1. Entity Mapping.

Overview: Entity mapping is the process of connecting Java objects (entities) to database tables in a way that the state of the objects can be stored, retrieved, and managed. This is a key aspect of Object-Relational Mapping (ORM), where the entity class represents a table in the database, and the fields of the entity class represent the columns of that table.

Key Concepts:

- **Entity Class:**
    - A Java class annotated with @Entity represents a table in the database.
    - The class must have a primary key field, typically annotated with @Id.
- **Table Mapping:**
    - Use the @Table annotation to specify the name of the database table and other details like schema.
- **Field Mapping:**
    - Each field in the entity class corresponds to a column in the database table.
    - The @Column annotation allows customization of the column name, length, nullable, etc.

2. JPA Annotations

Overview: Java Persistence API (JPA) provides a set of annotations to map entities, relationships, and inheritance hierarchies in a database. These annotations make it easy to define how Java classes and their fields correspond to database structures.

Key Annotations:

- **@Entity**: Specifies that the class is an entity.
- **@Table**: Specifies the table name and other table-related information.
- **@Id**: Marks the primary key of the entity.
- **@GeneratedValue**: Defines the strategy for primary key generation (e.g., IDENTITY, SEQUENCE).
- **@Column**: Configures the column properties like name, length, nullable, etc.
- **@OneToOne, @OneToMany, @ManyToOne, @ManyToMany**: Defines relationships between entities.
- **@JoinColumn**: Specifies the foreign key column in a relationship.

3. Persistence Context

Overview: The persistence context in JPA is a managed environment in which entity instances are managed by the EntityManager. It keeps track of the entities and their state, ensuring synchronization with the database.

Key Concepts:

- **EntityManager:**
  - The EntityManager manages the persistence context.
  - It provides an API to perform CRUD operations on entities.
- **Persistence Context:**
  - It is a set of entity instances where the identity of each entity is managed within a particular persistence context.
  - The EntityManager controls the lifecycle of entities in this context.
  - Entities within a persistence context are tracked, and any changes made to them are synchronized with the database.
- **Transaction Management:**
  - Changes made to entities are not immediately persisted to the database.
  - The persistence context ensures that all changes are committed as a single transaction.

4. Entity Lifecycle

Overview: Entities in JPA have a well-defined lifecycle that is managed by the persistence context. The lifecycle includes states such as transient, persistent, detached, and removed, which define the entity's relationship with the persistence context.

Key Lifecycle States:

- **Transient:**
  The entity is created but not associated with a persistence context, and not yet persisted in the database.
- **Persistent:**
  The entity is managed by the persistence context. It is stored in the database, and any changes to it are tracked.
- **Detached:**
  The entity is no longer associated with the persistence context but still exists in the database.

- **Removed:**
  The entity is marked for deletion from the database. It is still in the persistence context until the transaction is committed.

Summary

1. **Entity Mapping:** The process of mapping Java classes to database tables.
2. **JPA Annotations:** Common annotations used for entity mapping and relationship management.
3. **Persistence Context:** The environment where entities are managed by the EntityManager.
4. **Entity Lifecycle:** The different states an entity goes through during its lifecycle.