Com S 430
Spring 2015
Homework 1

Start by downloading hw1.zip from the **`examples/week2`** directory. All of the problems require you to make some modifications to existing code. Please leave the formatting alone for the parts of the code that you don't modify (that makes it easier to run a diff utility if we need to). You should submit a zip archive of the same files on Blackboard, including your modifications and comments.

Please start early. There are likely to be questions, ambiguities, errors, or other issues. Post questions on Piazza or talk with the staff.

1. Take a look at the class **`Trajectory`**. Modify it so that it is thread-safe, without changing the public API (including the questionable decision to use **`java.awt.Point`**). Add a comment to the class briefly explaining your changes.

2. Take a look at the class **`ImmutableTrajectory`**. Modify it so that it is actually immutable, without changing the public API. Add a comment to the class briefly explaining your changes.

3. Take a look at the class **`Mystery`**. Modify it so that it is thread-safe, without changing the public API. Note that all instances of **`Mystery`** must have distinct id numbers. Add a comment to the class briefly explaining your changes.

4. This problem refers to the sample code classes **`Client`**, **`SimpleServer`**, **`FakeDatabase`**, and **`NoSuchEntryException`**. First try running it. You'll need to start **`SimpleServer`** and then run **`Client`** (it is easier to see what's going on if you run the server in a command shell rather than in Eclipse). The idea is that **`SimpleServer`** hosts a very slow database. The client can submit an id number as a key and the server returns the corresponding record (a string), if the key is in the database. The class **`Client`** is a text-based UI in which you can enter an id to submit, or print the cache of all records previously queried. The query is satisfied from the local cache if possible, otherwise the client connects to the server to get the record.

You will probably notice that when the client has to submit a query to the server, it cannot respond to anything you type until the response is received. Very annoying.

First, read the code. If there is anything you don't understand, see me or post a question.

Here are your tasks:

a) Arrange that when the client has to submit a query to the server, it is done in a separate thread so that the UI remains responsive.

b) Then, eliminate the race conditions you have created in (a). It should continue to be the case that the local cache contains no duplicates.
c) Now the bottleneck is on the server side. Arrange that each query to the server is executed in a separate thread, so that the server can handle multiple queries concurrently.
d) Add a comment to the top of `Client` explaining all your changes.
e) *Optional*: fix it so that even if the user asks for the same id several times in a row, at most one request for that id is ever sent to the server.