Java OOP Practice Questions with Instructions

1. Create a class called `Person` with attributes `name` and `age`.
   - Provide a constructor to initialize both attributes.
   - Create two objects of the class and display their details using a method.
2. Create a class called `Dog` with attributes `name` and `breed`.
   - Initialize attributes using a constructor.
   - Provide setter and getter methods for both attributes.
   - Demonstrate updating and retrieving attribute values.
3. Create a class called `Rectangle` with attributes `width` and `height`.
   - Provide methods to calculate area and perimeter.
   - Display the calculated values.
4. Create a class called `Circle` with an attribute `radius`.
   - Provide getter and setter methods for radius.
   - Provide methods to calculate area and circumference.
   - Display the calculated values.
5. Create a class called `Book` with attributes `title`, `author`, and `ISBN`.
   - Implement a collection (e.g., `ArrayList`) to store multiple books.
   - Provide methods to add and remove books from the collection.
6. Create a class called `Employee` with attributes `name`, `jobTitle`, and `salary`.
   - Provide a method to calculate yearly salary.
   - Provide a method to update salary with a raise.
7. Create a class called `Bank` to manage multiple `Account` objects.
   - Define a separate `Account` class with attributes `accountNumber`, `holderName`, and `balance`.
   - Provide methods to deposit and withdraw money.
   - In the `Bank` class, implement methods to add and remove accounts.
8. Create a class called `TrafficLight` with attributes `color` and `duration`.
   - Provide a method to change the light color.
   - Provide methods to check whether the light is red or green.
9. Create a class called `Employee` with attributes `name`, `salary`, and `hireDate`.
   - Implement a method to calculate years of service based on the hire date.
10. Create a class called `Student` with attributes `name`, `grade`, and a list of `courses`.

- Provide methods to add and remove courses.
- Display student details along with enrolled courses.

11. Create a class called `Library`.

- Maintain a collection of books.
- Provide methods to add, remove, and display books.

12. Create a class called `Airplane` with attributes `flightNumber`, `destination`, and `departureTime`.

- Provide methods to check flight status.
- Provide a method to apply delays to departure time.

13. Create a class called `Inventory`.

- Maintain a collection of products.
- Provide methods to add and remove products.
- Provide a method to check for low inventory (below a threshold).

14. Create a class called `School`.

- Maintain collections of students, teachers, and classes.
- Provide methods to add and remove students and teachers.
- Provide a method to create new classes.

15. Create a class called `MusicLibrary`.

- Maintain a collection of songs.
- Provide methods to add, remove, and display songs.
- Provide a method to play a random song.

16. Create an abstract class called `Shape`.

- Define abstract methods `calculateArea()` and `calculatePerimeter()`.
- Create subclasses `Rectangle`, `Circle`, and `Triangle` to implement these methods.

17. Create a class called `Movie` with attributes `title`, `director`, `actors`, and `reviews`.

- Provide a method to add reviews.
- Provide a method to display reviews.

18. Create a class called `Restaurant`.

- Maintain menu items, prices, and ratings.
- Provide methods to add/remove items.
- Provide a method to calculate average rating.

19. Create a travel booking class with the following:

- Methods to search for flights and hotels.
- Methods to book and cancel reservations.

20. Create a class called `BankAccount` with attributes `accountNumber`, `accountHolderName`, and `balance`.

- Provide methods to deposit, withdraw, and check balance.
- Create a subclass `SavingsAccount` with attribute `interestRate` and a method to apply interest.

21. Create a class called `Vehicle` with attributes `make`, `model`, and `year`.

- Create subclasses `Car` (with trunk size) and `Truck` (with payload capacity).
- Provide a method in each subclass to display details.

22. Create a class called `Customer` with attributes `name`, `email`, and `purchaseHistory`.

- Provide methods to add purchases and calculate total expenditure.
- Create a subclass `LoyalCustomer` with a `discountRate` attribute and a method to apply discounts.

23. Create a class called `Course` with attributes `courseName`, `instructor`, and `credits`.

- Create a subclass `OnlineCourse` with attributes `platform` and `duration`.
- Provide methods to display details and check certificate eligibility based on duration.

24. Create a class called `ElectronicsProduct` with attributes `productID`, `name`, and `price`.

- Provide methods to apply discounts and calculate final price.
- Create a subclass `WashingMachine` with a `warrantyPeriod` attribute and a method to extend the warranty.

25. Create a class called `Building` with attributes `address`, `numberOfFloors`, and `totalArea`.

- Create subclasses `ResidentialBuilding` (with number of apartments) and `CommercialBuilding` (with office space).
- Provide methods to calculate total rent for each type.

26. Create a class called `Event` with attributes `eventName`, `date`, and `location`.

- Create subclasses `Seminar` (with number of speakers) and `MusicalPerformance` (with performer list).
- Provide methods to display details and check for scheduling conflicts.

27. Create a class called `CustomerOrder` with attributes `orderID`, `customer`, and `orderDate`.

- Create a subclass `OnlineOrder` with attributes `deliveryAddress` and `trackingNumber`.
- Provide methods to calculate delivery time and update tracking status.

28. Create a class called `Reservation` with attributes `reservationID`, `customerName`, and `date`.

- Create subclasses `ResortReservation` (with room number) and `RailwayReservation` (with seat number).
- Provide methods to check status and modify reservation details.

29. Create a class called `Pet` with attributes `name`, `species`, and `age`.

- Create subclasses `Dog` (with favorite toy) and `Bird` (with wing span).
- Provide methods to display details and calculate pet's age in human years.

30. Create a class called `GymMembership` with attributes `memberName`, `membershipType`, and `duration`.

- Create a subclass `PremiumMembership` with attributes `personalTrainerAvailability` and `spaAccess`.
- Provide methods to calculate membership fees and check for special offers.