# Exercises: Create Ethereum Wallet with C# and Nethereum

In this Exercise we are going to be learning how to create **Ethereum** **wallet** with **C#, .NET Core**, and **Nethereum** libraries. During the exercise you will install several **NuGet** packages, write the source code for different functionalities and in the end, you will **send** and **receive** ether coins with your wallet.

# Download the code template from our Resources.

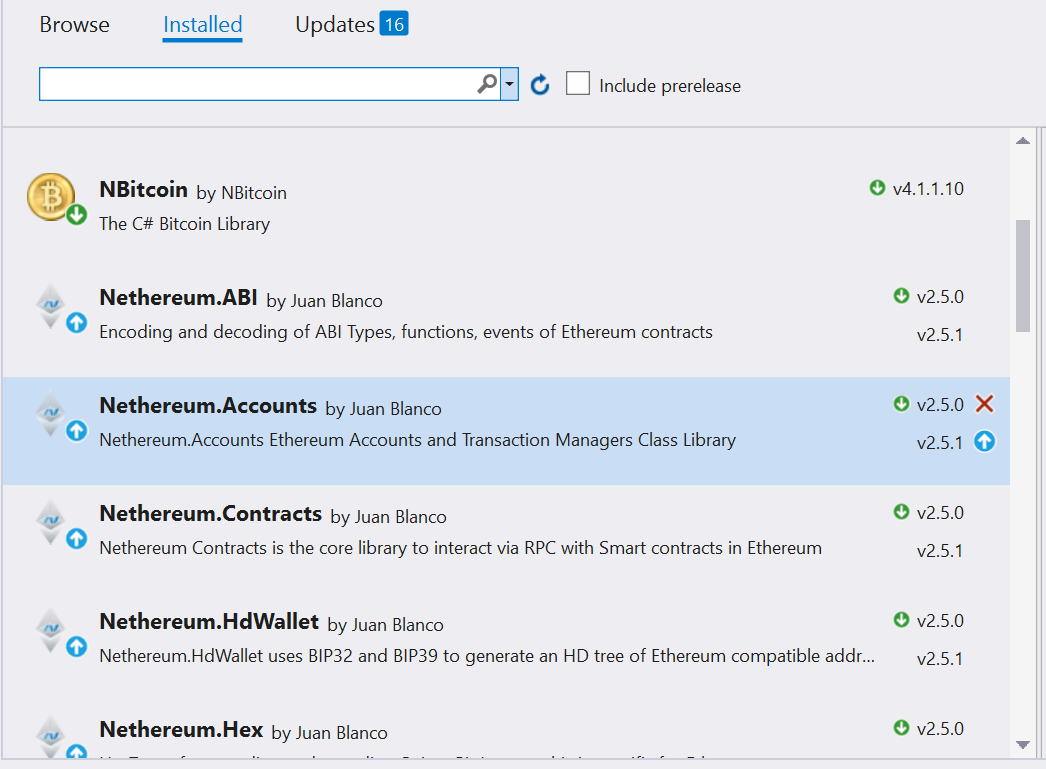
Extract the files from the provided Resources and you should have something like this:



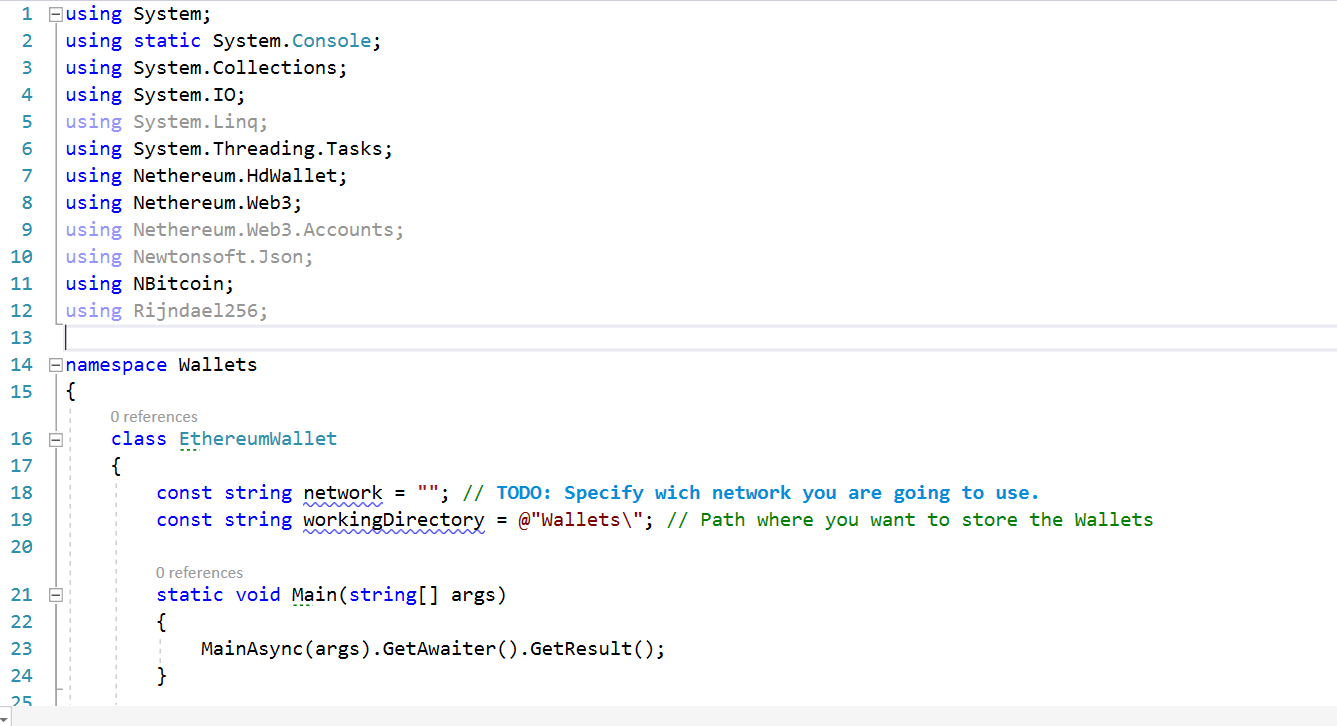
# Open the project.

Make sure that you have all necessary **NuGet** packages.

1. **Nethereum.KeyStore** by Juan Blanco
2. **Nethereum.HdWallet** by Juan Blanco
3. **Nethereum.Web3** by Juan Blanco
4. **Rijndael256.Core** by 2Toad, LLC

