**Detailed Technical Explanation**

This service is part of a Spring Boot microservice responsible for managing **loyalty point calculations and legacy data migration** for different services like **Hotel**, **Casino**, and **Restaurant**.

**Purpose**

The LoyaltyService is the **core business logic layer** in this loyalty application. It acts as the service component in a typical **3-tier architecture** (Controller → Service → Repository), handling:

* Migration of legacy data
* Calculation of loyalty points
* CRUD operations on loyalty points
* Aggregation of loyalty summaries

**Design Patterns Used**

**1. Strategy Pattern**

The most important design pattern used here is the **Strategy Pattern**:

* The LoyaltyMigrationStrategy is a **common interface** implemented by different strategies (HotelStrategy, CasinoStrategy, etc.).
* The strategyMap dynamically maps each strategy type (HOTEL, CASINO, RESTAURANT) to its implementation.
* This allows the application to be **open for extension but closed for modification** (SOLID principles).

**2. Dependency Injection (DI)**

Spring's @Autowired annotation ensures **loose coupling** and **testability** by injecting repository and strategy dependencies automatically.

**3. Repository Pattern**

Used to abstract data access logic using interfaces such as:

* UserRepository
* LoyaltyPointRepository
* HotelLegacyRepository, etc.

This pattern allows clean separation between business logic and persistence.

**4. DTO Pattern**

Data Transfer Objects (LoyaltySummaryDTO, SourcePointsDTO, etc.) are used to **encapsulate response data**, ensuring no exposure of internal entities.

**Method-Level Explanation**

**✅public void migrateLegacyDataToLoyaltyPoints(Long userId)**

* **Purpose:** Migrate data from legacy hotel/casino/restaurant tables into the LoyaltyPoint table.
* **How:** Iterates over each LoyaltyMigrationStrategy and saves the resulting loyalty points.

**✅ public LoyaltySummaryDTO getUserLoyaltySummary(Long userId)**

* **Purpose:** Provides a summarized view of total loyalty points by category for a user.
* **Logic:** Fetches user’s name and all loyalty points, categorizes them, and returns a DTO.

**✅ public LoyaltyDetailsDTO getUserLoyaltyDetails(Long userId)**

* **Purpose:** Gives detailed breakdown of loyalty points by individual sources (hotel names, casino branches, etc.).
* **Logic:** Groups loyalty points into a map by type and converts them to a detailed DTO.

**✅ public void addLoyaltyPoints(LoyaltyAddRequestDTO dto)**

* **Purpose:** Allows manual addition of loyalty points.
* **Logic:** Uses the calculatePoints method to determine points based on billing amount and saves the record.

**✅ public void updatePoints(Long id, int newPoints)**

* **Purpose:** Update the points of an existing record.
* **Logic:** Finds the record, updates the points, and saves.

