Introduction to Object-Oriented Programming (OOP) Practice Problems

These problems are designed to help you understand and apply the core principles of Object-Oriented Programming (OOP): Encapsulation, Inheritance, Polymorphism, and Abstraction. Each problem provides a scenario and outlines the key concepts to focus on.

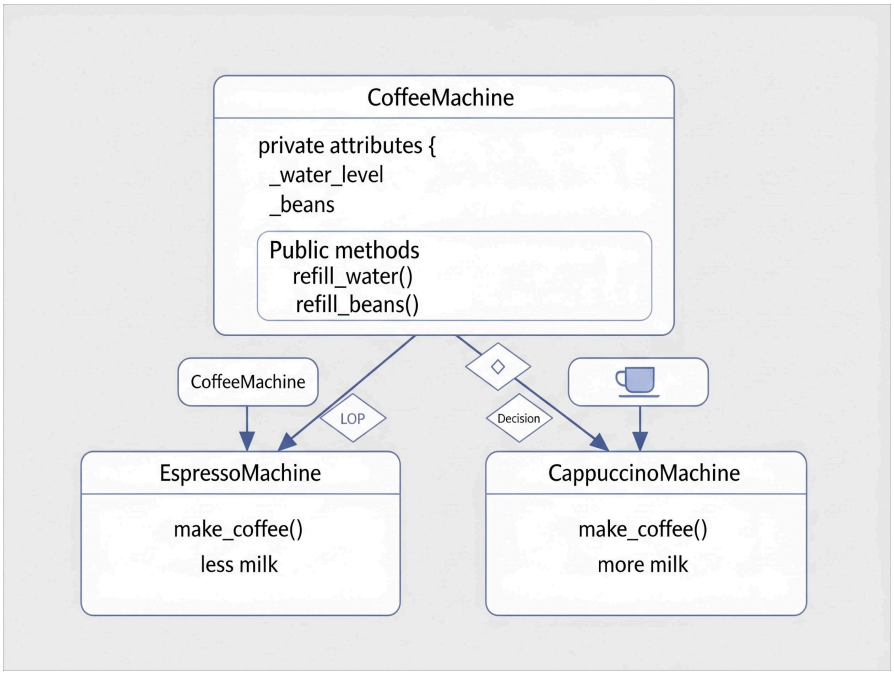
Problems

1. Coffee Machine Simulator ☕

Concepts: Encapsulation, Inheritance, Polymorphism

Problem: Create a base class `CoffeeMachine` with private attributes like `_water_level`, `_beans`. Subclasses `EspressoMachine` and `CappuccinoMachine` should inherit from it. Each subclass overrides a method `make_coffee()` differently (espresso has less milk, cappuccino has more). Use getters/setters to refill ingredients.

Goal: Show how subclasses behave differently but share the same structure.

