Loan Lenders

Description

Objective:

To work with One to Many Relationship between entities and implement as Bidirectional and usage of group function query method

Concept Explanation:

1. When one instance of an entity is associated with multiple instances of another entity, then we call that relationship **one-to-many.**
2. That relationship will be **bi-directional** if both the entities are aware of each other and can navigate from one entity to another.
3. The **Group Function Query** Method in JPA involves using aggregate functions like `COUNT`, `MAX`, `MIN`, `SUM`, etc., along with grouping criteria to perform operations across groups of data

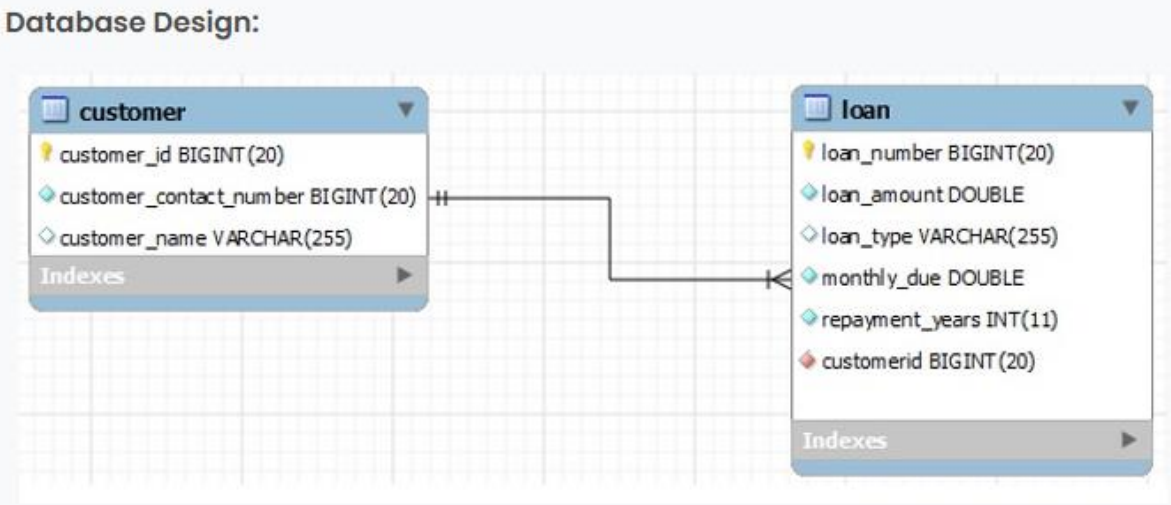
Concept Implementation:

1. We establish a one-to-many relationship between the Customer and Loan entities by providing appropriate annotation with the proper attribute pointing to the Customer above the loanList attribute in the Customer entity.
2. The Loan entity holds a reference to the Customer entity, establishing a many-to-one relationship, where each loan is associated with one Customer. Provide proper annotation that specifies the foreign key column in the Loan entity, enabling bidirectional retrieval of Customer and Loan details.
3. Use JPA Custom Query methods with the proper group functions to find the loan with the maximum due amount and the minimum repayment duration in years in the appropriate repository interface

Loan Lenders

National Bank wants to manipulate and store the details of the different loans taken by the customers from their bank. Develop a Spring Data JPA application to perform the task.

Database Design:



Note:

- Customer and Loan has one to many relationships.
- Establish the relationship between Customer and Loan as Bi-directional.

Functionalities:

1. Add Customer

National Bank will store each customer information by providing the details like customer id, customer name and customer contact number. This information will be stored in the Customer table.

Utility class:

Component Name	Method Name	Method Description	Arguments	Output
BankDAO	addCustomer	This method stores the customer details in the Customer table	Customer customer	void

Model class

Component Name	Attributes	Methods
Customer	cutomerId - long customerName - String customerContactNumber - long The loan details - the attribute name should be "loanList"	getter and setter methods
Loan	loanNumber - long loanType - String loanAmount - double repaymentYears - int monthlyDue - double The customer information- the attribute name should be "customer".	getter and setter methods

2. Allocate a Loan to the Customer

To the already existing customer**,** the National Bank wants to add a loan by providing the details like loan number, loan type, loan amount, repayment years and monthly due amount.

Utility class:

Component Name	Method Name	Method Description	Arguments	Output
BankDAO	allocateLoanTo Customer	To the already existing customer add the loan. This method should add the loan object to the "Loan" table.	long customerId, Loan loan	Void

3. Find

This functionality should list out the loan which has the maximum monthly due amount and the minimum repayment duration in years. (Hint: One or more loan may have the maximum monthly due amount and the minimum repayment duration in years)

Utility class:

Component Name	Method Name	Method Description	Arguments	Output
BankDAO	loanWithMaxDueMinYears	This method should return the list of loan which has the maximum monthly due amount and the minimum repayment duration in years.		List<Loan>

Design Constraints

1. All the classes and methods should have public access Specifiers.

2. All the attributes should have private access Specifiers.

3. Use Annotation for all the mapping.

4. customer_id and loan_number should be marked as the primary key.

5. Use Spring Data JPA API for persisting the object into the database.

6. The above tables should get created by hibernate automatically with the proper parent and child relationship.

7. The table should be created with the correct table name and column name as specified in the database design diagram.

8. Do not alter the code skeleton and do not include any additional packages.

9. The column names and method name should be given as specified in the document.

10. Do not change the property name given in the application.properties files, but you can change the value and you can include additional property if needed.

11. In the pom.xml we are given with all the dependencies needed for developing the application.