# EComm JPA Project Description

**Summer 20**

**Projects will be collected from UTD BOX on 7/23 @ 11 AM.**

# Introduction

This project is a continuation of the JDBC project and demonstrates an alternative method of persisting objects using Java Persistence API and Hibernate.

# Supporting YouTube Video

A video that describes the setup, testing, and export of the project can be found at: <https://youtu.be/Hf0F8c2cd8s>

# Deliverables

The version turned in on the due date is the final version. Only those portions that work correctly will receive credit as described in the following “Grading Criteria”. This policy is intended to encourage teams to finish well before the deadline and to verify the correct operation before final submission.

Each team will place the following materials in the UTD Box folder used to submit the JDBC project.

* A completed project report file: “JPA Project Evaluation – Team XX.docx” that includes contributing team member names and their Net-IDs, and the module they are responsible for completing. .
* A library jar file exported from your hibernateProjectForStudents project. This jar used by the instructor’s hibernateProjectTesting Eclipse project to execute the unit tests.
* The project’s source code. You can deliver the source code by the entire hibernateProjectForStudent project directory onto the UTD Box folder. Submission that do not include the entire project will not be graded and will be considered late when resubmitted.

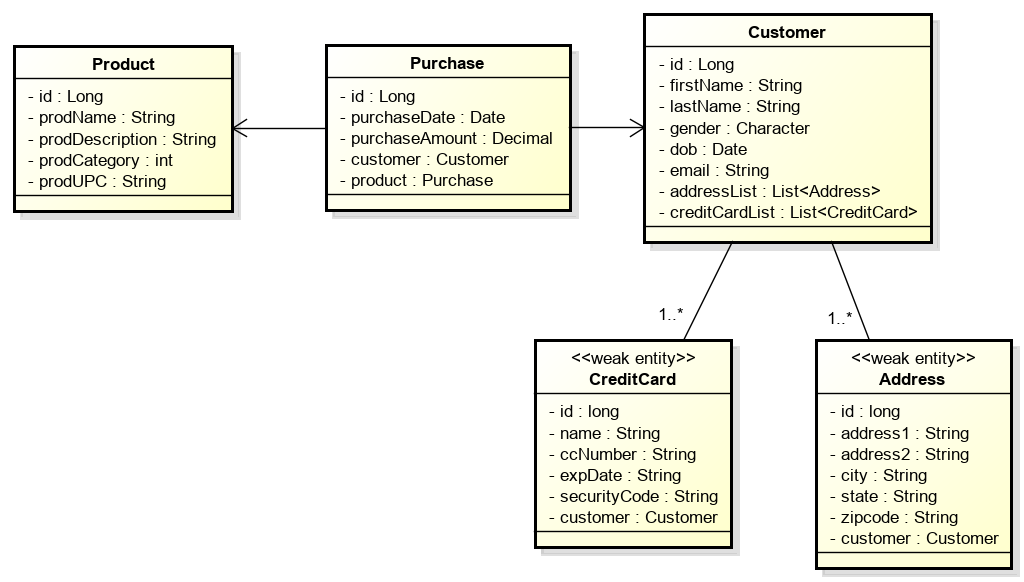
# Grading Criteria

Grades are awarded based on both team and individual contributions. The items marked in **green** will be evaluated on a team basis. The items marked in **red** describe the development of the three modules and will be evaluated on an individual bases.

1. **10 Points**. Providing a library JAR file from hibernateProjectForStudents that correctly compiles against the Eclipse project hibernateProjectTesting on the instructor’s personal Eclipse workspace. This will be accomplished by importing the submitted jar into the hibernateProjectTesting project build path.
2. **30 Points**. Executing the application PopulateTable.java and successfully populating the Customer, Address, CreditCard, Product, and Purchase tables created by Hibernate’s auto-generate option set in the persistence.xml configuration file.
3. **10 Points** is awarded on a team basis for the implementing all three modules.
4. **50 Points**. Executing services unit tests provided in the hibernateProjectTesting packages \*.unitTesting.service packages without error. See the section Service Modules.

# Entity Classes

The following classes represent the entities that will be persisted by the delivered services. Note that Java source files implementing these classes have been provided with the project materials. (hibernateProjectForStudent.zip)



**Note**: There are some difference with these classes compared to the JDBC project. For example, there 1-N associations between Customer and Address / CreditCard. Please review these classes and make no assumptions based on the JDBC project.