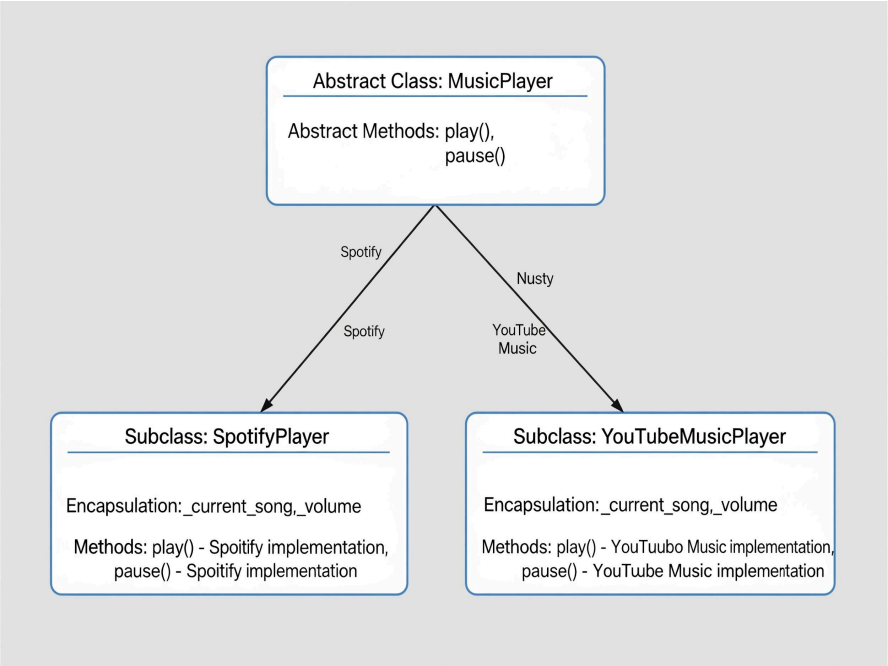


2. Online Music Player 🎵

Concepts: Encapsulation, Inheritance, Polymorphism, Abstraction

Problem: Create an abstract class `MusicPlayer` with abstract methods `play()` and `pause()`. Subclasses `SpotifyPlayer` and `YouTubeMusicPlayer` implement these differently. Use encapsulation to store private attributes like `_current_song`, `_volume`.

Goal: Demonstrate abstract interface + polymorphic play behavior.



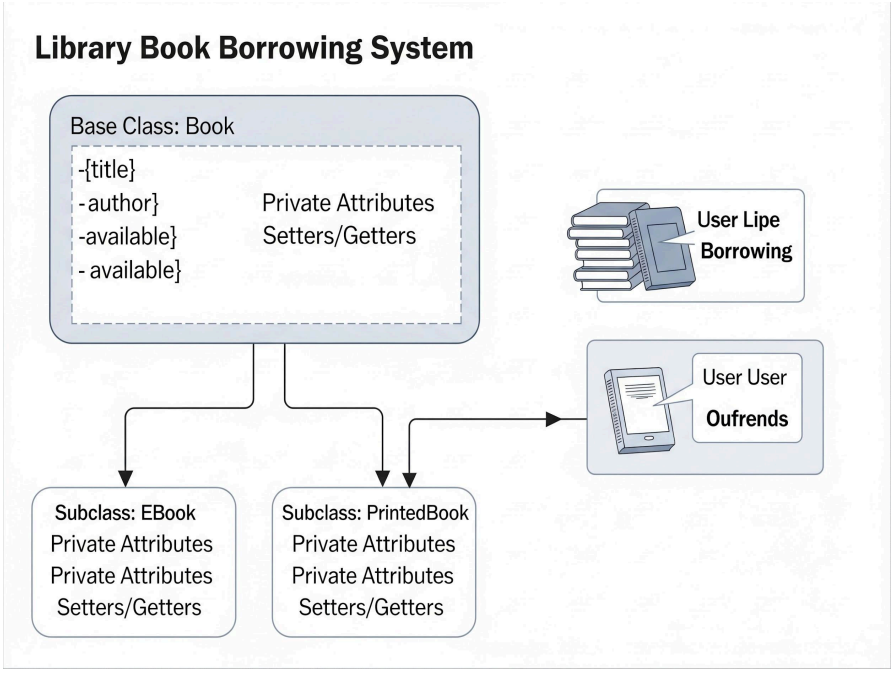
3. Library Book Borrowing System 📚

Concepts: Encapsulation, Inheritance, Polymorphism, Abstraction

Problem:

- Base class: `Book` → attributes: `title`, `author`, `available`.
- Subclasses: `EBook`, `PrintedBook` (different borrow behavior).
- Abstract class: `User` → methods `borrow_book()` and `return_book()`.
- Subclasses: `Student` and `Teacher` (different borrow limits).
- Use private attributes and setters/getters for book info.

Goal: Shows abstraction for user roles, encapsulation for book info, and polymorphism in borrowing.