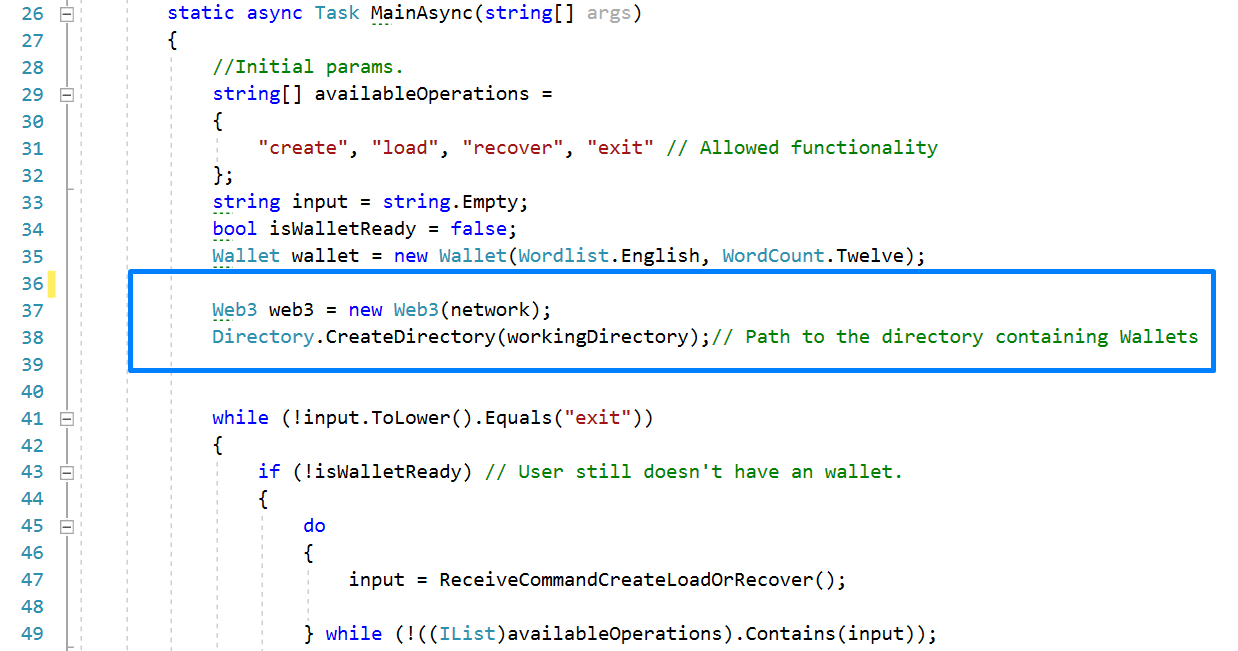


# Now if you check you should have Code template with many TODO’s for you to code.

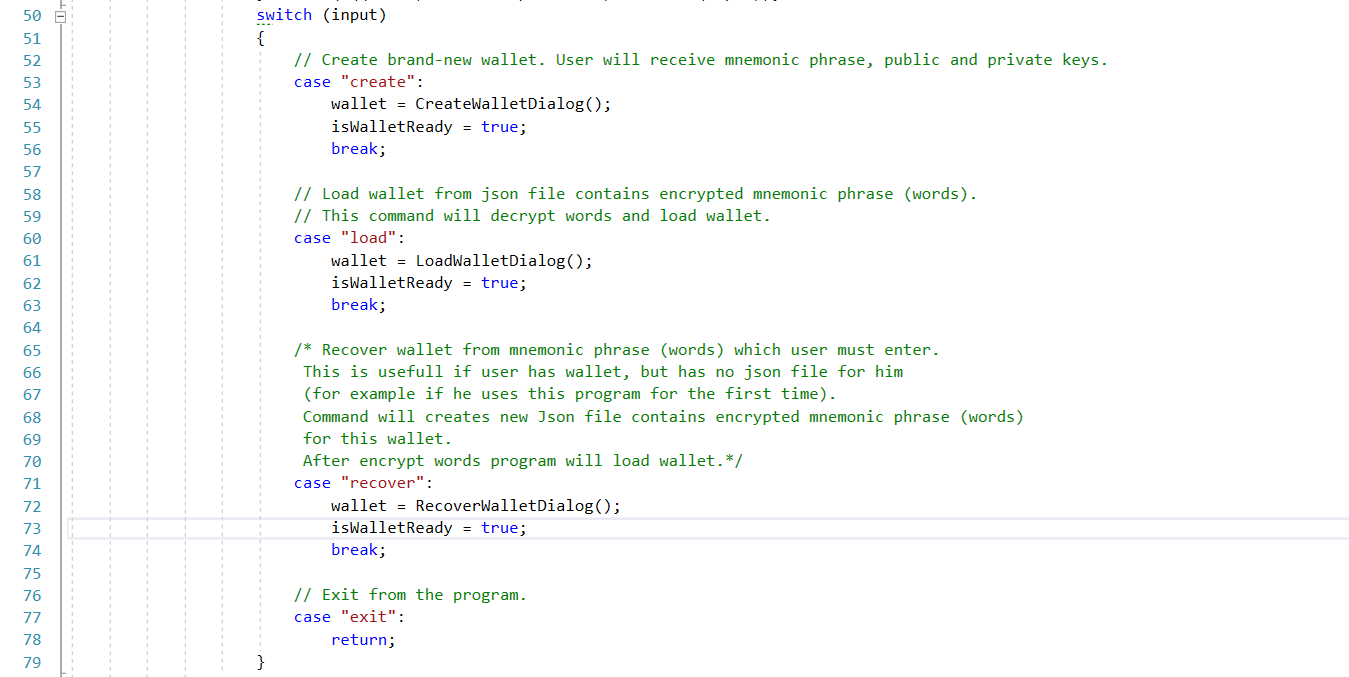


Here you only need to fill the missing information in the **constant** string “**network**” which we will use in the exercise. In this case it is **ropsten** testnet with **infura.io**. We will use another **constant** for the **working directory**.  
The **main** method will only call the **async** method **MainAsync()**

