Overview:

The contained is an implementation of the two phase commit protocol for distributed transactions. The terminology used is adapted from the wiki page on the topic, most notably the concept of the *coordinator* that issues requests to *cohorts*. There are classes that correspond to these.

Requirements:

The third-party libraries used are included in this submission. The version of Java with which it was compiled is:

```
java version "1.5.0_05"

Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_05-b05)

Java HotSpot(TM) Client VM (build 1.5.0_05-b05, mixed mode, sharing)
```

Instructions:

The submission comes with a **simple** demonstration script. Simply run `run.sh`. See the "Demonstration" section below for an explanation of what it's doing.

Contents:

The contents of this submission are:

`bin/`

Directory containing compiled classes

`data/`

Directory created by running the example script, contains one subdirectory per Sleepycat BDB environment (one per cohort on this server) (will not exist until you run `run.sh`

`lib/`

External, non JDK libraries used (see section below).

`README/`

Where various formats of this document live.

`run.sh/`

Demonstration script. Must be run from directory where the script is located.

`src/`

Java source code.

Libraries used:

Communication:

The Inter-Process Communication used was that included in the Hadoop library, a subproject of Nutch, the open source search engine. It was used because it provides a lightweight binary protocol for exchange of hand-serialized messages, and is a library with which I have a decent familiarity and experience of proven utility. Plus, major components of it were written by our department's own Mike Cafarella.

Persistence:

For stable, ACID disk storage, I used Sleepycat's Berkeley DB JavaEdition. It is a relatively small footprint, high-performance, embedded data storage system. This spares us from having to set up a separate storage server on each assembly member. As of the release of JE 3.0 (May 17, 2006) a new Persistence API is available, eliminating the need for low level serialization or transaction management, so I've taken this opportunity to familiarize myself with that API. The use of annotations in the code are in service of this new API.

Other than these two libraries, all compiled code included in this submission has been authored solely by me.

Demonstration:

The demonstration script `run.sh` sets the classpath to include the compiled source and the libraries. It then calls the main method of SimulatedDistributedStore with arguments to create 5 cohorts (listening on ports 10001-10005). It does this, and then creates a coordinator, which issues 1000 instructions, to modify 10 data elements.

I did not set up the coordinator with a user interface...though this would be trivial to do. Rather I have the demonstration execute 1000 instructions to the coordinator and allow the user to see the output of these instructions.