**🐍 DAY 18 – HOME ASSIGNMENTS: Encapsulation, Abstraction, and Interfaces**

(*Private Attributes, Getters & Setters, Abstract Classes, Interfaces (using abstract methods), and Polymorphic Designs.*)

**🔐 Part A: Encapsulation Basics**

1. **BankAccount Class**:
   * Make balance a **private attribute**.
   * Add deposit() and withdraw() methods.
   * Prevent withdrawals if balance is insufficient.
2. **Student Class**:
   * Make name and age **private**.
   * Use get\_name(), set\_name() and get\_age(), set\_age() methods.
   * Validate age to be a positive integer.
3. **Product Class**:
   * Make price private.
   * Use a method to apply discount (without direct access).

**⚡️ Part B: Abstraction with Abstract Classes**

1. Create an abstract class Shape:
   * Abstract method: area()
   * Abstract method: perimeter()
2. Create concrete subclasses:
   * Rectangle: Implement area() and perimeter().
   * Circle: Implement area() and perimeter().
3. Test by creating instances of Rectangle and Circle and calling their methods.

**🕹️ Part C: Interface Simulation with Abstract Methods**

1. Create an abstract class Vehicle:
   * Abstract methods:
     + start\_engine()
     + stop\_engine()
   * Implement concrete subclasses:
     + Car
     + Motorcycle
   * Test instances for both.
2. Create an abstract class Database:
   * Abstract methods:
     + connect()
     + disconnect()
     + execute\_query(query)
   * Create concrete classes:
     + MySQLDatabase
     + PostgreSQLDatabase
     + SQLiteDatabase

**🎨 Part D: Polymorphism with Abstraction**

1. Create an abstract class Animal:
   * Abstract method: make\_sound()
2. Subclasses:
   * Dog: Returns Woof
   * Cat: Returns Meow
   * Cow: Returns Moo
   * Test polymorphism by looping over instances and calling make\_sound().

**👥 Part E: Encapsulation + Abstraction Combined**

1. Build a Bank System:
   * Abstract class Account:
     + Private balance
     + Abstract method: calculate\_interest()
   * Subclasses:
     + SavingsAccount: 4% interest.
     + CurrentAccount: 1% interest.
   * Methods:
     + get\_balance() – Returns the balance.
     + set\_balance() – Updates balance with checks.
     + calculate\_interest() – Returns calculated interest.
   * Test by creating instances and invoking methods.

**🏢 Part F: Realistic Challenges**

1. **Online Shopping System**:
   * Abstract class: User
     + Abstract method: get\_user\_type()
   * Subclasses:
     + Customer returns Customer for user type.
     + Admin returns Admin for user type.
   * Encapsulate user\_id and user\_email as private.
2. **Employee Hierarchy with Encapsulation**:
   * Abstract class: Employee
     + Abstract method: calculate\_salary()
     + Private attribute: base\_salary
   * Subclasses:
     + FullTimeEmployee (adds benefits and bonuses).
     + PartTimeEmployee (hourly rate and hours worked).

**⚔️ Part G: Bonus Challenges for Day 18**

1. Create an abstract class PaymentMethod:
   * Abstract method: pay(amount)
   * Subclasses:
     + CreditCard
     + PayPal
     + BankTransfer
2. Create an abstract class Notification:
   * Abstract method: send\_message()
   * Subclasses:
     + EmailNotification
     + SMSNotification
     + PushNotification
3. Build a robust **Student Registration System**:
   * Abstract class: Person
     + Abstract method: get\_role()
   * Subclasses:
     + Student (encapsulates marks and validates range).
     + Teacher (encapsulates salary).
     + AdminStaff (encapsulates department).