CSE 462 Project #1: Database Programming

Name: _____ Date: February 21, 2011

***** Submit electronically no later than 11:59pm on March 25th, 2011. *****

This project explores important issues in database programming using a modern object-oriented programming language: Object-Relational (OR) impedance mismatch, database-oriented application design, database API usage. At the end of this exercise, you will be able to:

1. OR impedance mismatch.
   (a) Explain some causes of the OR impedance mismatch.
   (b) Describe the main implications of the OR impedance mismatch.
   (c) Identify and compare approaches for coping with the OR impedance mismatch.
2. Database-oriented application design.
   (a) Isolate all database code from application business logic.
   (b) Identify and select approriate strategies for materializing data from the database.
   (c) Make both business logic and data access/modification code independently testable.
3. Database API usage.
   (a) Use JDBC PreparedStatement to query and modify the database.
   (b) Use JDBC ResultSet to retrieve query results.
   (c) Group database modifications into transactions.

You will work either individually or in a group of two. Your task is to implement the missing functionality in the reference project available for download from the course web site. You will submit your modified project as a zip file (`p1.zip`), which should also contain one text file (.txt, .pdf) with your answers to the questions below. Use `submit_cse462` to submit `p1.zip`.

A set of complementary slides for the project is also available from the course web site. It provides a brief introduction to database application design and programming and discusses the provided reference project. It is strongly recommended that you go through the slides in order to complete the project.

Task #1. [0 points] Your first task is to read and understand the reference project source code, including all provided comments. Start by reading the `Driver` class in the `edu.buffalo.cse.cse462` package and make sure you understand it. Then, create the necessary tables in your Oracle schema with the DDL statements provided as comments in the model interfaces (`edu.buffalo.cse.cse462.model` package). Run `Driver` and check that it produces an output similar to Listing 1. Finally, run `Driver` in debug mode and step through/into the code to see which lines and files are executed.

```
Username: demian
Password:
This is our customer:
[cid: 1] Smith, Joe was born on 1980-10-10 and lives in Buffalo-NY
=> Insert customer.
=> Make insertion permanent.
=> Find recently inserted customer.
[cid: 1] Smith, Joe was born on 1980-10-10 and lives in Buffalo-NY
=> Delete recently inserted customer.
=> Find recently deleted customer.
null
=> Abort recent work-- undoes the delete.
=> Find recently inserted customer.
[cid: 1] Smith, Joe was born on 1980-10-10 and lives in Buffalo-NY
=> Delete recently inserted customer.
=> This time, make deletion permanent.
=> Find recently deleted customer.
=> Should not be here.
null
```

Listing 1: Output of the `Driver` program.

Task #2. [200 points] The reference source has a number of stub methods (methods that simply return a trivial value) with the comment "*TODO: required implementation for CSE462 Project #1*." Your task is to implement the bodies of all such methods:

1. [10 points] Package `edu.buffalo.cse.cse462.model.impl`.
   - [10 points] Implement stub methods `equals` and `toString` in `ProductImpl`, `OrderImpl`, and `OrderEntryImpl`.
2. [150] Package `edu.buffalo.cse.cse462.internal.dao.oracle`.
   - [10 points] Implement both `update` stub methods in `OracleCustomerDAO`.
   - [30 points] Implement all stub methods in `OracleProductDAO`.
   - [40 points] Implement all stub methods in `OracleOrderEntryDAO`.
   - [70 points] Implement all stub methods in `OracleOrderDAO`.
3. [40 points] Package `edu.buffalo.cse.cse462`.
   - [5 points] Complete `testCustomer` by adding tests for the `OracleCustomerDAO` `update` methods.
   - [10 points] Implement `testProduct` by adding tests for all `OracleProductDAO` methods.
   - [10 points] Implement `testOrderEntry` by adding tests for the `OracleOrderEntryDAO` methods.
   - [15 points] Implement `testOrder` by adding tests for all `OracleOrderDAO` methods.

Task #3. [50 points] Provide short answers for the questions below.

1. [10 points] List *two significant advantages* of using JDBC code directly where data access or modification is necessary and *two significant advantages* of using an ORM layer. Justify your answer.
2. [20 points] Consider each of the fetching strategies discussed in the complementary slides. Briefly describe one hypothetical (yet realistic) scenario in which each strategy would be better than the other two. Do not use a scenario based on the reference project's data model.
3. [20 points] When designing a data access layer, it is not always clear what fetching strategy to implement for a given model object, as they normally depend on several factors which may not be available at design time. For instance, nature and size of the data, supported applications and their functionalities, complexity of data processing and business rules, etc. In some cases, you many need to create multiple DAO objects *for the same model object*, each implementing a different fetching strategy. Briefly describe one hypothetical (yet realistic) scenario in which lazy and eager loading are used for the same model object(s). It suffices to identify the model object(s) with dual loading strategy, specify the functionalties requiring each of the loading strategies, and justify why the loading strategies are adequate for the specified functionalities. Again, do not use a scenario based on the reference project's data model.