OOP in JavaScript Practice Problems

Classes and Objects

1. Create a `Person` class with properties for name and age. Add a method to display the person's details.
2. Write a `Car` class with properties for make, model, and year. Add a method to display the car's details.
3. Implement a `BankAccount` class with deposit and withdraw methods.
4. Create a `Rectangle` class with methods to calculate the area and perimeter.
5. Write a `Circle` class with a method to calculate the area.

Inheritance

6. Extend the `Person` class to create a `Student` class with an additional property for the student ID.
7. Create a `Teacher` class that extends `Person` and includes a property for the subject they teach.
8. Write a `Dog` class that extends an `Animal` class with a method to make the dog bark.
9. Create a `Shape` class and extend it to create `Triangle` and `Square` classes, each with methods to calculate area.
10. Implement an `Employee` class that extends `Person` and adds a property for job title and a method to display the job title.

Encapsulation

11. Implement a `PrivateCounter` class that uses private fields to keep track of a count value.
12. Write a `Temperature` class with private fields for Celsius and Fahrenheit and public methods to convert between them.

13. Create a `SecureBankAccount` class with private fields for balance and methods to deposit and withdraw money.

14. Write a `User` class with private fields for username and password, and methods to validate the password.

15. Implement a `LibraryBook` class with private fields for title and author, and methods to borrow and return the book.

Polymorphism

16. Create a `Vehicle` class with a `move` method, and extend it to create `Car` and `Bicycle` classes that override the `move` method.

17. Write an `Instrument` class with a `play` method, and extend it to create `Guitar` and `Piano` classes that override the `play` method.

18. Implement a `Payment` class with a `process` method, and extend it to create `CreditCardPayment` and `PaypalPayment` classes that override the `process` method.

19. Create a `Notification` class with a `send` method, and extend it to create `EmailNotification` and `SMSNotification` classes that override the `send` method.

20. Write a `Shape` class with a `draw` method, and extend it to create `Circle`, `Square`, and `Triangle` classes that override the `draw` method.

Abstraction

21. Implement an abstract `Vehicle` class with an abstract `start` method, and create concrete subclasses `Car` and `Motorcycle`.

22. Create an abstract `Employee` class with an abstract `calculateSalary` method, and implement it in `FullTimeEmployee` and `PartTimeEmployee` classes.

23. Write an abstract `Animal` class with an abstract `makeSound` method, and create concrete subclasses `Dog` and `Cat`.

24. Implement an abstract `Account` class with an abstract `getAccountDetails` method, and create concrete subclasses `SavingsAccount` and `CheckingAccount`.
25. Create an abstract `Appliance` class with an abstract `turnOn` method, and implement it in `WashingMachine` and `Refrigerator` classes.

Static Methods and Properties
26. Write a `MathUtils` class with static methods to calculate the sum, difference, product, and quotient of two numbers.
27. Create a `Counter` class with a static property to keep track of the number of instances created.
28. Implement a `Logger` class with static methods for logging messages at different levels (info, warning, error).
29. Write a `DateUtils` class with static methods to format dates in different ways.
30. Create a `Configuration` class with static properties for application settings.

Prototype and Prototypal Inheritance
31. Create a `Person` prototype with properties for name and age, and a method to display the person's details.
32. Write a `Car` prototype with properties for make, model, and year, and a method to display the car's details.
33. Implement a `BankAccount` prototype with methods to deposit and withdraw money.
34. Create a `Rectangle` prototype with methods to calculate the area and perimeter.
35. Write a `Circle` prototype with a method to calculate the area.

Modules
36. Create a module that exports a `Person` class and import it in another file to create instances.
37. Write a module that exports a `Calculator` class with methods for basic arithmetic operations,

and import it in another file to use the methods.

38. Implement a module that exports a `LibraryBook` class with methods to borrow and return books, and import it in another file to create instances and use the methods.

39. Create a module that exports utility functions for working with arrays, and import it in another file to use the functions.

40. Write a module that exports a `Logger` class with methods for logging messages, and import it in another file to use the methods.

Mixins

41. Create a mixin to add logging functionality to any class.

42. Write a mixin to add validation methods to a class.

43. Implement a mixin to add event handling capabilities to a class.

44. Create a mixin to add authentication methods to a class.

45. Write a mixin to add serialization methods to a class.

Composition

46. Implement a `Team` class that uses composition to manage a collection of `Player` objects.

47. Create a `Library` class that uses composition to manage a collection of `Book` objects.

48. Write a `Company` class that uses composition to manage a collection of `Employee` objects.

49. Implement a `School` class that uses composition to manage a collection of `Student` objects.

50. Create a `ShoppingCart` class that uses composition to manage a collection of `Product` objects.

These problems cover a wide range of OOP concepts and should provide plenty of practice for mastering OOP in JavaScript.