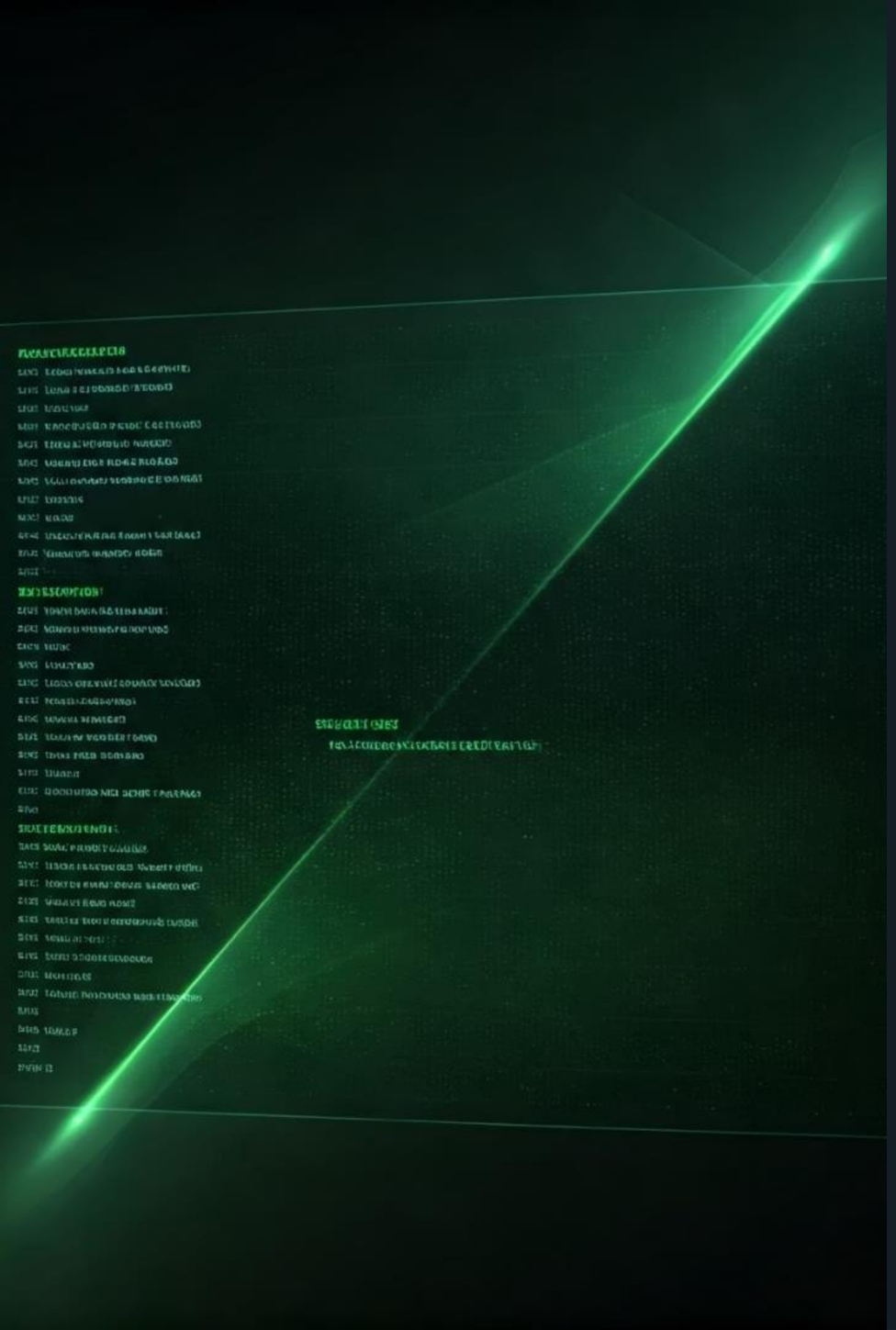


Introduction to JPA: Core Principle and Overview

Java Persistence API (JPA) maps Java objects to database tables. This ORM lets developers avoid direct SQL, working instead with familiar Java entities. It simplifies database operations and enforces object-oriented data management.