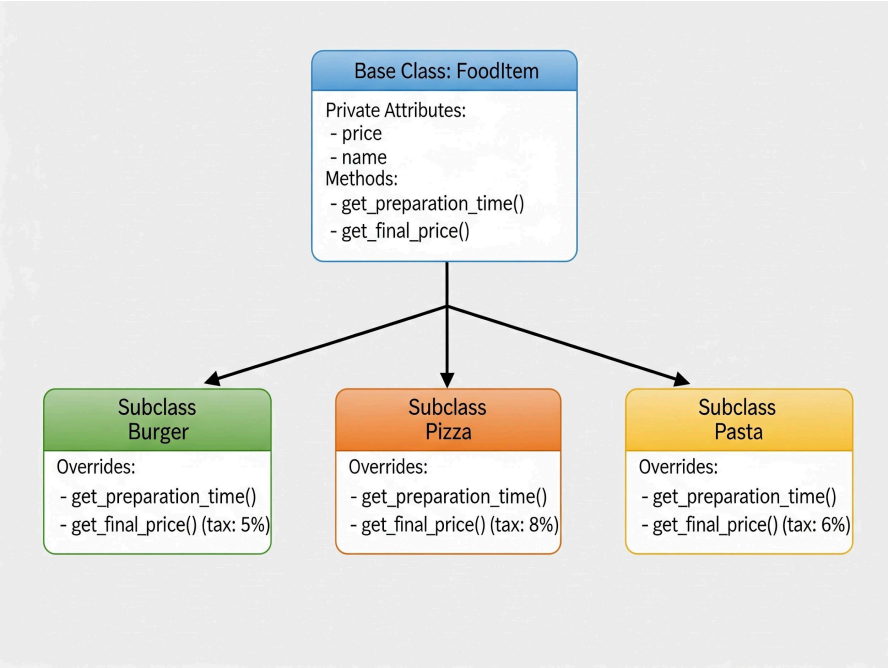
4. Food Ordering System 🍔

Concepts: Encapsulation, Inheritance, Polymorphism

Problem:

- Base class `FoodItem` with private `price` and `name`.
- Subclasses: `Burger`, `Pizza`, `Pasta`.
- Override method `get_preparation_time()`.
- A method `get_final_price()` applies different taxes (override it).

Goal: Demonstrate polymorphism through price/tax behavior.



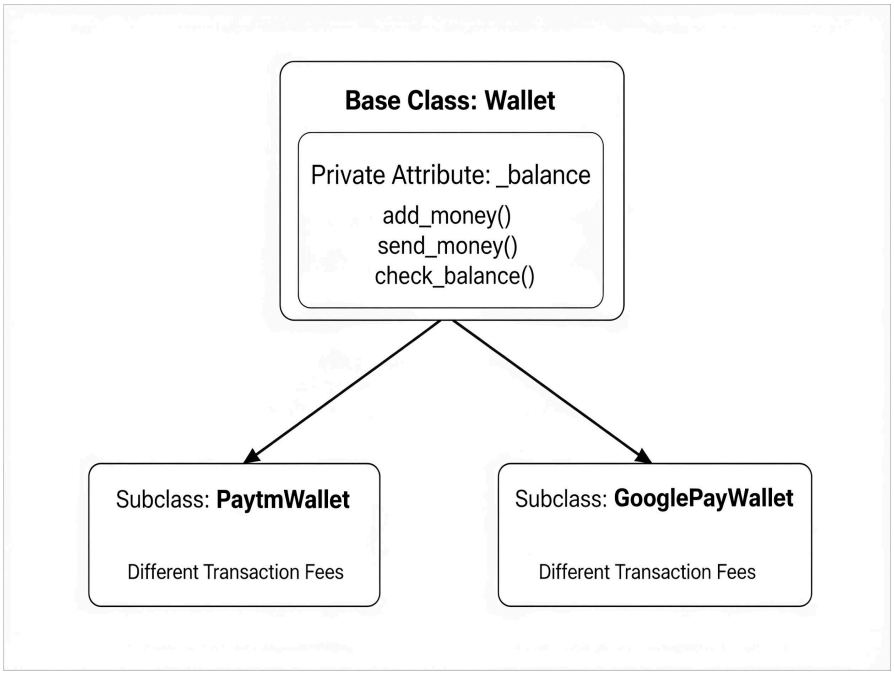
5. Digital Wallet 💳

Concepts: Encapsulation, Inheritance

Problem:

- Base class `Wallet` with private `_balance`.
- Subclasses: `PaytmWallet`, `GooglePayWallet`.
- Common methods: `add_money()`, `send_money()`, `check_balance()`.
- Each subclass has different transaction fees.

Goal: Emphasize encapsulation (balance protection) + inheritance (shared structure).