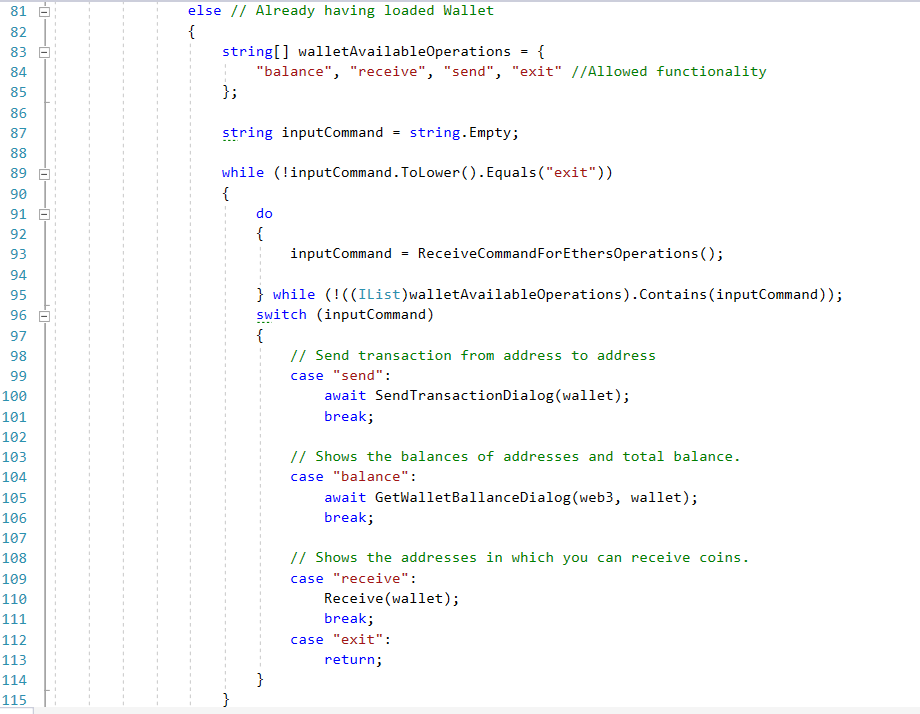
1. The main logic is in the **MainAsync** function. Here we will **call the methods** serving for the wallets needs.   
   First we create variables needed and create the working directory if necessary.



1. Then begin the long **while cycle** whit the nested while cycle inside. Wallet work in **two stages**.   
   First we **choose the working wallet**. The options are:
   1. **create** new wallet and save it to **json file**
   2. **load** existing wallet **from file** and
   3. **recover** existing wallet from **mnemonic phrases** and **save it to new json file**
   4. **exit** from the program

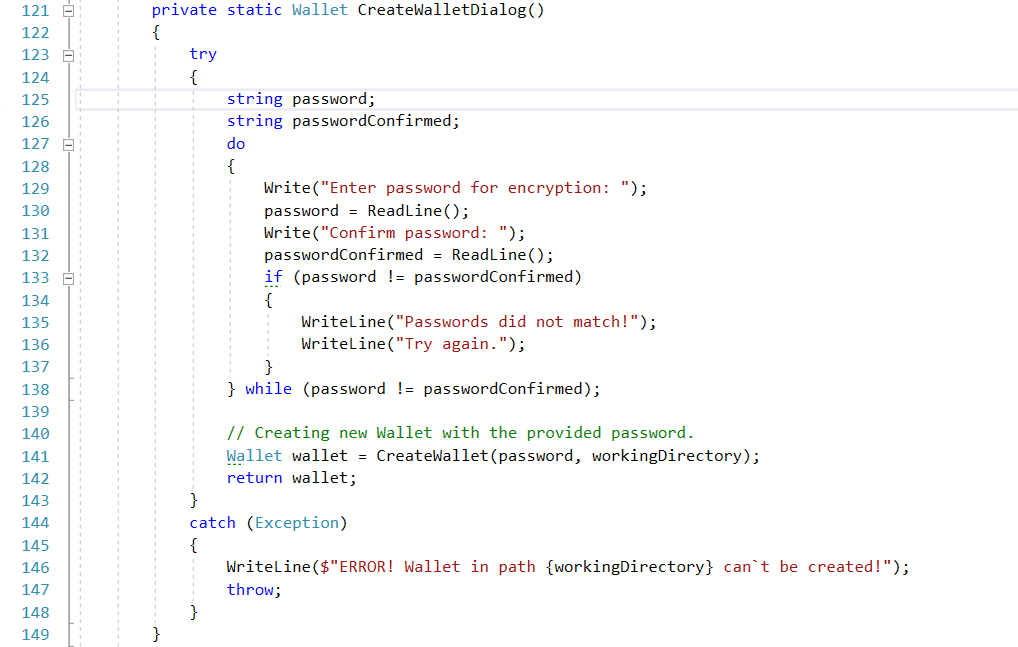


1. **Cycle** continue with the second stage. There we can:
2. **receive** addresses for receiving coins
3. check **balances**
4. **sending** coins
5. **exit** from the program