Fundamentals of Object-Relational Mapping

Entity Mapping

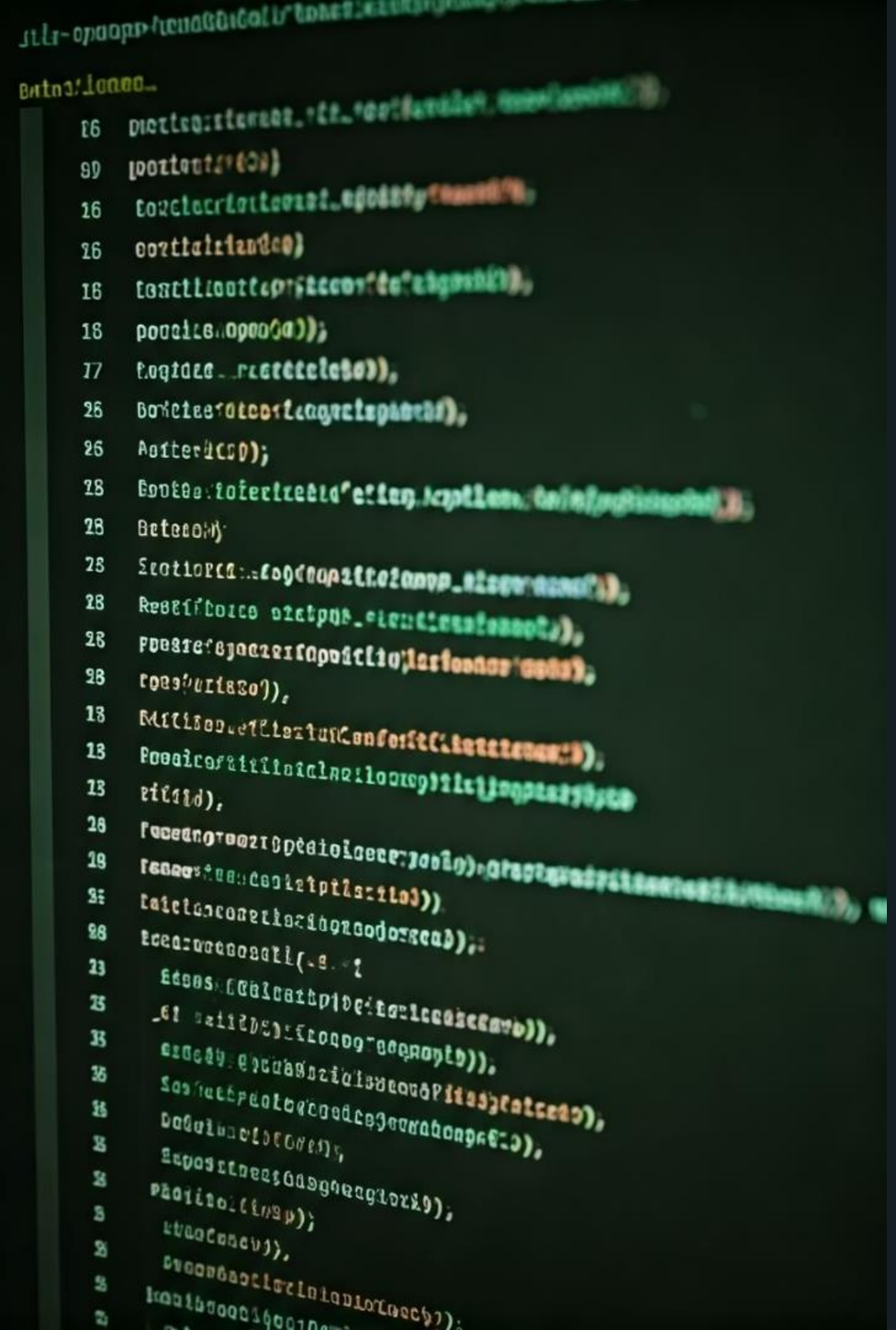
Java classes map to database tables.