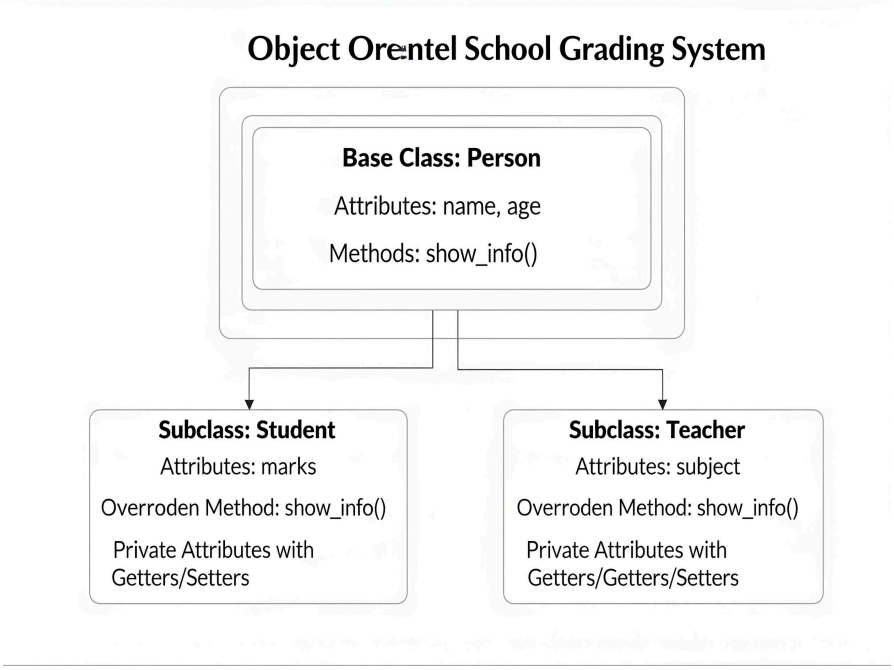
6. School Grading System 🎓

Concepts: Encapsulation, Inheritance, Polymorphism

Problem:

- Base class: `Person` (name, age).
- Subclass: `Student` → marks; `Teacher` → subject.
- Both override a `show_info()` method.
- Private attributes with getters/setters.

Goal: Basic but covers 3 OOP pillars neatly.



7. ATM System 🏧

Concepts: Encapsulation, Inheritance, Polymorphism, Abstraction

Problem:

- Abstract class `Account` with abstract methods `deposit()` and `withdraw()`.
- Subclasses: `SavingsAccount`, `CurrentAccount`.
- Use encapsulation for `_balance`.
- Override `withdraw()` to handle overdraft limits differently.
- Polymorphism: access accounts via same interface.

Goal: Realistic use case combining all four concepts cleanly.

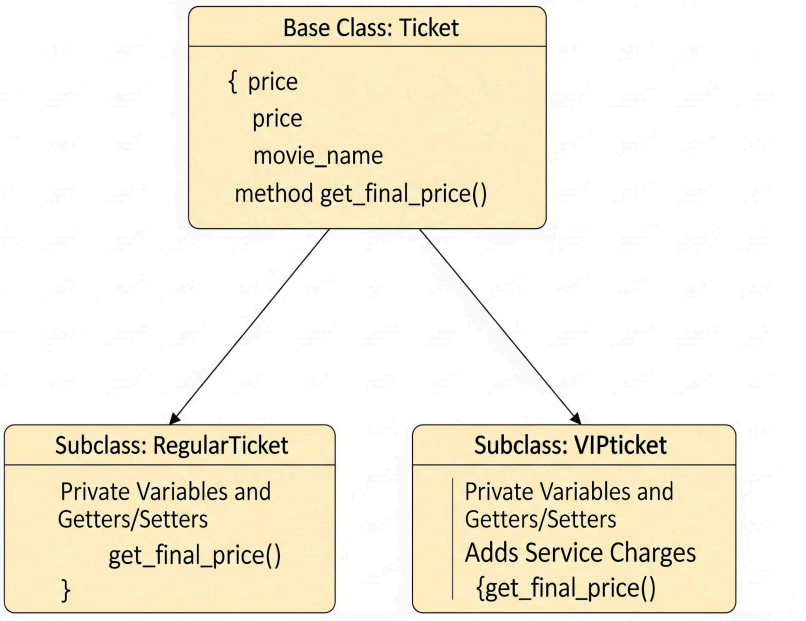
8. Movie Ticket Booking 🎬

Concepts: Encapsulation, Inheritance, Polymorphism

Problem:

- Base class `Ticket` with attributes `price`, `movie_name`.
- Subclasses: `RegularTicket`, `VIPticket`.
- Override method `get_final_price()` (VIP adds service charges).
- Use private variables and getters/setters.

