Full Stack Development
Lab Design 2

Architecture Design

1. Lab objectives

In this lab, we will review the specification provided by a BA on behalf of the client. Based on this, we will develop an initial high-level architecture.

2. The Specification

1. Bank accounts are kept in a main-frame system.
2. Bank account data has the following structure
    a. Account number – integer 4 digits
    b. Current balance – Integer representing the balance in USD
    c. Available balance – Integer >= 0 representing how much of the balance is available for use
    d. Account status – Integer code indicating
        i. 0 = normal status
        ii. 1 = account hold
        iii. 2 = closed
        iv. 3 = other
    e. Transaction limit – Amount that can be withdrawn in any single transaction
    f. Session limit – Amount that can be withdrawn cumulatively in a session.
3. Account objects are created and populated by querying data from the main-frame by account number
4. After each transaction that changes the balance, the new data must be written to the main-frame.
5. Operations allowed on the bank account are:
    a. Debits: increase the balance
    b. Credits: decrease the balance
    c. Querying the balance, available balance and amount left in the session limit

6. All amounts credited or debited must be > 0
7. Debits and Credits can not be done on accounts with a non-zero status
8. Accounts may not be put into overdraft as the result of a credit.

The following is the Java interface implemented by the main-frame database connector

BankDB.java

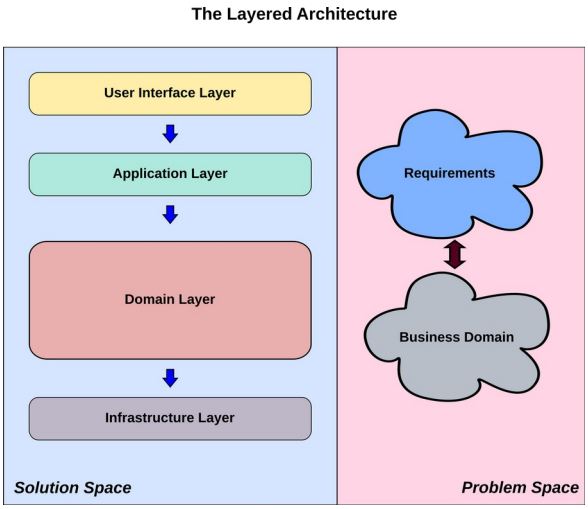
```java
package mainframe;

public interface BankDB {
    public int[] getAccount(int acctNumber);
    public boolean writeAccount(int[] data);

}
/*
 * data structure
 * 0 - account number
 * 1 - status
 * 2 - balance
 * 3 - available balance
 * 4 - transaction limit
 * 5 - session limit
*/
```

3. Putative Architecture

Remember the layered architecture from the first module.



What would be a possible architecture that be deployable as a microservice?

In terms of how we would organize the Java code, what sort of packages and classes would be appropriate to implement the functionality?

How well does this design conform to the SOLID principles as well as the design concepts of modularity, suppleness, high cohesion and low coupling?

End Lab