Spring Core – 30 Industry Level Tasks

1. Design a Notification System where Spring controls which notification service (Email/SMS/Push) is injected at runtime.
2. Build a Payment Gateway Selector where Spring decides the implementation (UPI/Card/NetBanking) based on configuration.
3. Convert a tightly coupled Java application into a loosely coupled Spring-based application using IoC.
4. Implement a Logging Framework where the logger implementation is managed by the Spring container.
5. Create a File Storage System where Spring switches between Local and Cloud storage without code changes.
6. Develop a Report Generator where Spring manages different report formats such as PDF and Excel.
7. Demonstrate how IoC improves unit testing by replacing real services with mock beans.
8. Implement constructor injection for a Banking Service and explain why it is preferred in production systems.
9. Use setter injection for optional dependencies in a User Profile module.
10. Refactor a field-injected service to constructor injection as part of a code quality improvement task.
11. Resolve multiple bean ambiguity using @Qualifier in a real-world application.
12. Use @Primary to define a default bean when multiple implementations exist.
13. Inject primitive values and collections using Spring dependency injection.
14. Demonstrate a circular dependency issue and resolve it using setter injection.
15. Implement the Strategy Design Pattern using Spring dependency injection.
16. Create a bean and trace its complete lifecycle from instantiation to destruction.
17. Use @PostConstruct and @PreDestroy in a database connection management bean.
18. Implement InitializingBean and DisposableBean and compare them with lifecycle annotations.
19. Configure lazy initialization for heavy beans to improve application startup performance.
20. Demonstrate singleton and prototype scopes using a real business scenario.
21. Implement resource cleanup logic using destroy methods.
22. Explain how Spring manages beans internally using ApplicationContext.
23. Create a custom BeanPostProcessor to modify bean behavior after initialization.
24. Implement logging using Spring AOP for all service layer methods.
25. Add performance monitoring using AOP to measure method execution time.
26. Simulate transaction management behavior using Spring AOP without a database.
27. Implement a security validation aspect before executing business logic.
28. Log exceptions using @AfterThrowing advice in Spring AOP.
29. Use pointcut expressions to target specific packages and methods.
30. Build an auditing mechanism using AOP to track method calls and input parameters.