C-DAC Mumbai
OOPJ Lab Assignment

Problem 1: Counter for Cups
Scenario: You are keeping track of how many cups of tea are prepared in your home.
Requirements:

1. Create a class TeaCup with instance variable: teaType (String).
2. Create a static variable totalCups to count all cups created.
3. Constructor should initialize teaType and increment totalCups.
4. Create getter for teaType.
5. Create a static method showTotalCups() to print total cups.

Input Example:
Cup1: teaType = "Masala Tea"
Cup2: teaType = "Green Tea"
Cup3: teaType = "Ginger Tea"

Expected Output:
Cup1 type: Masala Tea
Cup2 type: Green Tea
Cup3 type: Ginger Tea
Total cups made: 3

Problem 2: Simple Mobile Tracker
Scenario: A shop wants to count how many mobiles are added to their inventory.
Requirements:

1. Create a class Mobile with instance variable: model (String).
2. Create a static variable totalMobiles to count total mobiles added.
3. Constructor should initialize model and increment totalMobiles.
4. Create a getter for model.
5. Create a static method showTotalMobiles() to print total mobiles.

Input Example:
Mobile1: model = "Samsung Galaxy M32"
Mobile2: model = "Redmi Note 12"

Expected Output:
Mobile1 model: Samsung Galaxy M32
Mobile2 model: Redmi Note 12
Total mobiles in stock: 2

Problem 3: Library Book Tracker
Scenario: A library in Delhi wants to track how many books are issued in total and details of each book.
Requirements:

1. Create a Book class with instance variables: title (String), author (String), issued (boolean).
2. Create static variable totalIssuedBooks to keep track of the total number of books issued.
3. Create a constructor to initialize the book details.
4. Create getters and setters for all instance variables.
5. Create a static method showTotalIssued() to print total issued books.
6. Write a main class to create **3 books**, issue some of them (issued = true), and show total issued books.

Input Example:
Book1: "Harry Potter", Author: "J.K. Rowling", Issued: true
Book2: "Five Point Someone", Author: "Chetan Bhagat", Issued: false
Book3: "Rich Dad Poor Dad", Author: "Robert Kiyosaki", Issued: true

Expected Output:
Book1 issued? true
Book2 issued? false
Book3 issued? true
Total books issued: 2

Problem 4: Employee Salary Manager
Scenario: A company in Bengaluru wants to maintain employee details and give a bonus to employees who have worked more than 5 years.
Requirements:

1. Create a class Employee with instance variables: name (String), salary (double), yearsOfService (int).
2. Create static variable totalEmployees to store the number of employees created.
3. Constructor should initialize all instance variables and increment totalEmployees.
4. Create getters and setters for all instance variables.
5. Create a method calculateBonus() that returns 5% of salary if yearsOfService > 5, otherwise 0.
6. Create a static method showTotalEmployees() to print total employees created.
7. Write a main class to create **3 employees**, print their bonuses, and print total employees.

Input Example:
Employee1: Name: "Ravi", Salary: 150000, Years of Service: 6
Employee2: Name: "Anita", Salary: 120000, Years of Service: 3
Employee3: Name: "Suresh", Salary: 100000, Years of Service: 5

Expected Output:
Employee Ravi Bonus: 7500.0
Employee Anita Bonus: 0.0
Employee Suresh Bonus: 0.0
Total employees: 3

Problem 5: Student Marks Calculator
Scenario: A school in Mumbai wants to calculate marks of students and also maintain total students in the class.
Requirements:

1. Create a class Student with instance variables: name (String), marks (int).
2. Create static variable totalStudents to count total number of students.
3. Constructor to initialize student details and increment totalStudents.
4. Getter and Setter for marks.
5. Method isPassed() returns true if marks >= 35, false otherwise.
6. Static method showTotalStudents() prints total students.
7. In main class, create **3 students**, check if they passed, and show total students.

Input Example:
Student1: Name: "Rahul", Marks: 78
Student2: Name: "Pooja", Marks: 34
Student3: Name: "Amit", Marks: 65

Expected Output:
Student Rahul Passed? true
Student Pooja Passed? false
Student Amit Passed? true
Total students: 3

Problem 6: Indian Railway Ticket Booking
Scenario:
You are building a mini ticket booking system. A passenger can book a ticket either by giving **name and age** or **name, age, and seat type**. The system should also count the **total tickets booked** using a static counter.
Tasks:

1. Create a Passenger class.
2. Implement **two constructors** (constructor overloading):
   Constructor 1 → Passenger(String name, int age)
   Constructor 2 → Passenger(String name, int age, String seatType)
3. Use a **static counter** to keep track of **total passengers booked**.
4. Print passenger details and total passengers.

Input Example:
Passenger1: "Ravi", 25
Passenger2: "Anita", 30, "AC Sleeper"
Passenger3: "Suresh", 40

Expected Output:
Passenger1: Name: Ravi, Age: 25, Seat: General
Passenger2: Name: Anita, Age: 30, Seat: AC Sleeper
Passenger3: Name: Suresh, Age: 40, Seat: General
Total Passengers Booked: 3

Problem 7: Indian Movie Ticket Booking
Scenario:
A cinema hall offers **Normal** and **Premium** tickets. A customer can book **just name** or **name with ticket type**. Keep track of **total tickets sold** using a static counter.
Tasks:

1. Create a Customer class.
2. Implement **two constructors**:
   Constructor 1 → Customer(String name)
   Constructor 2 → Customer(String name, String ticketType)
3. **Static counter** to track tickets sold.
4. Print customer details and total tickets sold.

