**📱 Telecom Domain Business Use Case**

**Title:** *Telecom Customer Billing & Plan Management System*

**Scenario:**  
A telecom company provides multiple **plans** (Prepaid, Postpaid, Data-only). Customers can subscribe to these plans and are billed based on their usage. The system must handle customer management, plan management, and billing calculations.

**1. OOP Concepts**

* **Encapsulation**:  
  Customer details (name, phone number, plan, usage data) are encapsulated inside a Customer class with private fields and public getters/setters.
* **Inheritance**:  
  Plan is an abstract class. Different plans like PrepaidPlan, PostpaidPlan, and DataPlan inherit from it. Each has different billing logic.
* **Polymorphism**:  
  Billing calculation is polymorphic — calling calculateBill() on a Plan object will execute different implementations depending on the plan type.
* **Abstraction**:  
  The Plan class defines an abstract method calculateBill(). Subclasses implement their own billing rules.

**2. Exception Handling**

* **InvalidPlanException**: Thrown if a customer tries to subscribe to a non-existent plan.
* **InsufficientBalanceException**: Thrown for prepaid customers if they try to use services without enough balance.
* **UsageLimitExceededException**: Thrown if a customer exceeds their plan’s fair usage limit.

**3. Collections Framework**

* **List<Customer>**: Stores all customers.
* **Map<String, Plan>**: Stores available plans with planCode → Plan mapping.
* **Set<String>**: Keeps track of unique phone numbers (no duplicates).

**Example Workflow**

1. Admin loads available plans into a Map.
2. Customer subscribes to a plan.
3. Usage is recorded (calls, SMS, data).
4. System calculates bill using polymorphic calculateBill().
5. Exceptions are raised if usage or balance rules are violated.
6. Bills are stored in a List for reporting.