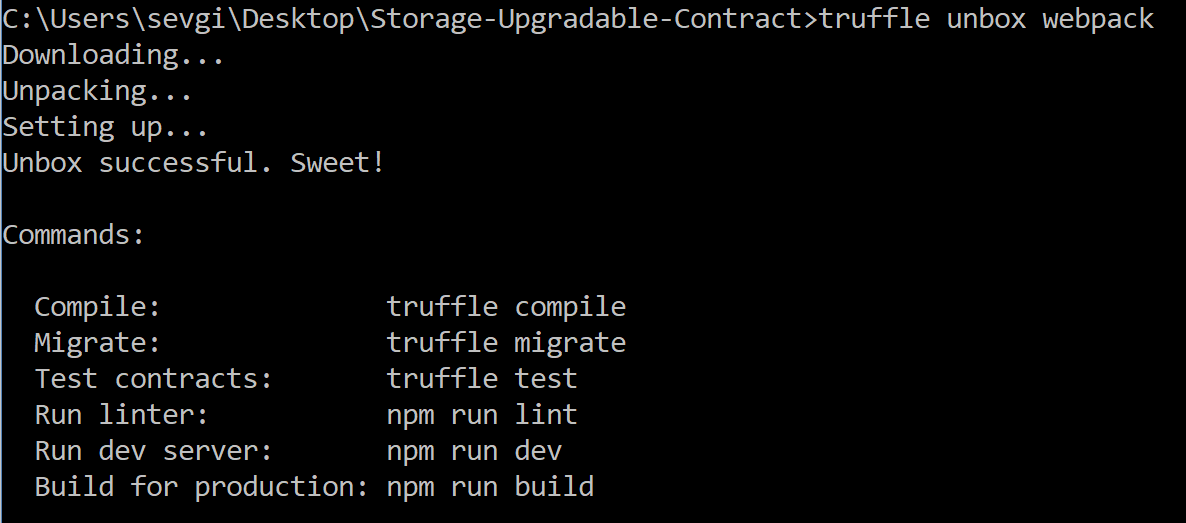
# Exercises: Upgradeable Contracts

This document describes the **exercise assignments** for the ["Blockchain Academy" course @ Software University](https://softuni.bg/courses/programming-fundamentals). In this lesson we learned the **basics of Solidity** programming language. The goal of this exercise is to get practical skills in writing simple smart contracts in Solidity, publishing and testing contracts in the Remix IDE.

