Nomura Java Developer Interview – Detailed Answers (3–7 Years)

1. What is record datatype and why is it required?

Record is a special type of class introduced in Java 14 (preview) and made stable in Java 16. It is used to create immutable data carrier classes with minimal boilerplate code. A record automatically generates constructor, getters, equals(), hashCode(), and toString() methods. Records are required to reduce verbosity, improve readability, and clearly represent data-centric objects. They are best suited for DTOs, value objects, and API response models.

2. What are sealed classes and why are they required?

Sealed classes (introduced in Java 15 as preview and finalized in Java 17) restrict which classes can extend or implement them. Using the 'sealed', 'permits', 'non-sealed', and 'final' keywords, developers can control inheritance. They are required to improve security, maintain strict domain modeling, and ensure better pattern matching by limiting subclass hierarchies.

3. Explain SOLID principles with examples

S – Single Responsibility Principle: A class should have only one reason to change. Example: A UserService should not handle database logging.

O – Open/Closed Principle: Classes should be open for extension but closed for modification. Example: Using interfaces to add new payment methods.

L – Liskov Substitution Principle: Subclasses must be replaceable by their parent classes without breaking functionality.

I – Interface Segregation Principle: Clients should not be forced to implement unnecessary methods. Example: Separate Printer and Scanner interfaces.

D – Dependency Inversion Principle: High-level modules should depend on abstractions, not implementations. Example: Injecting interfaces using Spring Dependency Injection.

4. How many .class files will be generated when one inner class is there inside a class?

When a class contains one inner class, the Java compiler generates two .class files: 1. OuterClass.class 2. OuterClass$InnerClass.class Each inner class results in a separate .class file.

8. Why do we use @Transactional in Spring Boot application?

@Transactional is used to manage database transactions declaratively in Spring Boot. It ensures ACID properties (Atomicity, Consistency, Isolation, Durability). If an exception occurs, the transaction is automatically rolled back. It reduces boilerplate code and provides better consistency for database operations.

9. Group by department and find the highest salary of an employee in SQL

This is achieved using GROUP BY with aggregate function MAX(). The query groups employees by department and retrieves the highest salary in each department. This is commonly used in reporting and analytics use cases.

10. What is repository in Spring and why do we use it?

A repository in Spring is a data access abstraction used to interact with databases. It is part of Spring Data JPA and acts as a bridge between the application and persistence layer. Repositories reduce boilerplate code, provide built-in CRUD operations, and support custom queries using method names or JPQL.