# MINI PROJECT TWO READ ME

**Aim:**

The purpose of this assignment is to explore object relational mapping (ORM) and the Java Persistence API (JPA) by designing and implementing a domain model with basic EntityManager functionality and work with NetBeans tooling to reverse engineer the MP1 database to JPA Entities, and make use of JUnit test cases

**Project Summary:**

* There are two java projects:
  + MP2 RE – This project is for Reverse Engineering MP1 database
  + MP2 Domain – This project is for mapping the objects of a Java class to the relational tables using ORM and JPA
* In **MP2 RE,** the “world” database (used in MP1) was reverse engineered using the Netbeans tooling “Entity Classes from Databases” to obtain JPA Entities in the form of four JAVA classes – City, Country, CountryLanguage and CountryLanguagePK which corresponded to all the relationships between the three relational tables present in the database
* In **MP2 Domain**, using a domain model, different entities were created using Java Persistence Unit and relationships were declared by different annotations. Use of JUnit test cases was made in order to execute the JPQL queries in order to make appropriate use of JUnit test fixtures as mentioned in the requirement

**Design & Development Insights:**

* The MP2 RE was a straightforward creation of entities using netbeans tool so would like to jump on design and development insights of MP2 Domain
* Initial setup involved creation of a database named tgajare\_itmd4515 and adding JPA , EclipseLink and MySQL JDBC libraries
* Created a Persistence Unit and connected it to the tgajare\_itmd4515 database
* Configured persistence.xml to drop­and­create the tables, as well as to generate both create.ddl and drop.ddl scripts
* For this project I chose the domain model consisting of 5 +1 entities:
  + Brand
  + Designer
  + Product
  + Customer
  + PurchaseOrder
  + Address
* Each entity has specific attributes giving more details about either the brand, designer, product, customer or order
* Customer and PurchaseOrder have @Temporal annotation in order to specify either the date of birth or date of order and time
* In all I have incorporated 5 relationships amongst all the entities:
  + One-one unidirectional between Designer and Brand
  + Many-Many bidirectional between Product and Orders
  + One-Many bidirectional between Customer and Product
  + One-Many bidirectional between Designer and Product
  + One-Many unidirectional between Customer and Order
* The fifth entity is Address which is annotated to be as Embeddable and used in Customer and Order
* Have considered following roles for future:
  + Designer
  + Customer
  + Admin
* Created a Junit test case to implement BeforeClass, AfterClass, Before, After to obtain an EntityManagerFactory, EntityManager and EntityTransaction
* Initialized database with data and persisted all Entity objects
* Displayed all the objects through established relationships
* Performed an update (Updated brand name) and remove (removed a customer entry) operation using EntityManager

**Requirements:**

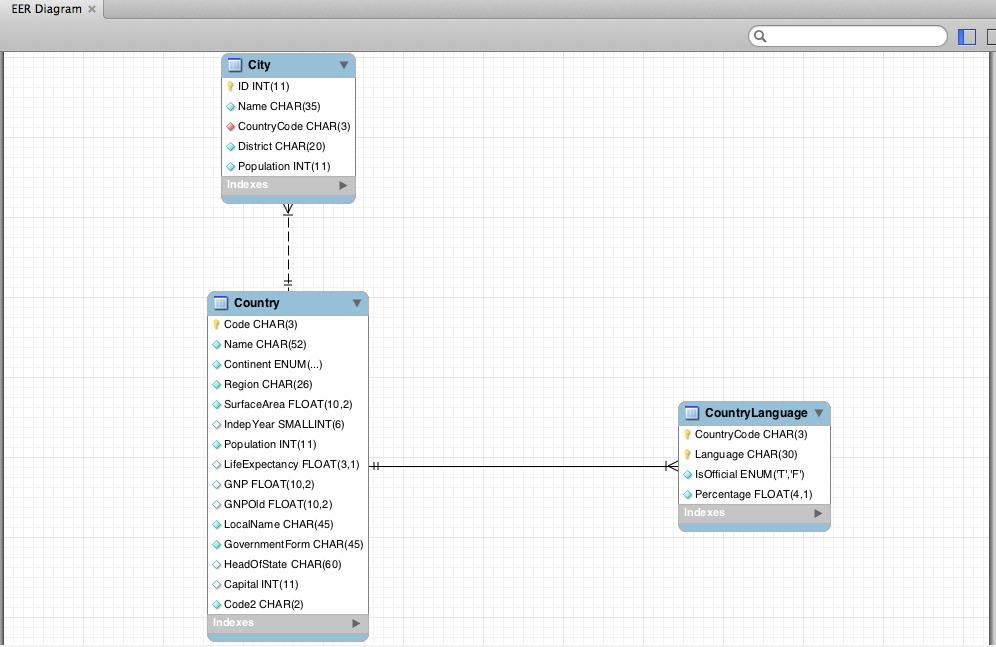
1. OS: Mac OX 10.8
2. Installation: MySQL WorkBench, NetBeans and JDK 7 for macosx
3. MySQL user: itmd4515 with password as itmd4515

**Expected Results:**

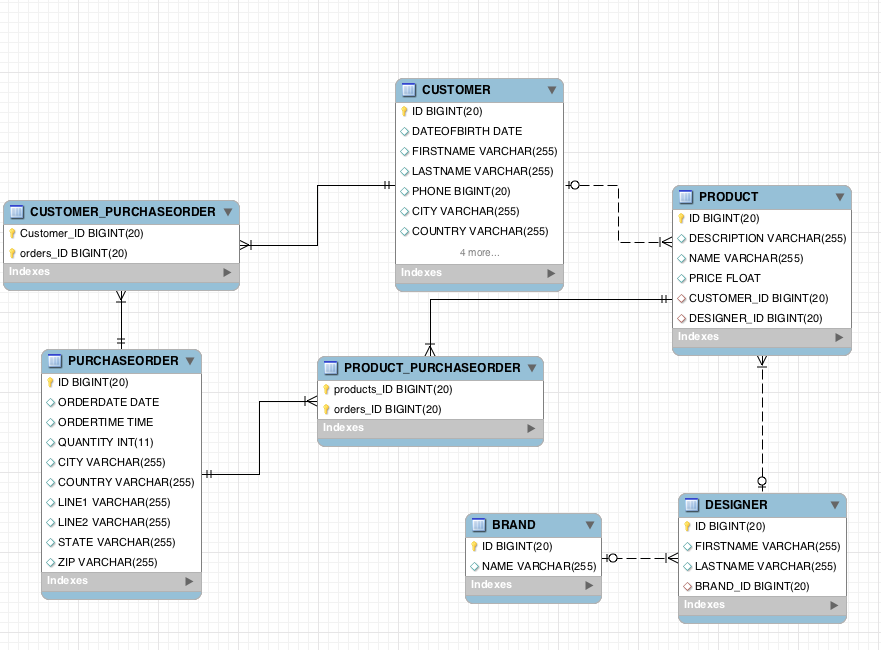
* When the project is run, it displays the result in the test window and creates the database tables in

**Screenshots:**

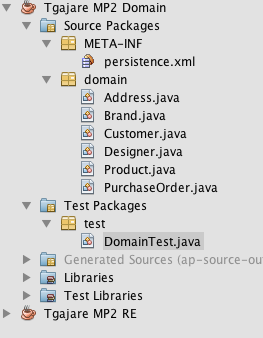
RE ERD Screenshot



Domain RE Screenshot



Project Structure



Test Result