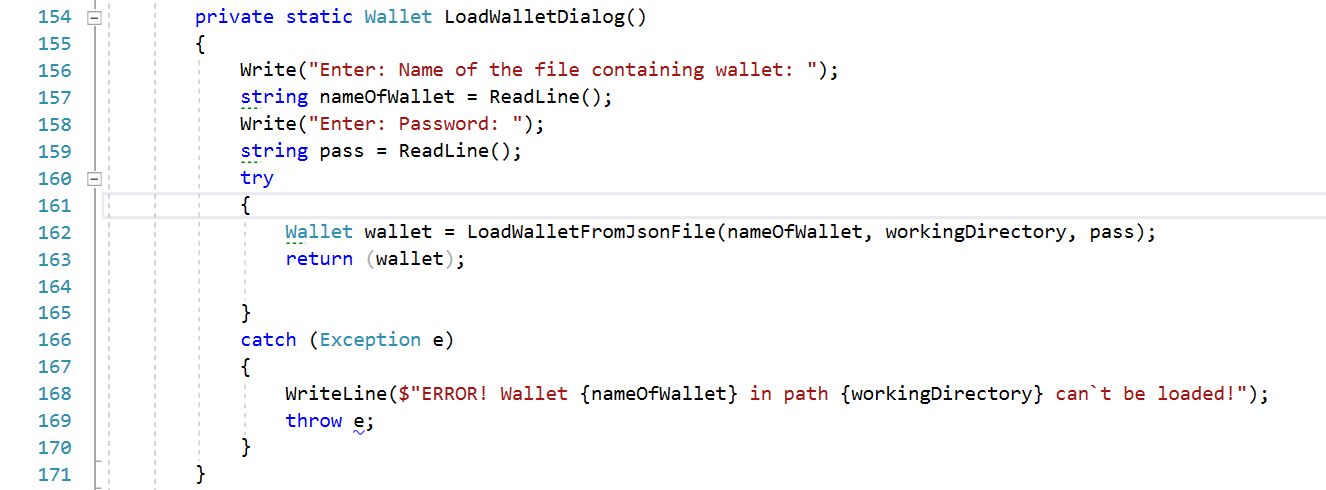


**We have created few Methods for the Dialogs that are needed so you don’t need to code them. But pleaes check them out and see how they work.**

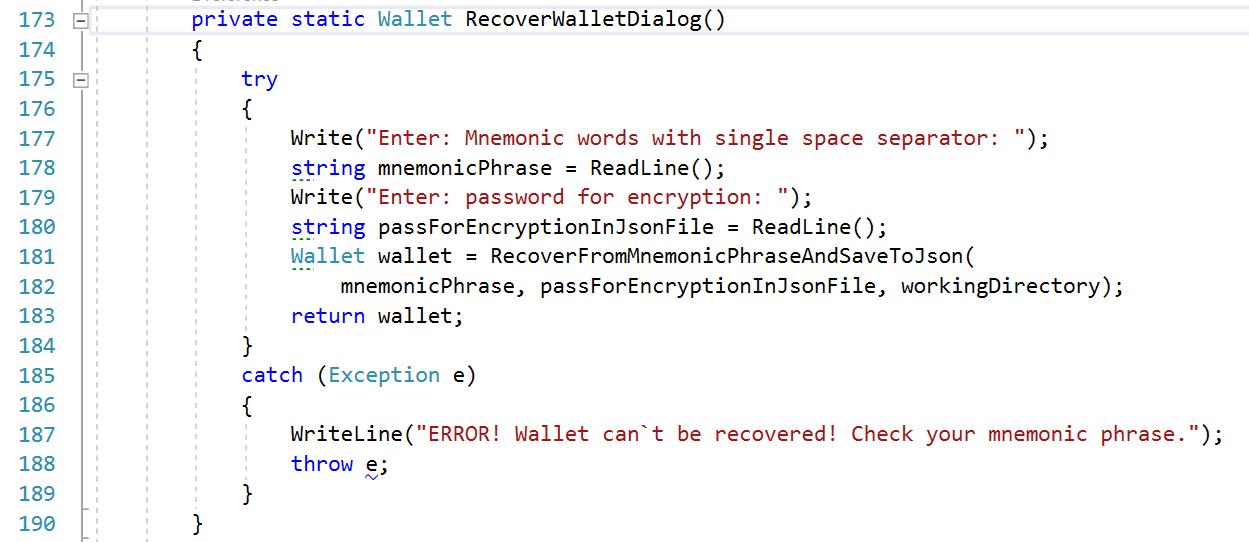
1. We have an **CreateDialog** Method that display’s the dialog needed for creation of Wallet



