CS401 MPP Final Exam

Maalouf, Assad

Name:_____ StudentId:_____

Problem 1 (10)	Problem 2 (10)	Problem 3 (10)	Problem 4 (10)	Problem 5 (10)	SCI Question (3)

Instructions

Five problems are given below, in addition to an SCI question. Choose four of these five problems to work on. Indicate which problems you are working on by marking them in some way (you could circle "Problem 3" if you are going to work on that one, for example). You *must* indicate clearly which problems you have chosen to do; if you do all five problems, there will be a penalty (and I will just grade the first four problems). Put code solutions in the corresponding packages. To answer the SCI question, create a text file SCI.txt in your project and type your answer into that file.

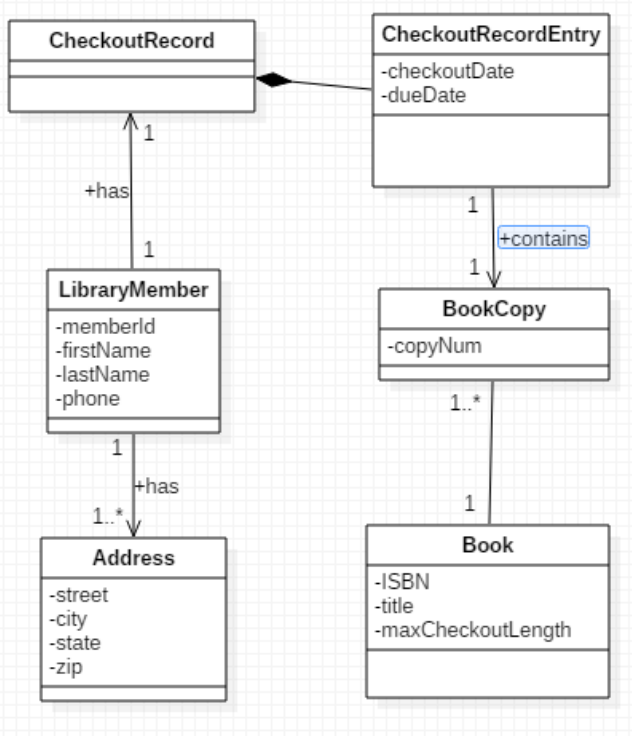
Problem 1. (10 points) Take a look at the package `prob1`, and the class `SampleProblem1`. There, a lambda expression is given in the class-level comments. Inside the class, the lambda is typed, represented as a method reference, and as a static nested class. There is also an `evaluator` method that evaluates all three expressions at values 2, 7.

Follow the same format and complete the code in the class file `Problem1` for the lambda

```
(Double x, Double y, Double z) -> x * y * z < x + y
```

Hint: To define the method reference, make use of a helper method.

Problem 2. (10 points) Use the code in the `helperclasses` package and the class diagram below to help you solve the following problem.



Write a stream pipeline inside the `main` method of the class `Problem2` (in package `prob2`) that does the following: It prints to the console the list of book titles –- in sorted order –- for which the book was checked out on June 27, 2015. The ordering of the book titles is as follows: First sort by the length of the title ascending order (smallest number of characters first), then by reverse of the title ("xza" comes before "axb"). Define your comparator using the declarative style described in class and make it available for testing using an auxiliary method. Use the data provided in the `TestData` class. Note that

a call for the list of `CheckoutRecordEntries` provided in that class has already been made for you in the `main` method – use the list provided there. *Note:* The date m/d/yyyy as a LocalDate is LocalDate.of(yyyy, m, d).

Problem 3. (10 points) The Library is having a contest. Library members submit their ids so they can participate. The library will take a special list of books and go through the `LibraryMember` participants and, one by one, checkout the next available book for the next member in the participant list. If, during the process, a member is found who, in this checkout process, has just checked out his 10th book (that is, he now has exactly 10 `CheckoutRecordEntries` in his `CheckoutRecord`), he wins the contest.

The code in `prob3.Problem3` checks a list of `LibraryMembers` for a possible winner of the contest, using a `List` of books obtained from `TestData`. However, the code does not compile because the `checkout` method on `LibaryMember` is capable of throwing a `LibrarySystemException` and so this checked exception needs to be handled in the middle of a call to the current stream's `filter` method.

Use the generic wrapper technique described in class to fix this exception-handling problem so that the code compiles properly.

Problem 4. (10 points) *(The reduce operation)* Implement the static method called `combine` in your class `Problem4` (in the package `prob4)` so that it transforms a `Stream<ArrayList<T>>` to an `ArrayList<T>` by finding the elements common in all the lists. Example: If these are the lists in the stream: ["Hello", "there"], ["goodbye", "there"] and ["there"], then the output of your `combine` method would be ["there"].

Your solution *must make use of the reduce method for Streams. Choose the right version of the reduce method and define a binary operator that given two ArrayList<T> returns their intersection in a new ArrayList<T>. Given list1 and list2 as two ArrayList<T> their intersection can be computed as follows*

```
(list1, list2) ->
list1.stream().filter(list2::contains).collect(Collectors.toCollection(ArrayL
ist<T>::new)
```

Test your solution by running the method `testCombine` that has been provided for you – this method is called from the `main` method in your class.

Problem 5. (10 points) (*Generic programming*) In the code below and in the AllPairs class in the prob5 package, an implementation of a method `allIncreasingPairs` is shown. This implementation accepts as input a list of Integers and returns all pairs (x,y) of Integers for which x, y belong to the input list and x < y. A test of this method is also provided in the Main class. Generalize this method so that it can accept as big a variety of list types as possible. In particular, your generic method should be able to form lists of pairs of Strings, Employees, and LocalDates – test methods for each of these types have also been provided (you need to uncomment them after you have written your code and complete the test method for Employees). You must use generic type variables as part of your approach to generalization.

```java
public static List<Pair<Integer, Integer>> allIncreasingPairs(List<Integer> list) {
    List<Pair<Integer, Integer>> returnVal = new ArrayList<>();
    for(int x: list) {
        for(int y: list) {
            if(x < y) {
                Pair<Integer, Integer> p = new Pair<>(x,y);
                returnVal.add(p);
            }
        }
    }
    return returnVal;
}
```

SCI (3 points) Write a short essay relating a point from the course to a principle from SCI. Richer content will receive more credit. Type your answer in the text file provided in the sci folder of your workspace.