# Service Modules

In the interest of ensuring that each team member contributes to the development of the project, the implementation has been divided into three modules: Customer, Product, and Purchase. One or more team members is to be responsible for one of the three modules. The correct execution of all three modules will be evaluated on a team basis for 10 points.

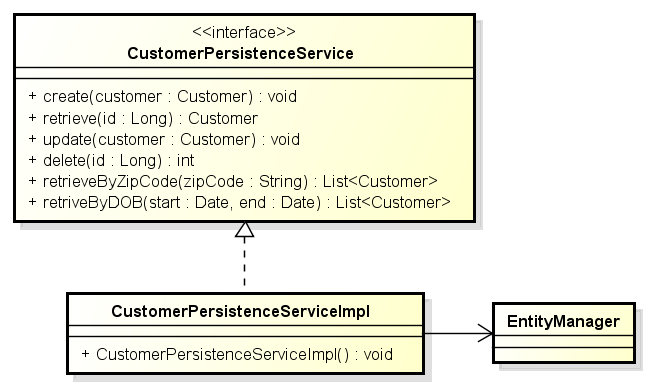
The following illustrates the Service Module interfaces that will be delivered with the project. Each services implementation maintains an EntityManager.

The delivered classes must implement these interfaces for the grading / testing to execute correctly on an individual basis. Note that Java interfaces for these services have been provided with the project materials.

## Customer Persistence Module

The CustomerPersistenceService is responsible for managing the persistence of Customer, Address, and CreditCard entities.

This service is to be implemented so that every operation on a Customer also manipulates its associated Address and CreditCard instances. For example, retrieving a Customer though its ID, also returns the Customer’s Address and CreditCard. Deleting a Customer also removes its Address and CreditCard. Etc.

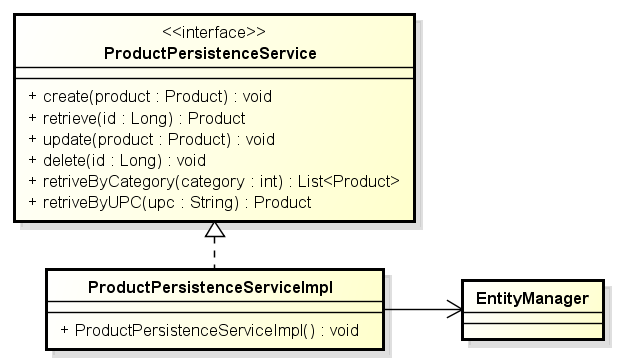


The CustomerPersistenceService provides the standard CRUD operation that allows the persistence and management of these instances. The service also provides operations that query for collections of Customer (Address & CreditCard) based on search criteria zip-code and date of birth ranges.

Notice that Address and CreditCard have not been provided persistence services in this design. This is because these are weak entities and attached to their Customer strong entity objects and all operations on a Customer are applied to their dependent Address & CreditCard objects.

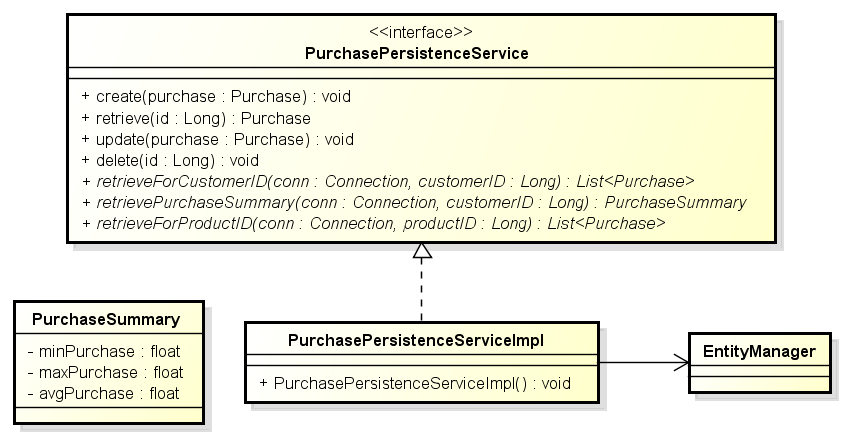
## Product Persistence Module

The ProductPersistenceService provides the standard CRUD methods for Products maintained in the ecommerce sites inventory. The service also provides operations that allow for the query for Products in certain product categories and retrieving Products by their UPC (Universal Product Code).



## Purchase Persistence Module

The PurchasePersistenceService provides the standard CRUD methods for Purchases made by Customers for Products. The services also allows for the retrieval of Purchases made by specific Customer, and for the retrieval of Purchases for specific Products. The service also provides a purchase summary of purchases made by specific customers.



