Shop Exercises

The following exercises assume you have already studied the Hibernate Shop application code and have got it working. Note than use case methods must not print anything to or read anything from any stream: if the exercise requires printing anything, a suitable value or object must be returned from the use case method (AFTER any Hibernate session or transaction has finished) and then appropriate printing can be carried out.

1. [10 minutes] Modify the method usecases.ShopService.PopulateDB to add a number of open and closed orders for a number of different customers to the database.

2. [30 minutes] Add a use case to get a list of books from the book catalogue. The console command line (as obtained from args[]) should specify 3 arguments The first is the command "get-books", the second (called "windowSize") an integer in the range 1 to 50, the third an integer (called "startPosition") an integer greater than or equal to 0. The usecase method should return a list of books from the catalogue. The books returned should be those that start at position startPosition and continue for windowSize books (or to the end of the list if that is fewer) in the list of all the books in the catalog if the list were ordered alphabetically by title. On return from the use case method, the list of books should be printed in a reasonably human-readable form.

3. [20 minutes] Add a use case to update the catalog price of a book. As usual, the parameters are specified by the console command line: The command should be "set-price", and it should take two arguments: the ISBN for the book whose price is to be changed and the new price. The use case method should return the book object whose price was changed and a suitable report should be printed out. If any error occurs, a suitable message should be printed.

4. [2 hours] Modify the system to keep track of recently viewed books. This will require adding an extra relationship between customers and books so that every Customer object has a List of the 5 most recently viewed books by that customer. Note that you will have to look up the online manual for Hibernate to learn how to implement Lists: they are slightly more complex than Sets as the order is important in Lists and hence require an extra position field to record which position in the List any particular element occurs in. Add the following use case to support and demonstrate this new functionality:

Add the use case view-book. This has the command "view-book" and takes, as parameters, the customers email address and the isbn number of the book to be viewed. A list of up to 5 books should be returned: the first one in the list should be the book requested, the remaining ones (up to a maximum of 4) should be the 4 most recently requested ones before this request. Note that no book should appear in the list more than once and that the act of requesting a book causes the list of most recently viewed books to be updated.
5. [3 hours] Modify the system to support book reviews, where a review is number of stars (0-5), and a single String containing the text of the review. This will require an extra class, `Review`. Modify the Book object to have a `Set of Reviews`. Each `Review` should have a `book` property, by which the `Book` it refers to can be found, and an `author` property, by which the authoring `Customer` can be found. You should also modify the `Customer` class to add a `nickName` property (and update the `populateDB` method accordingly), which will be used to identify Customers on review displays without revealing their proper name or email address. Modify the `populateDB` to insert a number of reviews into the database.

Add a use case to print book reviews for a book. The command name should be "get-reviews", and it should take an ISBN for the book you want the review for (and the email address for the customer if you have implemented the "recently viewed books" exercise above and want to incorporate that as well). The use case method should return a book object and, on return, the book details and the set of reviews for the book should be printed in a reasonably human-readable form, including the `nickName` of the author for each review.

Add a use case to add a review for a book: The command should be `add-review`. The remaining parameters should be the email address of the `Customer` who is the author of the review, the ISBN of the book the review is for, number of stars and the string text of the review. If a customer adds a second review for a book, it should replace his or her first review. No output is required other than an indication of success in the case that there is no errors, but appropriate error messages are required otherwise.