Java Persistence API (JPA): Summary Notes

1. Overview:
The Java Persistence API (JPA) is a specification used for Object-Relational Mapping (ORM) in Java. It maps Java objects to relational database tables, simplifying data management and eliminating the need for direct SQL queries.

2. Object-Relational Mapping (ORM) Concepts:
• Entity – Represents a table in the database.
• Persistence Context – Environment where entity instances are managed.
• Entity Manager – Interface for performing CRUD operations.
• Relationships – Define associations using annotations like @OneToOne, @OneToMany, @ManyToOne, @ManyToMany.

3. JPA Annotations for Mapping:
• @Entity – Marks a class as a persistent entity.
• @Table(name="table_name") – Specifies the database table name.
• @Id – Marks the primary key field.
• @GeneratedValue(strategy=GenerationType.IDENTITY) – Specifies auto-generation of primary key values.
• @Column(name="column_name") – Maps a field to a specific column.
• @Transient – Marks a field that should not be persisted.
• @JoinColumn – Defines a foreign key column for relationships.

Example Entity:
```
@Entity
@Table(name = "students")
public class Student {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "student_name")
    private String name;

    @Column(name = "email")
    private String email;
}
```

4. JPQL (Java Persistence Query Language):
JPQL is an object-oriented query language used to query entities instead of database tables. It supports SELECT, UPDATE, and DELETE operations and provides portability across databases.

```
// Select all students
Query q1 = em.createQuery("SELECT s FROM Student s");

// Select students with a specific name
Query q2 = em.createQuery("SELECT s FROM Student s WHERE s.name = :name");
q2.setParameter("name", "Ravi");

// Update example
em.createQuery("UPDATE Student s SET s.email = 'new@mail.com' WHERE s.id = 1").execute
```