Note: The method retrivePurchaseSummary() must use the equivalent aggregate SQL functions AVG, MIN, and MAX to be receive credit.

# Provided Materials

You have been provided two archived Eclipse projects (zip files) that will be imported into your Eclipse workspace. The last section of this document provides instruction on how the import is accomplished.

These projects are:

## hibernateProjectForStudents.zip

This project provides the basis for your Services implementations. This includes the Entity classes and Service interfaces as Java source. Your implementation will be written within this project, using these classes / definitions. The library jar file you create and submit for grading will be built from the work performed in this project.

**Do not change either the provided interfaces or the packaging structure. If changes are made, the unit tests will not compile (much less execute) and your team receives no credit for the work.**

## hibernateProjectTesting.zip

This project contains programs and unit tests that you will use to configure and test your service implementations.

**Except where noted in the source code, teams must not modify the code / contents of this project**. Changing the contents of the testing project will likely result in the unit tests in the instructor’s own unmodified testing project no longer working (no passing) and the loss of project points.

The hibernateProjectTesting project includes:

1. The application PopulateTables.java uses built-in JPA to both auto-generate the schema and populate the database generated by your team’s. This includes a number of CSV files containing the data used to populate the CUSTOMER, ADDRESS, CREDIT\_CARD, etc. tables.
2. A number of Junit test classes that will test the correctness of your CustomerPersistenceService, PurchasePersistenceService, and ProductPersistenceService implementations.
3. The class PersistenceManager.java that is used to provide access to a singleton instance of a EntityManager used by the unit test classes. **Note**: The text file persistence.xml contains the DataSource configuration and must be updated with the parameters needed to access your MySQL installation. You must not modify the URL’s schema (simple\_company).

## hibernateSampleCode2.zip

This project contains a working example of a Hibernate / JPA application that should provide a useful example of applying JPA to your simple company schema.

# Suggested Development Process

The following steps detail a suggested process for executing this project.

1. Import projects (Student, Testing, & Samples) into your Eclipse workspace.
2. Be sure that the Testing project’s Eclipse build path includes the Student project. You adjust the testing project’s build-path to include your hibernate project.
3. Annotate the five entity classes in the package cs4347.hibernateProject.ecomm.entity. These must be correctly annotated before the PopulateTable utility will work correctly in the following steps.
4. Using the MySQL workbench, manually create an empty schema “simple\_company”. This is a schema with no tables. Although JPA will auto-generate the tables, JPA will not create the schema.
5. Use the program PopulateTables.java (provided in the Test project) to both generate the entity tables and populate these tables from the data contained in the CSV files. Note that the entity classes must be annotated for this step to work.
6. Verify that the tables have been created in simple\_company and are populated with 100 rows each, although the Purchases table should contain 1000 rows.
7. Implement the interfaces CustomerPersistenceService, ProductPersistenceService, and PurchasePersistenceService. Note that the Student project has provided stubbed out implementation of the three interfaces.
8. Use the provided service unit test (see the package \*.unitTesting.service) to validate the correctness of your service implementations.
9. When all the tests work correctly, package ONLY the contents of the hibernateProjectForStudents project into a library jar file that will be submitted for grading. See the section “Exporting an Eclipse Project as a Library JAR File” at the end of this document.
10. Test that the library jar file works correctly by including the library jar in the hibernateProjectTesting project’s build path. NOTE: You will need to remove the project hibernateProjectForStudents from the testing project’s build path for the validation to work.

# Implementation Requirements

1. Note that your entity class ID annotations must include a GeneratedValue annotation with the annotation argument  
   **strategy = GenerationType.IDENTITY**. This will force the table to use an auto-increment key column. This restriction ensures that all of the tests that rely on known IDs will work correctly.
2. The implementation of PurchasePersistenceService.retrivePurchaseSummary() must use the equivalent aggregate JPQL / SQL functions AVG, MIN, and MAX to be receive credit.
3. Except where noted in the source code, teams must not modify the code / contents of the Eclipse project hibernateProjectTesting. Doing so will likely result in the loss of points when the unit tests in the instructor’s own unmodified project no longer work.
4. The application DataSource will be configured with the contents of the configuration file ‘persistence.xml’ in the hibernateProjectTesting project. A persistence.xml’ file has been provided in the project that attaches to the DBMS running on the local machine. You will need to modify the id (maybe) and password. Do not change the default schema name (simple\_company).