Java Assignment 1

This assignment is about making a very simple key-value-database thread-safe.

You will need and should have got the following files:
- IAction.java
- Database.java
- BankAccount.java
- MoneyTransfer.java
- Program.java

The Database class implements a very simple key-value-database for integers.
There is a method readInteger(long objectId, byte propertyId) for reading a value from the database, and a method writeInteger(long objectId, byte propertyId, int value) for writing a value to the database. From the objectId and the propertyId a key to the underlying hash-map is created.
There is also a method transaction(IAction action) for packaging a number of database reads and writes into one transaction, which should be executed as an atomic operation.

The BankAccount class uses the Database to store the current balance of the account. Each created BankAccount instance will have a unique objectId. The objectId together with a propertyId for each property uniquely defines the key for each property of each object instance in the Database. In the BankAccount class there is only one property, which is defined by BALANCEID.

The MoneyTransfer class includes code for transferring an amount of money from one bank account to another bank account. It implements interface IAction and thus can be sent to the transaction(IAction action) method of the Database. It also implements interface Runnable and thus can be sent to a Thread constructor.

The Program class implements a simple test, which shows that the Database does not work when you have multiple concurrent threads. It only prints something out when an error occurs.

Task 1

Copy the above listed files, save them in a folder task1 and change the package declarations in the copied files from package task0 to package task1.

Your task is to modify the Database class to make it thread safe by using intrinsic locks. You should do it in the simplest possible way without any requirements regarding performance. However, you should make sure there is no risk for deadlock within the Database.

Task 2

Copy the above listed files, save them in a folder task2 and change the package declarations in the copied files from package task0 to package task2.

Your task is to modify the Database class to make it thread safe by using explicit locks. You should do it in the simplest possible way without any requirements regarding performance. However, you should make sure there is no risk for deadlock within the Database.

Task 3

Copy the above listed files, save them in a folder task3 and change the package declarations in the copied files from package task0 to package task3.

Your task is to modify the Database class to make it thread safe by using locks in such a way that the Database is performant when accessed from multiple concurrent threads. You should only lock the database entries, which are accessed during a read operation, write operation or a transaction. You should also make sure there is no risk for deadlock within the Database (under the assumption described below).

To be able to avoid deadlocks when executing concurrent transactions you need to know in advance which database entries that are accessed during a transaction. Therefore, you should extend the IAction interface to include a method getKeys(), which returns a list of keys (data type: List<long>).

You can assume that the code in an execute() method only accesses database entries whose keys are in the returned result of the corresponding getKeys() method. In other words, it is okay if there is a risk for deadlocks when getKeys() does not return a correct list.

Grading

To get the grade G (godkänt) you need to solve task 1 and task 2. To get the grade VG (väl godkänt) you need to solve task1, task2 and task3, and submit you solutions before 1:00pm March 11, 2014.

Submission

Your submission should be as a zip-file including all your java-files in the correct folder structure (task1, task2, task3) to iLearn2 (https://ilearn2.dsv.su.se/course/view.php?id=150).

God luck!