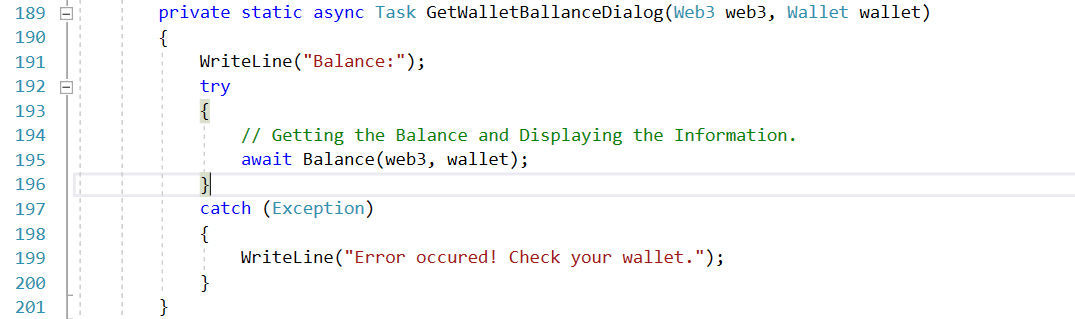
1. Here is the LoadWalletDialog Method that displays the dialog needed for Loading an Wallet



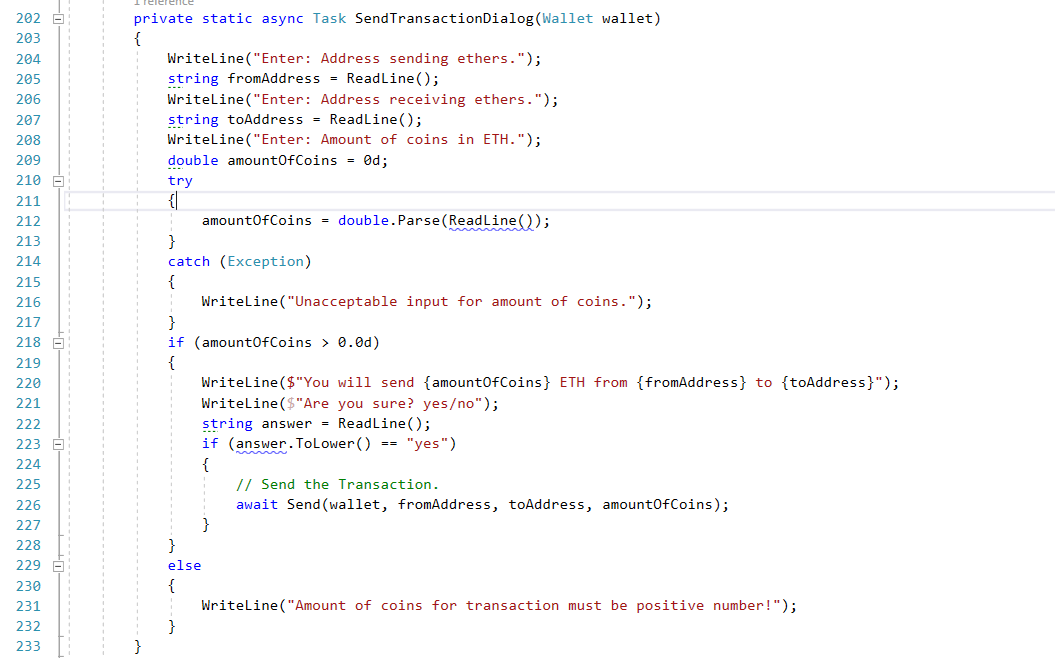
1. Here is the **RecoverWalletDialog** Method that displays the dialog needed for Recovering an Wallet



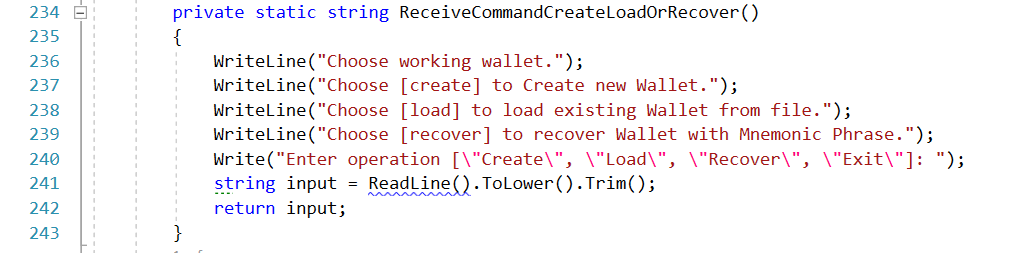
1. Here is the **GetWalletBallanceDialog** that displays the dialog needed for Showing the **Balance** of an Wallet



