**Common Methods in CriteriaBuilder**

* cb.equal(x, y): Creates a condition for equality (x = y).
* cb.notEqual(x, y): Creates a condition for inequality (x != y).
* cb.greaterThan(x, y): Creates a condition for x > y.
* cb.lessThan(x, y): Creates a condition for x < y.
* cb.like(x, pattern): Creates a condition for pattern matching (LIKE).
* cb.isNull(x): Checks if x is NULL.
* cb.isNotNull(x): Checks if x is not NULL.
* cb.and(predicates...): Combines multiple predicates with AND.
* cb.or(predicates...): Combines multiple predicates with OR.
* CriteriaBuilder: Factory for query parts
* CriteriaQuery: Represents a query object
* Root: Represents table/entity
* Predicate: WHERE conditions
* **CriteriaBuilder**: Entry point to create queries and predicates.
* **CriteriaQuery<T>**: Represents the actual query; T denotes the expected result type.
* **Root<T>**: Refers to the root entity from which attributes and joins are derived.
* **Predicate**: Represents conditions in WHERE or HAVING clauses.
* **Expression<T>**: Used for calculations, conditions, or data transformations.

**Why Use Specification + CriteriaBuilder?**

**Benefits of Using Specification and CriteriaBuilder:**

• **Cleaner Code**: Avoid cluttering the repository with numerous query methods.

• **Reusable Filters**: Easily reuse query components across multiple parts of the application.

• **Flexibility**: Dynamic filtering based on any combination of parameters.

• **Type-Safe**: Avoid raw SQL or JPQL, which can be error-prone.