OOP Java Problem Statements (No Collections Allowed)

Answer Any 5

1. Custom Dynamic Array (Without Using ArrayList)

> Design a class `MyArray` that mimics a dynamic array (like `ArrayList`) using just primitive arrays internally.

Requirements:

- `add(int element)`
- `get(int index)`
- `size()`
- `resize()` internally when array gets full

---

2. Inventory Management System

> Design classes for an e-commerce system that manages `Product`, `Warehouse`, and `Order`.

Rules:

- A product has ID, name, price, and quantity.
- A warehouse holds a fixed-size array of products.
- An order requests products — check stock before confirming.

No Collection types; use fixed-size arrays or custom structures.

---

3. Banking System with OOP

Build a simple banking system with classes like `Bank`, `Account`, and `Customer`.

Features:

- Create new account
- Deposit/withdraw money
- Print account statement

Challenge: Use inheritance for different types of accounts (e.g., `SavingsAccount`, `CurrentAccount`) with different withdrawal rules.

---

4. Car Simulation

Create a `Car` class with properties like speed, fuel, and distance traveled.

Add:

- A method `drive(int hours)` that updates distance and reduces fuel
- A subclass `ElectricCar` that handles fuel as battery percentage

Demonstrate **inheritance and polymorphism**.

---

5. Shape Area Calculator

Create a base class `Shape` and extend it with `Circle`, `Rectangle`, `Triangle`.

Each shape should implement a method `getArea()` and print its type.

Demonstrate:

- Inheritance
- Abstraction
- Method overriding

---

6. Employee Salary Calculation

> Model a company with different employee types (`FullTime`, `PartTime`, `Contractor`).

Each class should:

- Inherit from `Employee`
- Implement its own `calculateSalary()` method

---

7. Matrix Operations (Without Collections)

> Build a `Matrix` class that supports:

- Addition
- Subtraction
- Transposition
- Multiplication

Use only 2D arrays and OOP principles.

---

8. Custom Date Formatter

> Create a class `MyDate` with day, month, year.
> Implement methods to:

- Validate the date
- Increment date by one day
- Format as `dd-mm-yyyy`

---

9. Student Grading System

Create classes for `Student`, `Subject`, and `ReportCard`.

Each:

- Student has a name, roll number, and array of Subjects
- Subject has name and marks
- ReportCard calculates total, average, and grade

No Collection APIs allowed.

---

10. Basic ATM Machine Simulation

Create a console-based simulation of an ATM machine with classes like:

- `ATM`
- `UserAccount`
- `Bank`

Features:

- PIN validation
- Balance checking
- Cash withdrawal

No data structures beyond arrays.
