Practice Questions

Question 1:
A vehicle rental company provides different types of **vehicles** for rent, such as **cars, bikes, and trucks**. Each vehicle type has a unique way of operating and pricing. The company needs a system where all vehicles can be rented through a common interface, but each type has its own **specific rental process**.
System Requirements:

1. **All vehicles** have a **name, rental cost per day, and max speed**.
2. Each vehicle **must have a rent() method**, but the behavior differs:
   - **Cars** are rented with a driver.
   - **Bikes** are self-driven.
   - **Trucks** require a **special license check** before renting.
3. The company should be able to **process any vehicle** through a common interface.
4. Use **polymorphism** (**method overriding**) to implement this system.

Expected Output:
Car: Toyota Camry is rented with a professional driver.
Bike: Yamaha MT-15 is rented for self-driving.
Truck: Volvo FH16 requires a special license before renting.

Question 2:
A company provides different types of online payment methods for customers, including **Credit Cards, Digital Wallets, and UPI Payments**. The system must handle multiple payment types seamlessly using **polymorphism**, allowing for dynamic method overriding.
Each payment method has a **common processPayment()** method, but the behavior varies:

- **CreditCardPayment** requires verification before processing the payment.
- **DigitalWalletPayment** applies a cashback offer before confirming payment.
- **BankTransferPayment** requires additional bank authentication before processing.

The system must handle **multiple transactions** with **different payment methods**, using polymorphism.

Expected Output:
Verifying credit card details...
Payment of $500 using Credit Card is successful.
Applying 5% cashback for Ahsan...
Payment of $300 via Digital Wallet was successful! Cashback of $25 applied.
Driver Ali is verifying license before renting the truck.

Question 3:

A food delivery application allows users to place orders from different types of restaurants, such as **Fast Food, Fine Dining, and Home Kitchen services**. Each restaurant has a different way of preparing and delivering orders. The system should:

1. Have a **common interface** for processing food orders.
2. Support multiple restaurant types (**FastFood, Gourmet, HomeKitchen**) that override the prepareOrder() method.
3. **FastFood:** Order is instantly prepared and marked as "ready for pickup."
   GourmetRestaurant: The meal requires a longer preparation time and is served at a high standard.
   HomeKitchen: Home chefs prepare meals only on pre-orders and use homemade ingredients.

Expected Output:

```
Preparing order at Kababjees...
Your Cheeseburger is ready for pickup!

Preparing gourmet meal: Salmon Steak at Jameel's Kitchen...
Plating the dish with elegant presentation...
Enjoy your exquisite meal...

Home-cooked meal being prepared for Mrs. Khan...
Using fresh homegrown ingredients...
Your home-cooked meal is ready. Bon appétit!
```

Question 4:

A ride-sharing app connects passengers with different types of drivers: **StandardDriver, LuxuryDriver, and RideSharingDriver**.

- Each **Driver** has a ride() method but provides different ride experiences.
- **StandardDriver**: Offers a normal car ride.
- **LuxuryDriver:** Offers a high-end experience with refreshments.
- **BikeTaxi:** Offers an eco-friendly ride.

Expected Output:

```
Standard Car Ride: Honda Civic
Arriving in 5 minutes...

Luxury Ride: Tesla Model X
Enjoy complimentary water and music...
```

Question 5:
A software company wants to create an employee management system. The company has three levels of hierarchy:

1. **Person –** Stores basic details such as name and age.
2. **Employee –** Extends Person and adds salary.
3. **Manager –** A special type of Employee who also has a teamSize.

Tasks:

- Implement **multilevel inheritance** where Manager extends Employee, and Employee extends Person.
- Implement **method overloading** in the Employee class to calculate the salary with and without a bonus.
- Override the displayInfo() method in Manager to include the team size along with other employee details.
- Automate the unique id assignment to each employee.

Question 6:
A digital library system wants to store information about different types of books. The system has the following book hierarchy:

1. **Book –** Contains the title and author of the book.
2. **EBook –** A digital version of the book that also includes a fileSize attribute.
3. **PDFBook –** A special type of EBook that can be protected with password security.

Tasks:

- Implement **multilevel inheritance** where PDFBook extends EBook, and EBook extends Book.
- Implement **method overloading** in EBook where one version of displayInfo() shows the book details without file size, and another version includes file size.
- Override the displayInfo() method in PDFBook to include protection status.