Java OOPs Assignment Questions – Extended Topics

🔐 Encapsulation

1. Create a class Patient with private fields and implement getter and setter methods with input validation.
2. Why should class fields be private in encapsulation? Illustrate with a program.
3. Modify an existing class Employee to use encapsulation and validate that salary must be greater than 0.
4. How does encapsulation improve maintainability in large applications? Support with an example.
5. Can encapsulation be achieved without setter methods? Explain with a Config class example.

🔁 Polymorphism

6. Demonstrate method overloading in a MathOperations class for add() method (int, float, double).
7. Create a class hierarchy where a superclass method is overridden in multiple subclasses and called through a common reference.
8. Write a program where you use polymorphism to store and invoke different types of PaymentMethod (e.g., CreditCard, UPI, Cash).
9. How does polymorphism support open/closed principle in SOLID design?
10. What happens if you overload a method but only change the return type? Write a program and explain.

🧬 Inheritance

11. Show single-level, multi-level, and hierarchical inheritance using animal classes like Animal, Dog, Cat, and Puppy.
12. Demonstrate the use of super keyword to access the base class members.
13. Can a constructor be inherited in Java? Create an example showing constructor chaining.
14. Implement an inheritance hierarchy for a university system: Person -> Student -> ResearchScholar.
15. What are the risks or limitations of inheritance in Java? Explain with code where inheritance causes confusion.

🔌 Interface

16. Create an interface Playable and implement it in Football, Cricket, and Basketball classes.

17. Can an interface extend another interface? Demonstrate with Device and SmartDevice.
18. Java 8 introduced default methods in interfaces. Explain and implement a default method.
19. Create a Vehicle interface and write classes Car and Bike that implement it. Add interface-based polymorphism.
20. What happens if a class implements two interfaces with the same method signature? Provide an example.

🧱 Abstract Keyword

21. Write an abstract class Notification with an abstract method notifyUser() and concrete method logNotification().
22. Can an abstract class have a main method and be executed? Prove with an example.
23. Implement abstract class Appliance and derived classes WashingMachine, AirConditioner using the operate() method.
24. How does abstraction differ from encapsulation? Write a program showing both in action.
25. What is the use of abstract classes in template-based design? Show a simple real-world template pattern.

📌 Static Keyword

26. Explain the difference between static block and static method with example.
27. Create a class IDGenerator that returns a unique ID using a static counter.
28. Can we override static methods in Java? Explain with a program.
29. Write a program showing shared memory for static variables among multiple objects.
30. How are static methods accessed without object creation? Give multiple examples.

🛑 Final Keyword

31. What happens when we mark a method as final? Create a parent-child class setup to show compile error on override.
32. Can a final class be subclassed? Write code with final class Vehicle and a subclass Car and observe the result.
33. Show the use of final variable with blank initialization in a constructor.
34. Final keyword with static variables: Write a program and explain the use-case.
35. Can final keyword prevent memory leaks or mutability issues? Discuss with a code sample.

🧩 Abstraction

36. Create an abstract class Bank and concrete classes SBI, ICICI, each implementing getRateOfInterest().
37. Write an interface Shape with method draw() and implement it in Circle and Rectangle.
38. Create a real-world example of abstraction using a login system with interface Authenticator.
39. How does abstraction support security and modularity in Java programs?
40. Design a system where abstraction is used to handle payment processing without exposing internal APIs.