Vehicle Toll Management System

Description:

Simulate a toll booth where different types of vehicles (Bike, Car, Truck) pass through. Each has different toll rates and lengths of stay.

OOP Concepts:

- **Abstraction**: Base class `Vehicle` with methods like `getTollAmount()`.
- **Inheritance**: `Bike`, `Car`, `Truck` classes inherit from `Vehicle`.
- **Polymorphism**: Process different vehicles with the same interface.
- **Encapsulation**: Keep details like plate number and toll amount hidden.

---

2. Banking System with Multiple Account Types

Description:

Create a banking simulation with multiple account types: `SavingsAccount`, `CurrentAccount`, and `FixedDeposit`. Each type behaves differently for interest, withdrawal rules, etc.

OOP Concepts:

- **Inheritance & Polymorphism**: `Account` base class, overridden `withdraw()` and `calculateInterest()`.
- **Encapsulation**: Keep balance and customer details private.
- **Composition**: A `Customer` object can “own” an account.

---

3. Restaurant Menu Simulation

Description:

Simulate a restaurant with different food items (Pizza, Burger, Pasta), each having different price, preparation time, and ingredients.

OOP Concepts:

- **Abstraction**: `FoodItem` interface or abstract class.
- **Inheritance**: `Pizza`, `Burger`, etc., implement/extend `FoodItem`.
- **Polymorphism**: Order any item using a common reference.
- **Encapsulation**: Item cost and cooking time should be private.

---

4. Library Book Management

Description:

Manage a small personal library where books can be issued, returned, and details printed. Include books of type: `Fiction`, `NonFiction`, `Magazine`.

OOP Concepts:

- **Inheritance**: `Book` as a base class, subclasses with custom data.
- **Abstraction**: Methods like `issueBook()`, `returnBook()`.
- **Encapsulation**: Track issue status privately.
- **No collections**: Only 2-3 book objects handled individually.

---

5. Digital Music Player Simulation

Description:

Simulate a basic MP3 player that can play, pause, skip, or repeat tracks. Each track is an object with title, artist, and duration.

OOP Concepts:

- **Encapsulation**: Track state of a song (playing/paused).
- **Abstraction & Polymorphism**: Common interface for audio media.
- **Composition**: A `Player` object contains `Track` objects (individually created, no list).

---

6. Zoo Animal Management System

Description:

Design a system to manage animals in a zoo. Each animal can eat, sleep, and make sounds. Include classes like `Lion`, `Elephant`, `Monkey`.

OOP Concepts:

- **Inheritance & Polymorphism**: `Animal` superclass, override `makeSound()`.
- **Encapsulation**: Private health/status fields.
- **Abstraction**: Common behaviors declared abstract or interface.

---

7. Student Course Enrollment Simulation

Description:

Simulate a student enrolling in 2–3 courses (e.g., Math, Physics, History). Each course can be enrolled in and has unique content and faculty.

OOP Concepts:

- **Composition**: `Student` has `Course` objects.
- **Abstraction**: Courses expose enroll and viewContent methods.
- **Encapsulation**: Grades, attendance stored privately.