4.Explain ORM implementation with Hibernate XML Configuration and Annotation Configuration

Hibernate provides two main ways to implement Object-Relational Mapping (ORM): XML Configuration and Annotation-based Configuration. Both methods allow developers to map Java objects (entities) to database tables, manage sessions, and perform CRUD operations.

Hibernate supports ORM implementation using:

- **XML Configuration**, where mappings are defined in external ..hbm.xml files.
- **Annotation Configuration**, where mappings are done directly in Java classes using JPA annotations like @Entity, @Table, @Id, and @Column.

1. **ORM Using Hibernate with XML Configuration**
    - Persistent Class

      A simple POJO (Plain Old Java Object) with private fields and public getters/setters
    - Mapping XML

      Defines how class properties map to table columns (.hbm.xml file).
    - Hibernate Configuration XML

      Contains database connection and Hibernate settings (hibernate.cfg.xml).
2. **ORM Using Hibernate with Annotation Configuration**
    - Persistent Class

      Annotated POJO with JPA annotations like @Entity, @Id, etc.
    - Hibernate Configuration XML

      Similar to XML config but no need for mapping file.

Mapping Style:

- XML Configuration: Mapping is defined in separate .hbm.xml files.
- Annotation Configuration: Mapping is done directly in the Java class using annotations.

Readability:

- XML Configuration: Less readable, as mappings are maintained separately from the code.
- Annotation Configuration: More readable, since mappings are in the same file as the class.

Maintainability:

- XML Configuration: Harder to maintain, especially in large applications with many entities.
- Annotation Configuration: Easier to maintain and refactor as everything is in one place.

Modern Usage:

- XML Configuration: Used in legacy or enterprise systems with strict separation of concerns.
- Annotation Configuration: Preferred in modern Spring and Hibernate projects for its simplicity.