SFWRENG 3BB4 – Software Design III
Concurrent System Design

Assignment 3

Due date: Friday, November 25, 2016, 11:59pm (EDT)

Instructions: Please read the questions carefully. Solutions must be submitted in the Avenue Dropbox created for the assignment. Note that the deadline is strictly enforced. Avenue tracks the exact time that submissions are uploaded and late submissions may be rejected.

Problem Statement

Bank accounts are among the simplest and most common every-day-life kind of example of resources that are shared and used in a concurrent environment. Consider the Java implementation of the class `Account` provided in page 2.

Questions

Question 1 The Java implementation of the class `Account` may cause a program using it to reach a deadlock state. Identify a scenario in which this occurs. Prove that this scenario indeed leads to a deadlock state by resorting to an appropriate FSP specification.

Question 2 Propose a solution for the problem described in the previous question. Update both the FSP specification used to illustrate the problem and the Java implementation of the class `Account` so as to reflect this solution.

Question 3 For the FSP specification corresponding to your proposed solution for deadlock avoidance define: (i) 1 (one) relevant safety property; (ii) 1 (one) relevant progress property; (iii) 2 (two) relevant logical properties. Comment on the informal understanding of such properties and prove whether or not they hold true.

Final Remarks

Provide a short report detailing any assumptions and design decisions that you make, both about your Java implementation and your FSP specifications.

```
class Account {

        double balance;

        public Account( )
        {
                balance = 0;
        }

        void synchronized withdraw( double amount )
             throws InsufficientFundsException
        {
                if ( balance > amount )
                {
                        balance -= amount;
                }
                else
                {
                        throw new InsufficientFundsException( );
                }
        }

        void synchronized deposit( double amount )
        {
                        balance += amount;
        }

        void synchronized transferTo( Account to, double amount )
             throws InsufficientFundsException
        {
                if ( balance > amount )
                {
                        balance -= amount;
                        to.deposit( amount );
                }
                else
                {
                        throw new InsufficientFundsException( );
                }
        }
}
```