CS401 MPP Final Exam

Name:_____ StudentId:_____

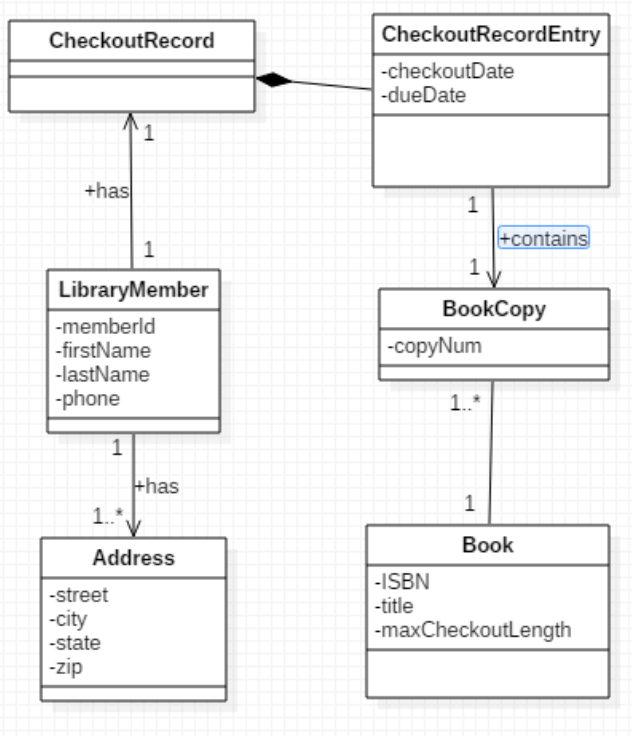
Problem 1 (10)	Problem 2 (10)	Problem 3 (10)	Problem 4 (10)	Problem 5 (10)	Problem 6 (10)	Problem 7 - SCI (3)

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) -> x * y < x + y
```

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: Print 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 (number of characters), then by alphabetical order.

Use the data provided in the `TestData` class – 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.

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 on behalf of 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.

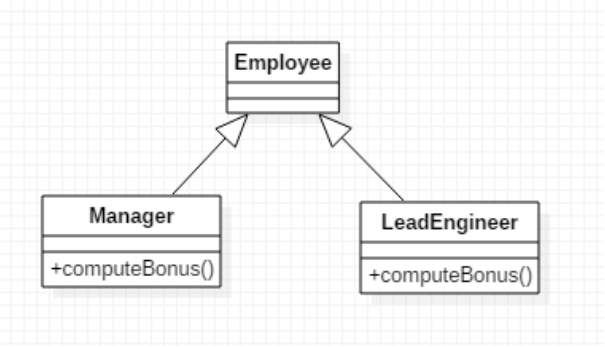
Use one of the standard techniques mentioned in class to fix this exception-handling problem so that the code compiles properly.

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 joining all the lists of the stream together. Example: If these are the lists in the stream: ["Hello", "there"], ["goodbye", "again"], then the output of your `combine` method would be ["Hello", "there", "goodbye", "again"].

Your solution *must make use of the reduce method for Streams.*

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)** In your `prob5` package, you will find three classes: `Employee`, `Manager`, `LeadEngineer`. Their relationships to each other are shown in the following class diagram:



The `computeBonus` methods in the classes `Manager` and `LeadEngineer` have identical implementations in both classes. Without changing the inheritance relationships shown here, use new features in Java 8 to modify the implementation of these classes so that there is no duplication of code.

Problem 6. (10 points) (*Generic programming*) In the code below, you see a method `conditionalRemove`, which removes from an `ArrayList` of `Strings` all those `Strings` that have length exactly equal to 5. Write the most general possible version of this method.

```
public class ConditionalRemove {
    static class StrLengthCondition implements Predicate<String> {
        public boolean test(String s) {
            return s.length() == 5;
        }
    }
    @SuppressWarnings("serial")
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<String>() {
            {
                add("Hello");add("Goodbye");
            }
        };
        System.out.println(conditionalRemove(list));
    }
    public static List<String> conditionalRemove(ArrayList<String> list) {
        StrLengthCondition cond = new StrLengthCondition();
        for(String s : list) {
            if(cond.test(s)) {
                list.remove(s);
            }
        }
        return list;
    }
}
```

Things to generalize:

1. From `String` to the most general type possible
2. From the `Predicate` type shown in the sample code to the most general type of `Predicate` (you may need to include a `Predicate` as a second argument to your `conditionalRemove` method)

In a `main` method, show that your `conditionalRemove` is capable of

A. Removing all `Strings` in an `ArrayList` of `Strings` in which the letter 'k' occurs
B. Removing all `Employees` in an `ArrayList` of Employees whose `salary` is < 60,000
C. Remove all `Integers` in an `ArrayList` of `Integers` which are greater than 100

Problem 7. (**SCI – 3 points**) Write one or two paragraphs relating a point from the course to a principle from SCI.