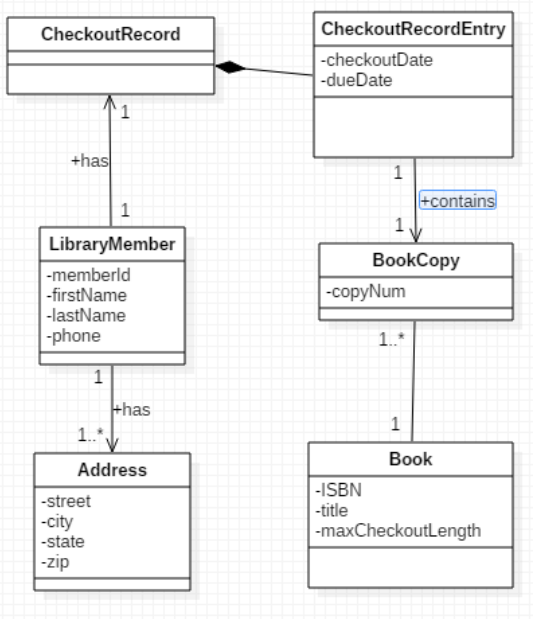
Practice Problems for Final Exam

These problems will help you review for the final exam. Some of the problems refer to smaller version of the Library System that we built in the course project. The classes have been coded for you and are in the helperclasses package. The TestData class provides data for you to test yoru code. The class diagram that models these classes and relationships is here:



Problem 1. For each of the queries below, do the following:

A. Create a stream pipleline that performs the query.
B. Turn your pipeline into a Lambda Library element
C. Replace the lambda expressions in your pipeline with local inner classes (these should not contain lambda expressions)
D. Use the TestData class to test your code.

Here are the three queries.

Query A: *Given a member's checkout record, determine whether some BookCopy in the record is overdue.* NOTE: A BookCopy is overdue if
(a) it is not available, and
(b) its due date is before now

Query B: *Given a BookCopy copy and a LibraryMember mem, return true if mem has ever checked out this copy*

Query C: *Given a list of all library members, return a list, in reverse sorted order (by first name), of the full names (first name + <space> + last name) of those library members who have never checked out a book*

Problem 2. For each of the lambda expressions below, do the following:

a. Assign an appropriate type (some functional interface)

b. Express it as a method expression

c. State the type of method expression you have used

d. Express it as an inner class

e. Evaluate the lambda, the method expression and the inner class inside an evaluator() method.

```
A.  () -> Math.random()
B.  (CheckoutRecord record) -> record.getCheckoutEntries()
C. (Long a, Long b) -> a.compareTo(b)
```

NOTE: In partC, you may not type the lambda as a BiFunction.

Problem 3. *Exception-handling within lambda pipelines.* High school students at Fairfield High School participate in an annual word contest. Students are given 15 minutes to type into a test computer as many words as they can think of that do not begin with one of the illegal letters. In this year's contest, the illegal letters are A, B, C, E, M, N, R, S, T.

The evaluator program for student submissions runs a method adjustWords which takes the list of words created by a participant, checks that all the words are legal, turns them into lower case words, and produces a new list with these modified words. The code for this is shown in the class WordGame in your prob3 package. However, it has been commented out because it is implemented as a lambda pipeline in which one of the lambda implementations throws an exception which is not supported by the interface type of the lambda.

Modify the implementation of this method so that the exception is handled in one of the standard ways. Then make sure that the main method executes test data as exepcted (you will need to uncomment the code in the main method).