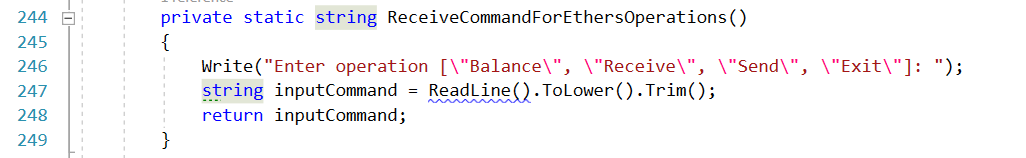
1. Here is the **SendTransactionDialog** that displays the dialog needed for **Sending** a **Transaction**



1. Here is the **ReceiveCommandCreateLoadOrRecover** that displays the available commands to the user.



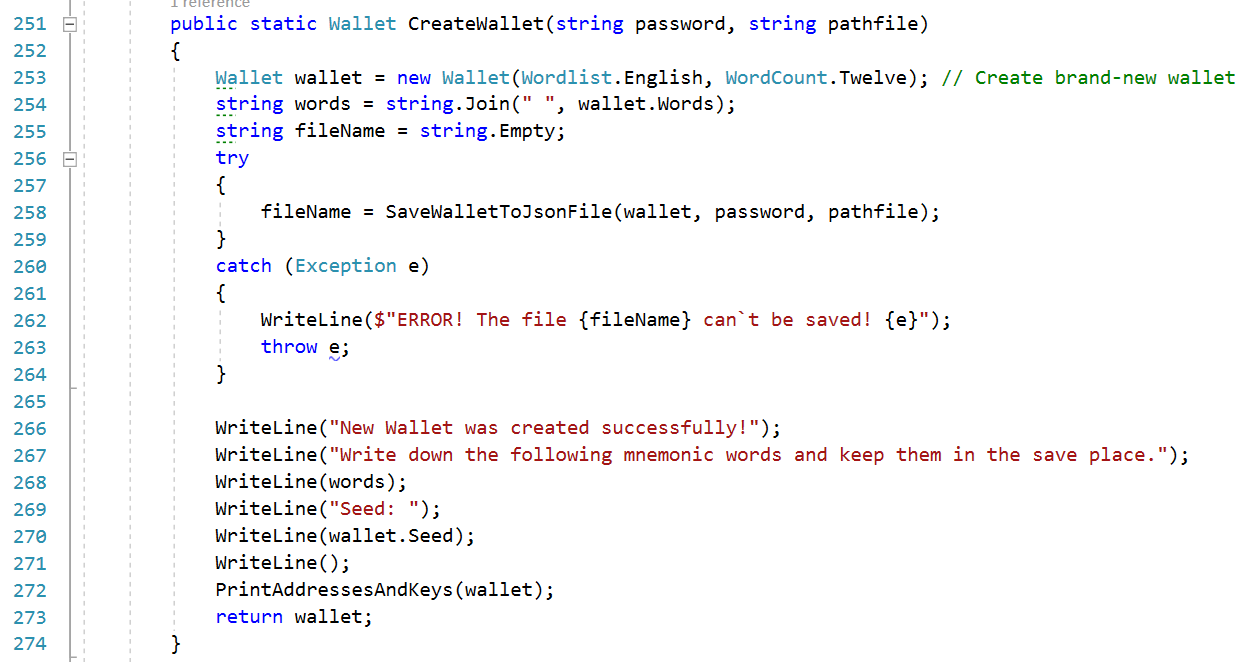
1. Here is the **ReceiveCommandForEthersOperations** that displays the available commands to the user.



