Simulate a Parking Lot System

Problem:

Design a parking lot where:

- There are limited parking slots.
- Only one vehicle can occupy a slot.
- Vehicles can be `Car`, `Bike`, or `Truck`.
- Each vehicle has a different parking charge.
- Track total revenue, and show parked vehicle info.

Constraints:

- No collections; use arrays or custom logic.
- Use `Vehicle` base class and inheritance.

OOP Concepts:
Encapsulation, Inheritance, Polymorphism, Abstraction

---

2. Implement a Bank System with Fixed Number of Accounts

Problem:

Simulate a bank with:

- A fixed number of accounts.
- Each `Account` has balance, owner name, account number.
- Perform deposit, withdraw, and transfer between accounts.

Constraints:

- Max 10 accounts (use array).
- Prevent overdraft.
- No `ArrayList` — only use arrays.

Bonus: Generate unique account numbers automatically.

OOP Concepts:
Encapsulation, Abstraction, Constructor overloading

---

3. Create Your Own Mini ATM

Problem:

Design an `ATM` machine that:

- Accepts a PIN to log in.
- Lets users check balance, withdraw, or deposit.
- There are max 5 users, each with a predefined PIN and balance.

Constraints:

- Use only arrays.
- Secure access via PIN validation.
- Track failed attempts.

OOP Concepts:
Encapsulation, Abstraction, Classes and Objects

---

4. Employee Promotion System (Without Collections)

Problem:

Design a system that:

- Has an `Employee` class with id, name, designation, salary.
- Allows promoting an employee to the next level:
    - `Junior` → `Mid` → `Senior` → `Lead`
- Salary increases with level.

Constraints:

- Use enums for levels.
- Store max 5 employees.
- No `List` or `Set` — use array of objects.

OOP Concepts:
Enums, Encapsulation, Inheritance, Polymorphism

---

5. Simulate a Library System

Problem:

Design a library with:

- Max 5 books.
- `Book` class with title, author, and issued status.
- Ability to issue a book, return it, and list available books.

Constraints:

- Don’t use any collections.
- Each book is uniquely identified.

OOP Concepts:
Classes, Methods, State Management, Constructors