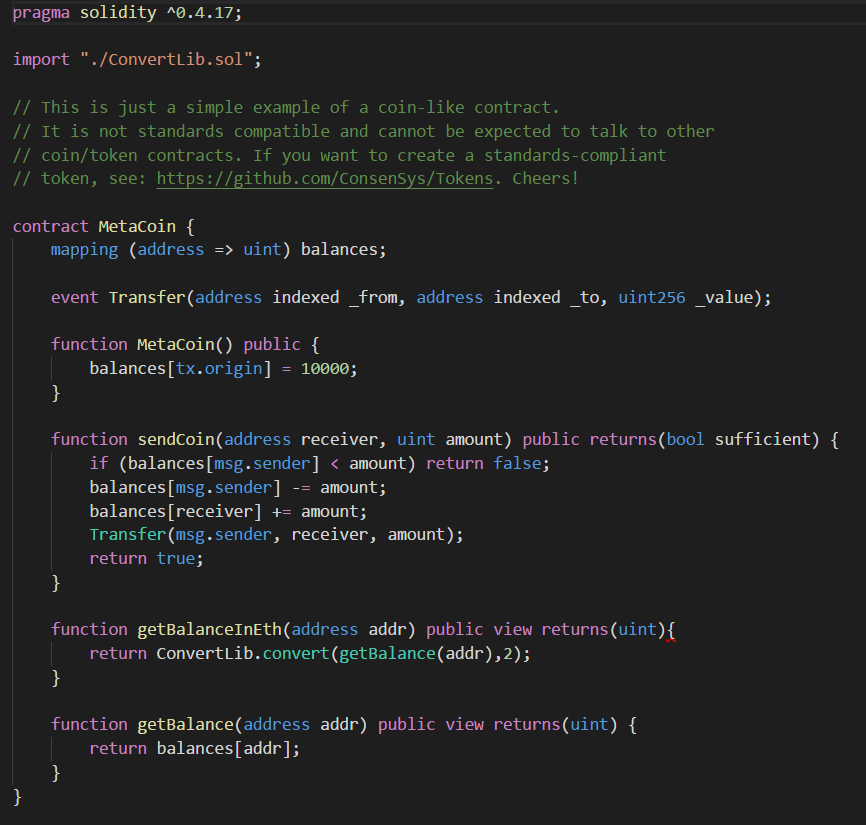
## Setup Project

1. Create new folder with the name **Storage-Upgradable-Contract**
2. Go into the directory and you can use **truffle unbox webpack** and truffle will download the truffle-webpack repository and install all dependencies



## Edit Contracts

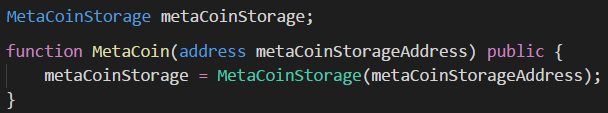
1. Edit the **MetaCoin.sol** contract which comes with truffle-webpack:



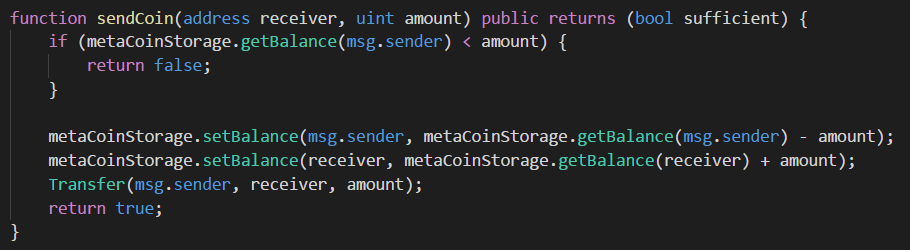
1. Let's identify the storage elements of the **MetaCoin** example. There is only one single element, which is **mapping(address => uint) balances**. We will get this out of the **MetaCoin** contract.
2. Now create a new contract called **MetaCoinStorage.sol** - contract with the modifiers for checking if the access is allowed and it is changing the addressSet mapping to the balances mapping. Here is the final contract:



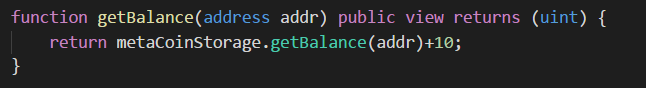
1. And now let's change the **MetaCoin** contract that it uses the **MetaCoinStorage** instead of directly accessing local variables.
2. First, we need to import the **MetaCoinStorage** contract.
3. Then remove the mapping balances and we need to assign the address of the **MetaCoinStorage** contract in the constructor:



1. In addition to that, we need to read/write to the **MetaCoinStorage** contract every time we make a change to the balance, instead of directly to the **MetaCoin** "balances" mapping:

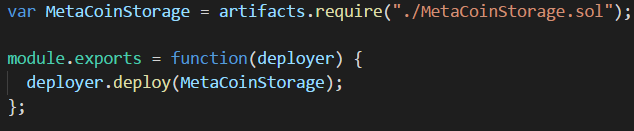


1. Now comes the tricky part. Let's say we found a bug in our **MetaCoin** contract. You might have noticed that I accidentally left a bug in the function getBalance function of the MetaCoin contract. Instead of outputting the balance, it will output the balance + 10. Here is the part, at the end it says "+10":

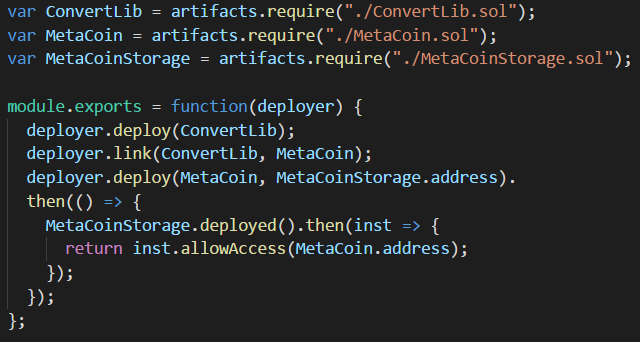


## Deploy and Test

1. Before fix the bug, let's deploy these contracts using the migrations of truffle.
2. We should expand the migrations and add a new one, because first, we have to deploy the **MetaCoinStorage** then we have to deploy the **MetaCoin** contract, last we have to allow the new MetaCoin contract to access the MetaCoinStorage contract.
3. Go to **2\_deploy\_Storage.js** file in migrations directory and change it to deploy the **MetaCoinStorage** contract:



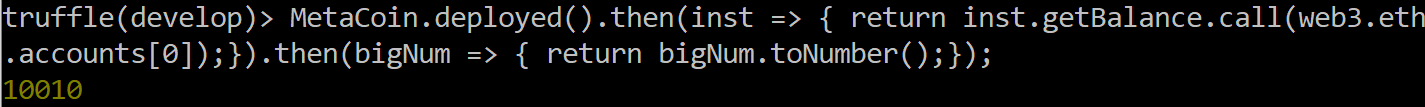
1. Then create new **3\_deploy\_MetaCoin.js** file in migrations directory and which deploys the **MetaCoin** contract and then allows the newly deployed address access to the **MetaCoinStorage** contract:



1. Now, let's start **testrpc** and then go to /**contracts** directory and start the developer console: **truffle** **develop** and migrate the contracts by the command: **migrate**
2. Now let's check directly on the command line how many coins we have in our first account. Let's run a command:

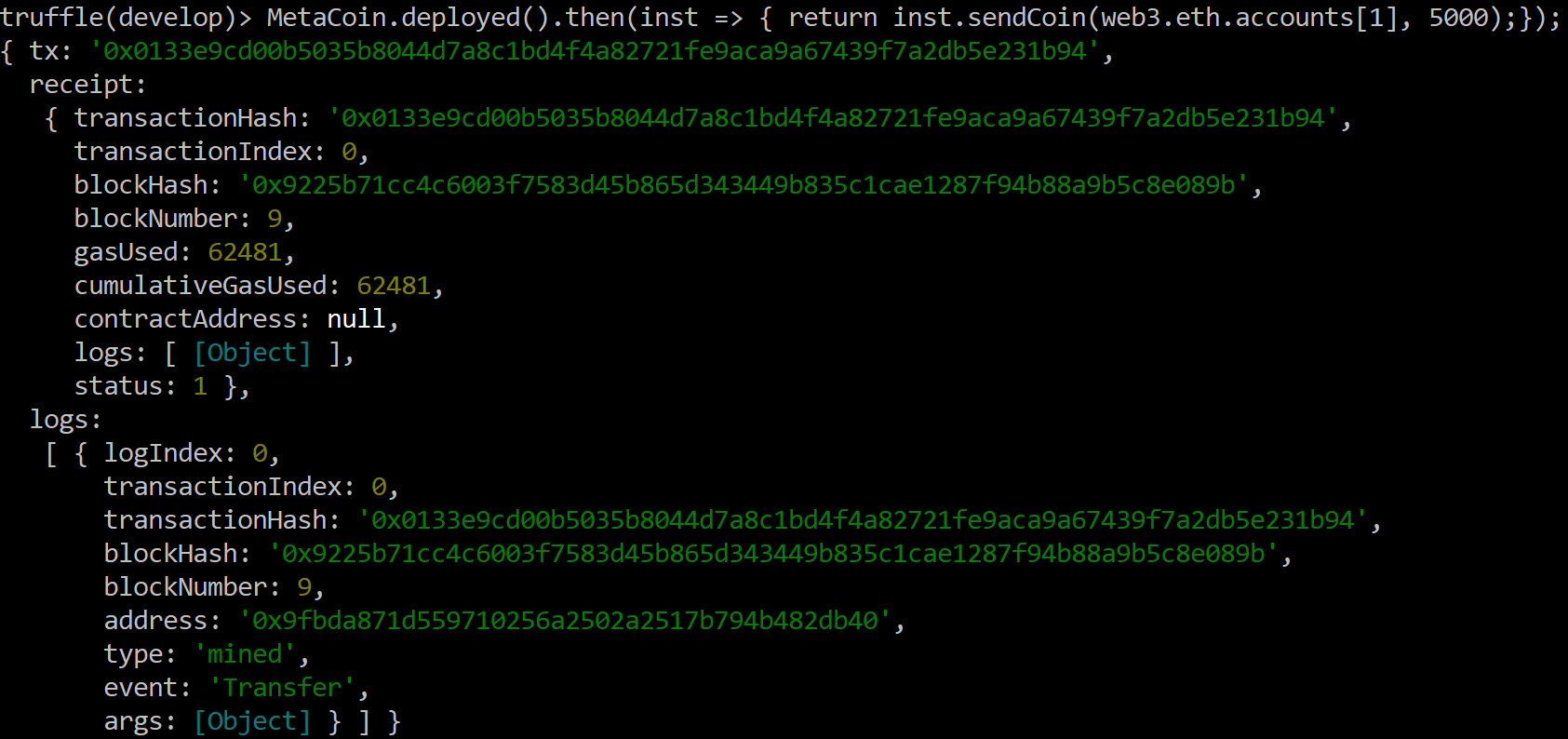
**MetaCoin.deployed().then(inst => { return inst.getBalance.call(web3.eth.accounts[0]);}).then(bigNum => { return bigNum.toNumber();});**

