**J2EE Layered Architecture Overview**

**J2EE (Java 2 Platform, Enterprise Edition)**, now known as **Jakarta EE**, follows a **multi-tier architecture** designed to separate application logic into layers. This separation improves:

* **Scalability**
* **Maintainability**
* **Reusability**
* **Security**

**🔹 1. Client Layer (Presentation Layer / Front-End)**

**Role:**

* Interface between **users and the application**.
* Captures user input and displays output.

**Technologies:**

* **HTML/CSS, JavaScript**
* JSP (JavaServer Pages)
* JSF (JavaServer Faces)
* Angular, React (modern front-ends)
* Mobile apps (Android/iOS) consuming REST APIs

**Real-world Example:**

* **Banking portal** UI where customers check balances or request statements.
* **Telecom mobile app** where users recharge or check data balance.

**🔹 2. Web Layer (Controller Layer / Servlet Layer)**

**Role:**

* Accepts client requests (usually HTTP)
* Acts as a controller and passes request to the business layer
* Controls navigation and session handling

**Technologies:**

* **Servlets**
* **JSP**
* **Filters**
* **Struts/Spring MVC** (in modern Java)

**Real-world Example:**

* A servlet validating a telecom customer’s login before showing their usage.
* JSP displaying e-commerce order confirmation.

**🔹 3. Business Layer (Service Layer / Logic Layer)**

**Role:**

* Implements **business logic**.
* Coordinates tasks between components.
* Handles transactions, security, and exception management.

**Technologies:**

* **Enterprise JavaBeans (EJB)**
* **Spring Framework / Spring Boot**
* **POJOs with annotations**
* **Dependency Injection** frameworks

**Real-world Example:**

* **Loan processing logic** in a banking application.
* **Data usage calculation** logic in telecom.
* **Discount coupon validation** in an e-commerce checkout.

**🔹 4. Integration Layer (Persistence / Data Access Layer)**

**Role:**

* Accesses and manages data from the underlying database or legacy systems.
* Converts Java objects into relational data (and vice versa).

**Technologies:**

* **JDBC (Java Database Connectivity)**
* **JPA (Java Persistence API)**
* **Hibernate**
* **JTA (for transactions)**

**Real-world Example:**

* JDBC call fetching account history from Oracle DB in a bank.
* JPA entity storing a user’s SIM card activation record.

**5. Enterprise Information System (EIS) Layer / Resource Layer**

**Role:**

* Interfaces with **external systems** like databases, ERPs, or message brokers.
* Often managed via connectors (JCA - Java Connector Architecture).

**Technologies:**

* **Database (Oracle, MySQL, PostgreSQL)**
* **Messaging systems (JMS, Kafka)**
* **Mainframes, ERP connectors**
* **NoSQL (MongoDB)**

**Real-world Example:**

* Sending transaction data to **SWIFT network** in banking.
* Publishing a usage event to **Kafka queue** in telecom billing system.

**Diagram of J2EE Architecture**

+----------------------+

| Client (Browser) |

+----------------------+

↓

+---------------------------+

| Web Layer (Servlet/JSP) |

+---------------------------+

↓

+---------------------------+

| Business Layer (EJBs) |

+---------------------------+

↓

+----------------------------------------+

| Integration Layer (JPA/JDBC/Hibernate) |

+----------------------------------------+

↓

+---------------------------+

| Database / Legacy System |

+---------------------------+

**Benefits of J2EE Layered Architecture**

| **Feature** | **Benefit** |
| --- | --- |
| **Modularity** | Code is separated by concern (UI, Logic, Data) |
| **Maintainability** | Easy to update one layer without affecting others |
| **Reusability** | Business logic can be reused across multiple UIs |
| **Security** | Authentication and role-based access managed clearly |
| **Scalability** | Each layer can be scaled independently |

**Industry Use Case Example: Telecom Recharge Portal**

| **Layer** | **Functionality** |
| --- | --- |
| Client Layer | User enters mobile number and selects plan |
| Web Layer | Servlet receives form, validates input |
| Business Layer | Checks if user is valid, applies promo code logic |
| Data Layer | Fetches current balance, stores new recharge |
| EIS Layer | Sends transaction message to billing server or Kafka topic |