1. DESIGN DECIONS

DATABASE SCHEMA DESIGN

The data is stored in three tables

1) ACCOUNT- one row for each account reference
2) TRANSACTION- one row for each transaction
3) TRANSACTION_LEGS- one row for each transaction leg.

An integer field is used as PRIMARY KEY for all 3 tables. If the size of the tables could get really big then having String as primary key will create problems, hence using a integer id as primary key, although this `id` is not used for any other purpose for now.

MySQL database is used. I prefer MySQL as it is open source and free and performs reasonably well.

SPRING FRAMEWORK FOR DEPENDENCY INJECTION AND TRANSACTION MANAGEMENT

Spring has been used for Transactional Management and dependency injection. The code is much cleaner using spring and all the boiler plate code is managed internally by spring including transaction management.

APACHE_COMMON-LANGS-LIBRARY

Open Source library for Utility methods.

SPRING NamedParameterJdbcTemplate FOR PERSISTENCE

I have chosen JDBC with spring over Hibernate. Although using hibernate would have made the code a bit cleaner it terms of not having to write all the SQL queries. I preferred JDBC for the following reasons:

1) I need not pollute all the model classes with hibernate annotations or use XML.

2) Hibernate tends to get a bit complicated when the relationships across model gets complex. Unnecessary join tables will be created. For example in this case TransactionLeg class doesn't have a transaction reference, as we don't need it in the model but we need to store it in TRANSACTION_LEG table. To handle this with hibernate we will have to either add transaction reference in the model or create an additional JOIN table for the relationship.

3) JDBC is slightly faster compared to Hibernate in terms of speed

The only disadvantage I found was I had to manually write all the queries with JDBC although the boiler plate code is handled by Spring. This was a bit time consuming when compared to doing it with hibernate.

Error Handling

Unchecked Spring DataAccessException is wrapped into an unrecoverable InfrastructureException

MULTITHREADING AND TRANSACTION MANAGEMENT

Handling transactions run my multiple threads and transaction management is a crucial requirement for the problem. Spring declarative transaction is used for transaction management and rollback.

The account details updated my one thread in a transaction needs to be visible to all other threads. For this PESSIMISTIC LOCKING is employed. The account is read in `SELECT ... FOR UPDATE` mode to make sure the account updated by one thread is visible to other threads.

I preferred not to use any CONTAINER like TOMCAT for this. I felt there is no need for it. It might make sense to deploy it on Tomcat in future to handle and manage connections, pool size etc.

TESTING

Unit tests have been written for each of the individual components. Along with this MultiThreadedTransferTest is written to test transactions initiated by multiple threads using the Executor framework.

2. SOFTWARE VERSION

JDK-1.7_51

JRE-1.7<51>

SPRING- 3.2.0.RELEASE

MYSQL server-MySQL 5.6

3. SCRIPT LOCATION

1. The database schema details are under **bank\ddl\createtables.sql**
2. The data source details are under **\src\main\resources\database\dataSource.xml**

Please change the user name and password for MYSQL database here