1. It should output that we have **10010** but we should have **10000.**



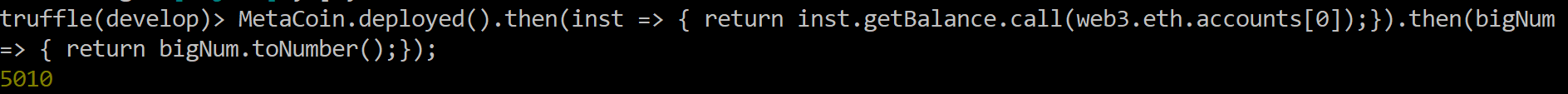
1. Still, let's assume we didn't find this bug immediately and let's send some coins around. Let's send 5000 coins from account[0] to account[1]:

**MetaCoin.deployed().then (inst => { return inst.sendCoin(web3.eth.accounts[1], 5000);});**



1. Let's check the balance again to make sure we just have **5000** coins left:

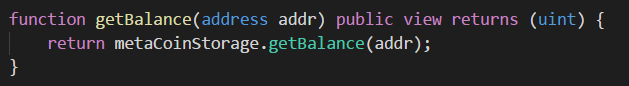
**MetaCoin.deployed().then(inst => { return inst.getBalance.call(web3.eth.accounts[0]);}).then(bigNum => { return bigNum.toNumber();});**

****

1. We have 5010 instead of 5000 coins.

## Fix the Bug and Test

1. Now let's fix the bug and re-deploy our contract. Our function from the MetaCoin contract with the bugfix looks like this:

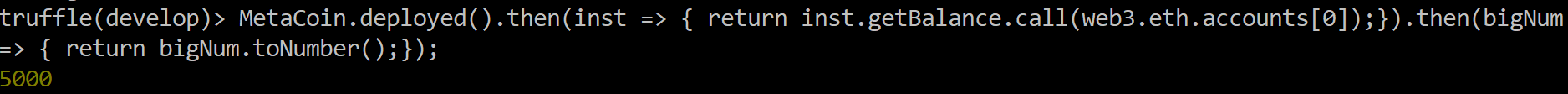


1. We should re-deploy the contract without changing the storage-layer. Truffle has us covered here, we can just run a single migration: **migrate -f** **3** will run migration #3



1. As you can see, it compiles everything, but migrates only **3\_deploy\_MetaCoin.js**. Now, let's check our balance again. It should be 5000 now:

**MetaCoin.deployed().then(inst => { return inst.getBalance.call(web3.eth.accounts[0]);}).then(bigNum => { return bigNum.toNumber();});**



1. This way, by having the storage in one contract and the logic in another one, it is easy to upgrade the logic part.

# What to Submit?

Create a **zip file** (e.g. your-username-upgradable-contracts.zip) holding the screenshots with your results.

Submit your **zip** file as **homework** at the course Web site.