Input Example:
Customer1: "Rahul"
Customer2: "Pooja", "Premium"
Customer3: "Amit"

Expected Output:
Customer1: Name: Rahul, Ticket: Normal
Customer2: Name: Pooja, Ticket: Premium
Customer3: Name: Amit, Ticket: Normal
Total Tickets Sold: 3

Problem 8: Bank Account Initialization
Scenario:
A bank wants to **initialize the interest rate** for all accounts **once** when the system starts. Each account has **account holder name**, **balance**, and **interest rate**. Students should practice **static blocks** for initialization and **setters/getters** to modify and access account details.
Tasks:

1. Create a BankAccount class.
2. Use a **static block** to initialize **interest rate** to 4%.
3. Create instance variables: name (String) and balance (double).
4. Create **setters and getters** for name and balance.
5. Print account details including interest rate.

Input Example:
Account1: Name="Rohit", Balance=5000
Account2: Name="Priya", Balance=15000

Expected Output:
Bank Interest Rate Initialized: 4.0%
Account1: Name=Rohit, Balance=5000.0, Interest Rate=4.0%
Account2: Name=Priya, Balance=15000.0, Interest Rate=4.0%

Problem 9: School Fee System
Scenario:
A school wants to **initialize the tuition fee** for all students once at program start. Each student has **name** and **class**. Use **static blocks** to set the fee and **setters/getters** to update/access student information.
Tasks:

1. Create a Student class.
2. Use a **static block** to initialize **tuitionFee** to 30000.
3. Create instance variables: name (String) and className (String).
4. Create **setters and getters** for name and className.
5. Print student details including tuition fee.

Input Example:
Student1: Name="Anjali", Class="10th"
Student2: Name="Vikram", Class="12th"

Expected Output:
School Tuition Fee Initialized: 30000
Student1: Name=Anjali, Class=10th, Tuition Fee=30000
Student2: Name=Vikram, Class=12th, Tuition Fee=30000

Problem 10: Student Marks Checker
Scenario:
Create a Student class with rollNo, name, and marks.

- Use a **parameterized constructor** to initialize all fields.
- Create a **getter and setter** for marks.
- In main, create **one student**, update marks using setter, and print student details.

Problem 11: Student Grade Calculator
Scenario:
Extend previous problem. Add method calculateGrade() which returns:

- "A" if marks ≥ 80
- "B" if marks ≥ 60
- "C" if marks ≥ 40
- "Fail" otherwise
- Create **2 students**, print marks and grades.

Problem 12: Bank Account Basic Info
Scenario:
Create BankAccount class with accountHolder and balance.

- Use **parameterized constructor** to initialize account.
- Create **getter and setter** for balance.
- In main, create **one account** and display details.

Problem 13: Bank Deposit & Withdrawal
Scenario:
Extend previous problem. Add methods:

- deposit(double amount) → adds to balance
- withdraw(double amount) → subtracts from balance
- Create **two accounts**, perform deposit/withdraw, display updated balance.

Problem 14: Bank Name Display
Scenario:
Add a **static variable** bankName = "CDAC Bank" and **static method** displayBankName() to BankAccount.

- Call displayBankName() from main.
- Create **one account** to verify instance creation.

Problem 15: Employee Auto-ID Generator
Scenario:
Create Employee class with id, name, basicSalary.

- Add **static counter** starting from 1001 for IDs.
- Default constructor → name = "Unknown", salary = 20000
- Parameterized constructor → accept name and salary
- Getter for all variables
- Create **2 employees** and display their IDs, names, salary.

Problem 16: Employee Net Salary
Scenario:
Extend previous problem. Add method calculateNetSalary():

- Add 10% HRA, 5% DA, deduct 2% PF from basicSalary
- Print net salary for **2 employees**

Problem 17: Library Book Addition
Scenario:
Create Book class with bookId, title, author.

- Constructor + Getters/Setters
- Create Library class with libraryName and **static totalBooks**
- Method addBook(Book b) → increments totalBooks
- Method displayTotalBooks() → prints totalBooks
- Add **2 books** to library and display total books

Problem 18: Vehicle Registration – Static Counter
Scenario:
Create Vehicle class with regNo, ownerName, vehicleType.

- Static variable: vehicleCount
- Constructor → auto-generate regNo as "MH-2025-" + vehicleCount
- Getter methods for all fields

- Create **2 vehicles**, display registration details

Problem 19: Vehicle Registration – Static Block
Scenario:
Add a **static block** to Vehicle class:

- Print "Welcome to CDAC Vehicle Registration Portal" when class loads
- Verify that the message prints **only once** when multiple vehicles are created

---

Problem 20: Ticket Booking System
Question:
Create a class Ticket with:

- passengerName (instance)
- ticketNo (instance, auto-generated using a static counter starting from 5001)
- Constructor to accept passengerName
- Method displayTicket() to show ticket details

Task:
Create 3 tickets and display their details.
Sample Input:
Passenger 1: Rahul
Passenger 2: Priya
Passenger 3: Amit

Sample Output:
Ticket No: 5001, Passenger: Rahul
Ticket No: 5002, Passenger: Priya
Ticket No: 5003, Passenger: Amit