. Company Statistics

// 6. Exit

// TODO: Demonstrate all OOP principles:

// - Encapsulation (private variables, public methods) // - Object creation and interaction

// - Static vs instance members

// - Code reusability

// Your implementation here

scanner.close();

}

}