1. The "after" code introduces a Computer interface and a ComputerFactory, replacing hard coded logic. But I think it includes unnecessary functions that were not in the original code before.
   Pattern implementation Quality : 3
   Code Readability : 4
   overall improvement : 4

2. The after code has different types of vehicles. It seems like an factory pattern but I'm not sure it correctly changes from the original code.
   Pattern implementation Quality : 2
   Code Readability : 5
   overall improvement : 3

3. The "after" code adds an Animal interface, concrete classes. Even though it is factory pattern, i'm not sure whether it was necessary.
   Pattern implementation Quality : 3
   Code Readability : 3
   overall improvement : 3

4. Refactored code introduces a Shape interface and a ShapeFactory, modularizing shape creation. The implementation is clean and improves flexibility and readability.
   Pattern implementation Quality : 4
   Code Readability : 4
   overall improvement : 4

5. The "after" code centralized database connection logic with a DatabaseConnection interface and a DatabaseConnectionFactory. The design is valid, with improved scalability and cleaner structure.
   Pattern implementation Quality : 5
   Code Readability : 5
   overall improvement : 5

6. Refactored code uses a Logger interface and a LoggerFactory, simplifying the addition of new loggers. The implementation adheres to design principles, enhancing readability and maintainability.

   Pattern implementation Quality : 3
   Code Readability : 5
   overall improvement : 4

7. No refactored code provided. Unchanged. Maybe the no change is the one model recommends.
   Pattern implementation Quality : 5
   Code Readability : 5
   overall improvement : 5

8. no files in test data

9. The "after" code introduces a Payment interface, concrete classes, and a PaymentFactory. The Factory Pattern is well-applied.
   Pattern implementation Quality : 5
   Code Readability : 5
   overall improvement : 5

10. I think it is different code,,?

11. The "after" code adds a Report interface, specific implementations, and a ReportFactory. The implementation is clear, but it miss some component. (fetchdata, spellCheck, fromatReport, and encrypt)
    Pattern implementation Quality : 2
    Code Readability : 5
    overall improvement : 3

12. Refactored code introduces a Notification interface, concrete implementations, and a NotificationFactory. The pattern is implemented well, simplifying readability and making the code flexible.

    Pattern implementation Quality : 5
    Code Readability : 5
    overall improvement : 5

13. No refactored code provided. Unchanged. Maybe similar with 7

14. The "after" code modularizes cache operations using a Cache interface and a CacheFactory. The implementation is flexible and enhances readability.
    Pattern implementation

    Quality : 3 (because it miss some methods)
    Code Readability : 5
    overall improvement : 5

15. Refactored code applies a Filter interface and a FilterFactory, decoupling filtering logic. The design is modular, improving scalability and readability.
    Pattern implementation Quality : 5
    Code Readability : 5
    overall improvement : 5

16. The "after" code uses a Validator interface, concrete classes, and a ValidatorFactory. The factory pattern seems overly implemented for this one, it could be easily replaced with another method.
    Pattern implementation Quality : 5
    Code Readability : 5
    overall improvement : 5

17. Refactored code applies a Parser interface and a ParserFactory to centralize and modularize format-specific parsing logic. The implementation is clear, improving modularity and readability.
    Pattern implementation Quality : 5
    Code Readability : 5
    overall improvement : 5