Oblig3: Blockchain Technology

Outputs, Inputs and Transactions

Last updated 15.10.2018 by Lars-Petter Helland

- You are supposed to work in groups of 1-3 students.
- You must register your group in one of the predefined Canvas-groups "Blockchain Oblig group x"
- You must submit your solution (a zipped Java project) no later than **Sunday 28. October**.

In Lab1, we explored how blocks are assembled, mined and added to an "immutable" chain of blocks. **Now**, we are going to look at how to **represent and record monetary transactions**. We will use Bitcoin as the basis for the assignment, but there will be some simplifications.

Keywords: *DSA Digital Signatures, KeyPair, PublicKey, PrivateKey, signatures, outputs, inputs, transactions, coinbase tx, UTXO-set, wallets.*

The task is to write a simple program in Java that demonstrates how the "mechanics" of Bitcoin-like monetary transactions work.

The purpose of doing this is to learn more in depth how things work that you can do by just reading. In addition, it is fun.

Supervision will be available in lab time **Friday 19.10.** and **26.10**, 0815+ - 1000.

All the details are explained as comments in the skeleton project (**download from Canvas to get started**).

Good luck!

Application.java

```
package no.hvl.dat159;

public class ApplicationSkeleton {

    public static void main(String[] args) throws Exception {
        /*
         * In this assignment, we are going to look at how to represent and record
         * monetary transactions. We will use Bitcoin as the basis for the assignment,
         * but there will be some simplifications.
         *
         * We are skipping the whole blockchain this time, and instead focus on the
         * transaction details, the UTXO and how money movements are represented.
         *
         * (If you want to, you can of course extend the assignment by collecting the
         * individual transactions into blocks, create a Merkle tree for the block
         * header, validate, mine and add the block to a blockchain.)
         */

        // 0. To get started, we need a few (single address) Wallets. Create 2 wallets.
        //    Think of one of them as the "miner" (the one collecting "block rewards").

        // 1. The first "block" (= round of transactions) contains only a coinbase
        //    transaction. Create a coinbase transaction that adds a certain
        //    amount to the "miner"'s address. Update the UTXO-set (add only).

        // 2. The second "block" contains two transactions, the mandatory coinbase
        //    transaction and a regular transaction. The regular transaction shall
        //    send ~20% of the money from the "miner"'s address to another address.
        //    Validate the regular transaction created by the "miner"'s wallet:
        //      - All the content must be valid (not null++)
        //      - All the inputs are unspent and belong to the sender
        //      - There are no repeating inputs
        //      - All the outputs must have a value > 0
        //      - The sum of inputs equals the sum of outputs (no fees)
        //      - The transaction is correctly signed by the sender
        //      - The transaction hash is correct
        //    Update the UTXO-set (both add and remove).

        // 3. Do the same once more.

        //    Now, the "miner"'s address should have two or more
        //    unspent outputs (depending on the strategy for choosing inputs) with a
        //    total of 2.6 * block reward, and the other address should have 0.4 ...

        // 4. Make a nice print-out of all that has happened, as well as the end status.
        //      For each of the "block"s (rounds), print
        //          "block" number
        //          the coinbase transaction
        //              hash, message
        //              output
        //          the regular transaction(s), if any
        //              hash
        //              inputs
        //              outputs
        //      End status: the set of unspent outputs
        //      End status: for each of the wallets, print
        //          wallet id, address, balance
    }
}
```

One possible output from running the program:

```
Block1:
CoinbaseTx(lmiCCBwX93nQk8BiV2fW5kvaFNIdNO8UPgxIwE9M8fY=)
   message=Genesis, output=Output [value=100, address=kFn1+kOrdmbP...=]

Block2:
CoinbaseTx(OFTi85+GpUfxtFSwI0dca+854PoOtXRQuZckkvxyYcA=)
   message=Hello, output=Output [value=100, address=kFn1+kOrdmbP...=]
Transaction(0Iz+qB65T3OCutB/dxyQBo2iORzFt2UkBFBnAlv2DHE=)
   inputs=
       Input [prevTxHash=lmiCCBwX93nQk8BiV2fW5kvaFNIdNO8UPgxIwE9M8fY=, prevOutputIndex=0]
   outputs=
       Output [value=20, address=f4WSHTd3/57vb7//EI8RF0L23J19f//rgozGAOFtnP0=]
       Output [value=80, address=kFn1+kOrdmbP/KHvrr9cD7wJ//QekWEuUQN29ayrI0w=]

Block3:
CoinbaseTx(HVkO3OLGTr1QzKsUKp3fxw/DVvheReRHbekCA1KuvBY=)
   message=Hello again, output=Output [value=100, address=kFn1+kOrdmbP...=]
Transaction(zzR7MngsWm/BzF9FQ8cKi67JwO2izO1Awj77i9ayPik=)
   inputs=
       Input [prevTxHash=OFTi85+GpUfxtFSwI0dca+854PoOtXRQuZckkvxyYcA=, prevOutputIndex=0]
       Input [prevTxHash=0Iz+qB65T3OCutB/dxyQBo2iORzFt2UkBFBnAlv2DHE=, prevOutputIndex=1]
   outputs=
       Output [value=20, address=f4WSHTd3/57vb7//EI8RF0L23J19f//rgozGAOFtnP0=]
       Output [value=160, address=kFn1+kOrdmbP/KHvrr9cD7wJ//QekWEuUQN29ayrI0w=]

UTXO:
Input [prevTxHash=0Iz+qB65T3OCutB/dxyQBo2iORzFt2UkBFBnAlv2DHE=, prevOutputIndex=0]
       --> Output [value=20, address=f4WSHTd3/57vb7//EI8RF0L23J19f//rgozGAOFtnP0=]
Input [prevTxHash=zzR7MngsWm/BzF9FQ8cKi67JwO2izO1Awj77i9ayPik=, prevOutputIndex=1]
       --> Output [value=160, address=kFn1+kOrdmbP/KHvrr9cD7wJ//QekWEuUQN29ayrI0w=]
Input [prevTxHash=HVkO3OLGTr1QzKsUKp3fxw/DVvheReRHbekCA1KuvBY=, prevOutputIndex=0]
       --> Output [value=100, address=kFn1+kOrdmbP/KHvrr9cD7wJ//QekWEuUQN29ayrI0w=]
Input [prevTxHash=zzR7MngsWm/BzF9FQ8cKi67JwO2izO1Awj77i9ayPik=, prevOutputIndex=0]
       --> Output [value=20, address=f4WSHTd3/57vb7//EI8RF0L23J19f//rgozGAOFtnP0=]

The miner's wallet:
Wallet [id=Miner's wallet, address=kFn1+kOrdmbP...=, balance=260]

My wallet:
Wallet [id=Lars-Petter's wallet, address=f4WSHTd3/57vb7...=, balance=40]
```