Fields correspond to table columns.

Relationships

Define one-to-many, many-to-one, and more.

Relationship keys maintain data integrity.



Entity Management and Annotations

@Entity

Marks class as persistent entity.

@Id, @Column

Identify primary key and map fields.

@OneToMany, @ManyToOne

Define relations between entities.

CRUD Operations with EntityManager

1 Create

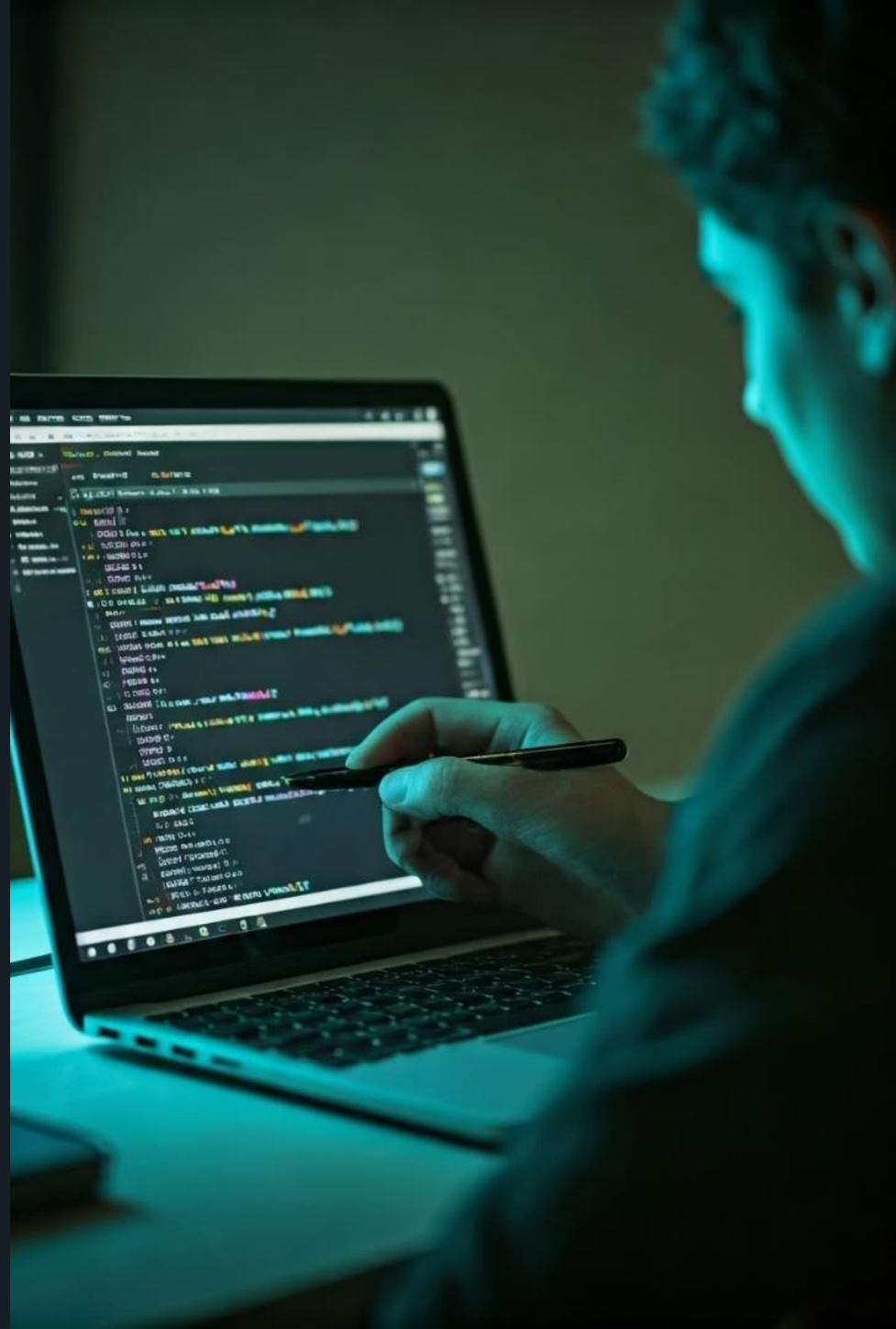
Persist new objects to the database.

