**Class Project 2: Record Store Application**

***Task 0: Start a new project. Call it <yourname>-RecordStore.***

***Task 1: Read the following problem description and requirements***

This is based on the record consignment store application from Lab 3.

Your job is to re-write this but instead of storing data in the application or in files, it should be stored in an embedded Derby database.

To remind you, here are the requirements.

You are the manager of a second hand record store. Your business model is that people – the consignors - bring in their unwanted records. You take the records, and sell them in your store. For every record sold, you and the consignor share the sale price.

A consigner can bring in any number of records. They may bring in more records at any time.

You tag every record with a unique number to keep track of them.

If a record doesn’t sell within 30 days, you notify the consignor to come and pick it up. If they don’t want to, you put it in your bargain basement for a discounted price.

When records go in the bargain basement, they are all priced at $1. Whatever price the record sells for, you give the consignor 40% of the sale price, and you keep 60%. You should keep track of what you owe each consigner. You should also be able to record money paid to each consigner.

Records that don’t sell from the bargain basement after a year are donated to charity. The consigner gets no payment.

You need a program to manage your consignors and your inventory.

Write a program which

* Lets you add a new consignor. A consignor has a name and phone number.
* When a consignor brings in a record(s) you should be able to add the records to your inventory. A record should be linked to its consignor in some way. A record has an artist, a title, and a selling price.
* When you sell a record, you should be able to store the sale date and the amount the record was sold for.
* You’ll need to store data about paying your consignors. How much money has a consignor made? How much have you paid out to them? Can your program store data about payments to consignors?
* A way of searching the records. Use this to search your catalog, and also to warn if you have too many copies of a certain record.
* Which records have been in the store for 30 days or more? Which consignors do you need to contact about picking up unsold records? How will you handle this?
* Has the ability to move a record to the bargain basement.
* Deleting a record from the bargain basement after a year – can you display a list of records which have been in the bargain basement more than a year?

You’ll probably need at least two tables, one for the records, and one for your customers. Maybe something like this? What data types do you think these would be? How will you store data about payments to consigners? Will you need another table?

Records

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Id | Title | Artist | Date Consigned | Original Price | Date sold | Bargain basement | Price sold for | Consigner ID |
|  |  |  |  |  |  |  |  |  |

Consigners

|  |  |  |  |
| --- | --- | --- | --- |
| id | name | Phone number | Money owed |
|  |  |  |  |

***Prototype Due Date***

By April 29, your code should be at least 70% complete. During class, you will trade projects with another student. Your job will be to review their code, make comments and give them feedback on their design. There is a code review worksheet in D2L. ***You need to complete this in class and check in with me in class to get credit for this part of the project.*** You will itemize features implemented and features not implemented, test their application and identify any errors that need fixing. The other student will do the same for your code.

You will need to upload the code review worksheet to D2L and also share a copy with the other student you are working with.

***To Submit***

Complete the code review by the end of the class on April 29.

When you have completed your code, zip your entire project directory. Upload it to the Class Project 2 Dropbox by the due date (which is probably May 4). Once you have uploaded your project, please validate with me that I can unzip, import and run your project.

***Grading***

*Code 40%* - your code should be modular, designed in an OO style, and you should consider re-use, flexibility, and adaptability. You should have implemented all the features required, and they should work when tested with valid and invalid inputs. Your code should be able to handle changes in requirements or specifications. Your code should be laid out and organized professionally, with descriptive variable names. Code should make an effort to deal with errors and unexpected events. You should use classes data structures appropriately.

*Database 40%* – Derby database created using appropriate structure, organization and data types. SQLExceptions and other likely errors handled correctly. Parameterized SQL statements used. Code should not crash when given invalid input.

*Completed Code Review 10% -* thorough code review for another students code with helpful feedback provided.

*Comments 10%* - Descriptive and useful Javadoc comments on all public classes and methods. Comments should also be present in the code, with a focus on the most complex methods and parts of code.