Goal: Small, simple polymorphism example with encapsulation.



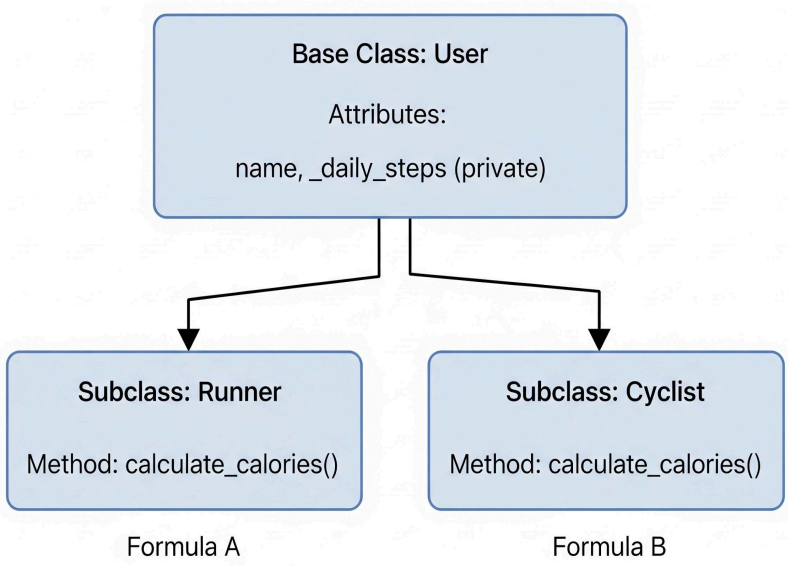
9. Fitness Tracker 🏃

Concepts: Encapsulation, Inheritance

Problem:

- Base class `User` → name, private `_daily_steps`.
- Subclasses: `Runner` and `Cyclist`.
- Both have method `calculate_calories()` but with different formulas.

Goal: Simple real-life example using encapsulation and inheritance.



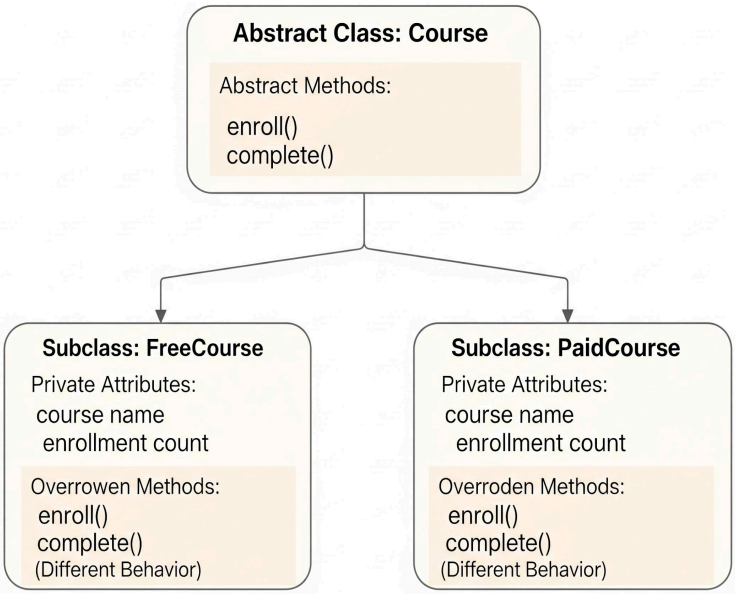
10. Online Course Platform 🎓

Concepts: Encapsulation, Inheritance, Polymorphism, Abstraction

Problem:

- Abstract class `Course` with abstract methods `enroll()` and `complete()`.
- Subclasses: `FreeCourse`, `PaidCourse`.
- Private attributes for course name and enrollment count.
- Override methods to show different behavior.

Goal: Practical educational scenario covering all OOP pillars.


Abstract Class: Course
Abstract Methods:
enroll()
complete()
Subclass: FreeCourse
Private Attributes:
course name
enrollment count
Overrowen Methods:
enroll()
complete()
(Different Behavior)
Subclass: PaidCourse
Private Attributes:
course name
enrollment count
Overroden Methods:
enroll()
complete()
(Different Behavior)