2 Read

Retrieve entities by ID or queries.

3 Update & Delete

Automatically track and apply changes.



Advanced Querying: JPQL & Native SQL

JPQL

Object-oriented, similar to SQL

Native SQL

Direct SQL queries when needed

Flexible Queries

Supports complex filters and joins

Transaction Management and Caching

Transactions

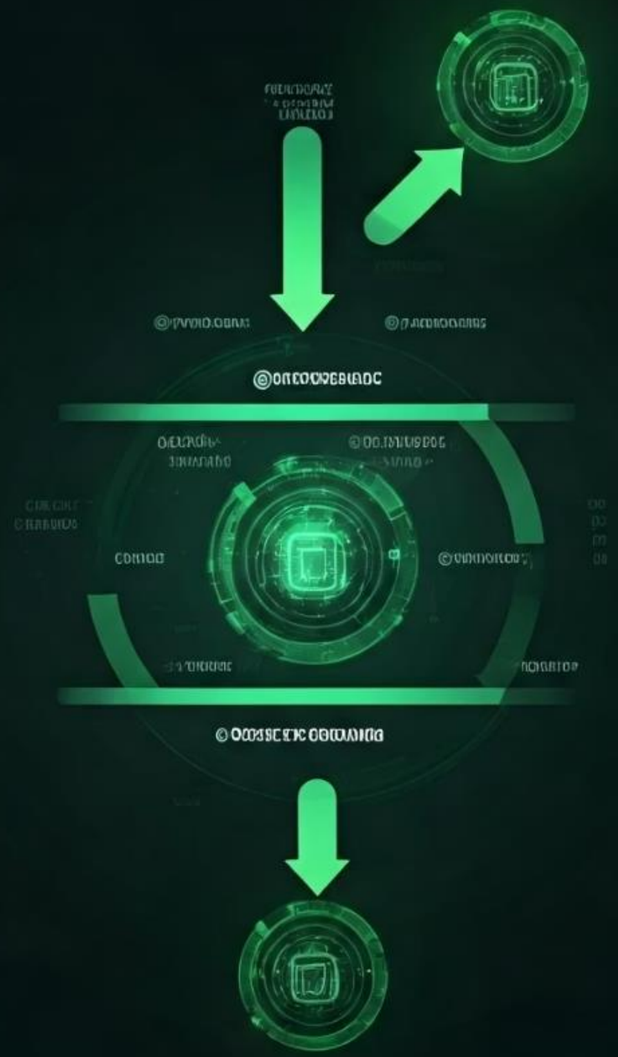
Declarative and programmatic management ensure data consistency.

First-level Cache

Built-in cache per EntityManager session.

Second-level Cache

Provider-dependent, improves performance across sessions.



Entity Lifecycle Callbacks

@PrePersist

Triggered before an entity is persisted.

@PostLoad

Executed after an entity is loaded.

Other Callbacks

Include @PreUpdate, @PostRemove, offering fine-grained control.

JPA Providers and Portability

- JPA is a specification, not an implementation.
- Popular providers: Hibernate, EclipseLink, OpenJPA, DataNucleus.
- Portability allows switching providers without code changes.

Choose providers based on features, performance, and community support.