**Hands on 2**

**How object to relational database mapping is done in Hibernate XML configuration file**

In Hibernate, object-to-relational mapping means linking Java class fields to columns in a database table. This mapping is done using an XML file with the .hbm.xml extension. In this XML file, we mention the class name and table name using the <class> tag. Inside that, we specify the primary key using <id> and other fields using <property>. Each field is matched with a column in the table. For example, if we have a Country class with fields code and name, we create an XML mapping file where we mention that code maps to the column co\_code and name maps to co\_name. This is how Hibernate knows which field goes to which column in the database.

### Explanation of implementing the end-to-end operations in Hibernate

#### SessionFactory:

SessionFactory is a factory class in Hibernate which is used to create Session objects. It is created once during the application startup using the configuration file. It is a heavyweight object and should be created only once. All database interactions happen through the sessions created by this factory.

#### Session:

Session is the main interface between the Java code and the database. It is used to perform create, read, update, and delete (CRUD) operations. Each session represents a single unit of work with the database and is created using the SessionFactory.

#### Transaction

In Hibernate, transactions are used to group multiple operations together so that they are either all successful or all rolled back if something goes wrong. The transaction ensures data consistency and integrity in the database.

#### beginTransaction()

The beginTransaction() method is used to begin a new database transaction. It marks the starting point of a group of operations that should be executed as one unit. After calling this method, Hibernate knows that the operations should be treated as a single transaction.

#### commit()

The commit() method is used to save the changes made during the transaction permanently into the database. If all the operations are successful, we call commit() to make sure the changes are stored and not lost.

#### rollback()

The rollback() method is used when something goes wrong during the transaction. It cancels all the changes made during the transaction and brings the database back to its previous state. This helps avoid storing incomplete or wrong data.

#### session.save()

The session.save() method is used to insert a new record into the database. It takes a Java object and stores it as a new row in the corresponding table. The values of the fields are mapped to the table columns based on the XML configuration.

#### session.createQuery().list()

The createQuery().list() method is used to run HQL (Hibernate Query Language) queries and fetch multiple records from the database. It returns a list of Java objects that match the query. This is mostly used to retrieve all data from a table.

#### session.get()

The session.get() method is used to fetch a single row from the database based on the primary key. It takes the class name and the id value, and returns the matching object from the database table.

#### session.delete()

The session.delete() method is used to delete a record from the database. It takes a Java object and removes the matching row from the table. The object must exist in the session or be fetched before calling delete.