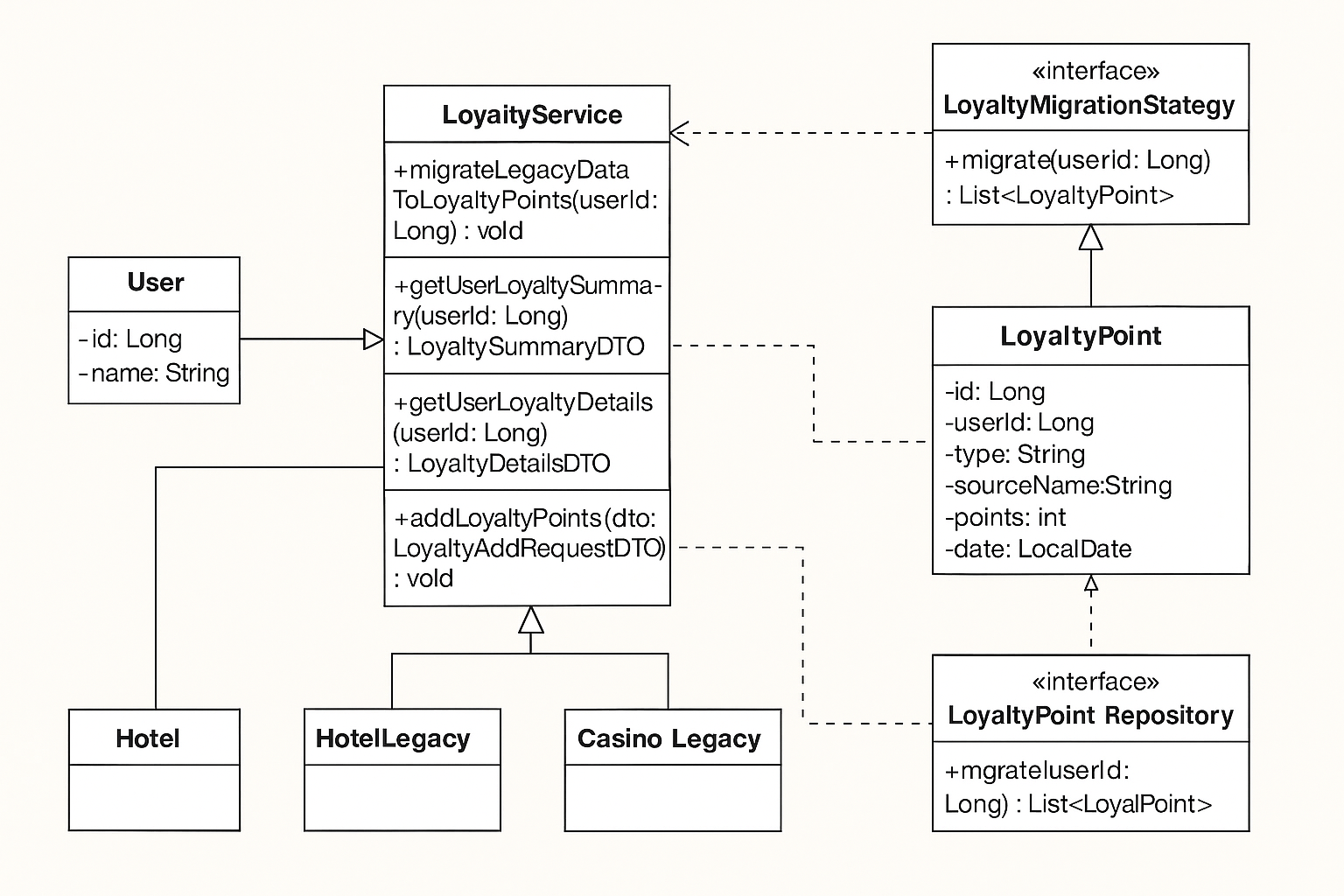
**✅ public void deletePoints(Long id)**

* **Purpose:** Deletes a loyalty point record by ID.

**✅ private int calculatePoints(String type, double billAmount)**

* **Purpose:** Business logic to determine points based on the bill amount.

**Class diagram :**



**Database Details:**

**Users Table**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| id | BIGINT (PK) | Unique user ID |
| name | VARCHAR | User's full name |
| email | VARCHAR | User's email |
| created\_at | TIMESTAMP | Account creation time |

**Sample Data:**

| **id** | **name** | **email** | **created\_at** |
| --- | --- | --- | --- |
| 1 | John Doe | john@example.com | 2023-01-01 12:00:00 |
| 2 | Jane Roe | jane@example.com | 2023-02-05 14:30:00 |

**Hotel Legacy Table**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| id | BIGINT (PK) | Unique record ID |
| user\_id | BIGINT (FK) | Foreign Key to users table |
| hotel\_name | VARCHAR | Name of hotel |
| bill\_amount | DECIMAL | Total spend |
| stay\_date | DATE | Date of stay |

**Sample Data:**

| **id** | **user\_id** | **hotel\_name** | **bill\_amount** | **stay\_date** |
| --- | --- | --- | --- | --- |
| 1 | 1 | Hilton Resort | 1000 | 2010-05-01 |
| 2 | 2 | Marriott | 500 | 2012-11-10 |

**Casino Legacy Table**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| id | BIGINT (PK) | Unique record ID |
| user\_id | BIGINT (FK) | Foreign Key to users table |
| casino\_name | VARCHAR | Name of casino |
| bill\_amount | DECIMAL | Total spend |
| visit\_date | DATE | Date of visit |

**Sample Data:**

| **id** | **user\_id** | **casino\_name** | **bill\_amount** | **visit\_date** |
| --- | --- | --- | --- | --- |
| 1 | 1 | Vegas Grand | 10000 | 2009-03-15 |
| 2 | 2 | Lucky 7 | 20000 | 2011-08-21 |

**Restaurant Legacy Table**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| id | BIGINT (PK) | Unique record ID |
| user\_id | BIGINT (FK) | Foreign Key to users table |
| restaurant\_name | VARCHAR | Name of restaurant |
| bill\_amount | DECIMAL | Total spend |
| dining\_date | DATE | Date of dining |

**Sample Data:**

| **id** | **user\_id** | **restaurant\_name** | **bill\_amount** | **dining\_date** |
| --- | --- | --- | --- | --- |
| 1 | 1 | Olive Garden | 200 | 2014-12-20 |
| 2 | 2 | BBQ Nation | 150 | 2013-07-30 |

**Loyalty Points Table**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| id | BIGINT (PK) | Auto ID |
| user\_id | BIGINT (FK) | Foreign Key to users table |
| type | VARCHAR | "HOTEL", "CASINO", "RESTAURANT" |
| source\_name | VARCHAR | Hotel/Casino/Restaurant name |
| points | INT | Calculated loyalty points |
| added\_date | DATE | When it was added to this table |

**Sample Data After Migration:**

| **id** | **user\_id** | **type** | **source\_name** | **points** | **added\_date** |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | HOTEL | Hilton Resort | 100 | 2024-05-12 |
| 2 | 1 | CASINO | Vegas Grand | 10000 | 2024-05-12 |
| 3 | 1 | RESTAURANT | Olive Garden | 40 | 2024-05-12 |