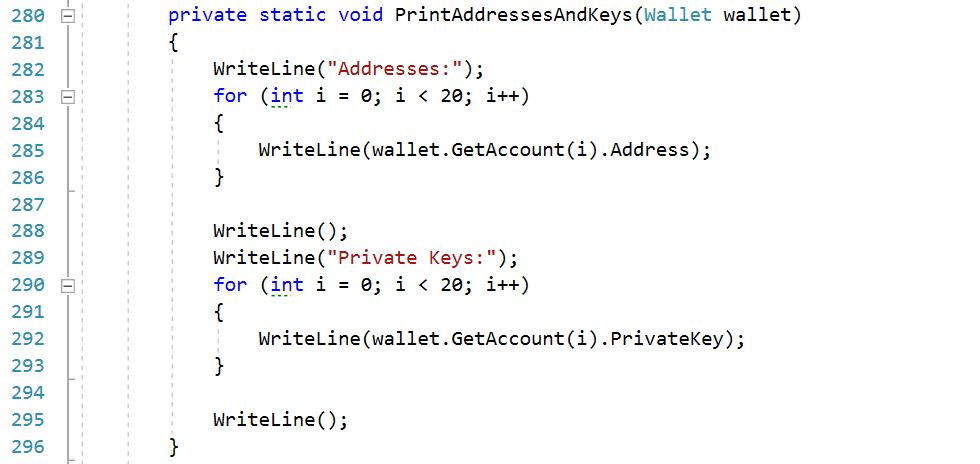
# Start implementing TODO’s

Now it is time to actually begin coding something. We have many TODO’s that we need to implement so the previous code can actually work.

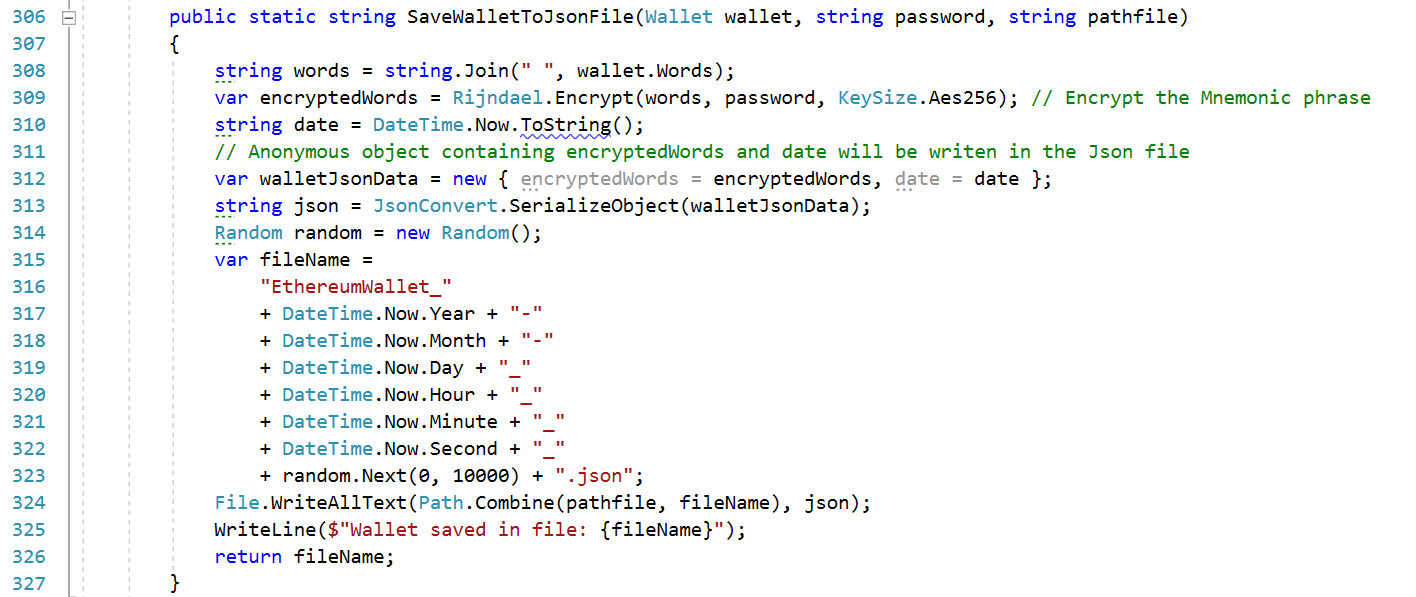
1. First we will write the **CreateWallet** method.  
   The **CreateWallet** method receive **password** and **path** where the json file containing wallet info will be saved. The method creates new wallet with 20 addresses and call **SaveWalletToJsonFile** method, then shows wallet`s mnemonic phrase (words), seed, addresses and private keys. Method **return** created wallet.



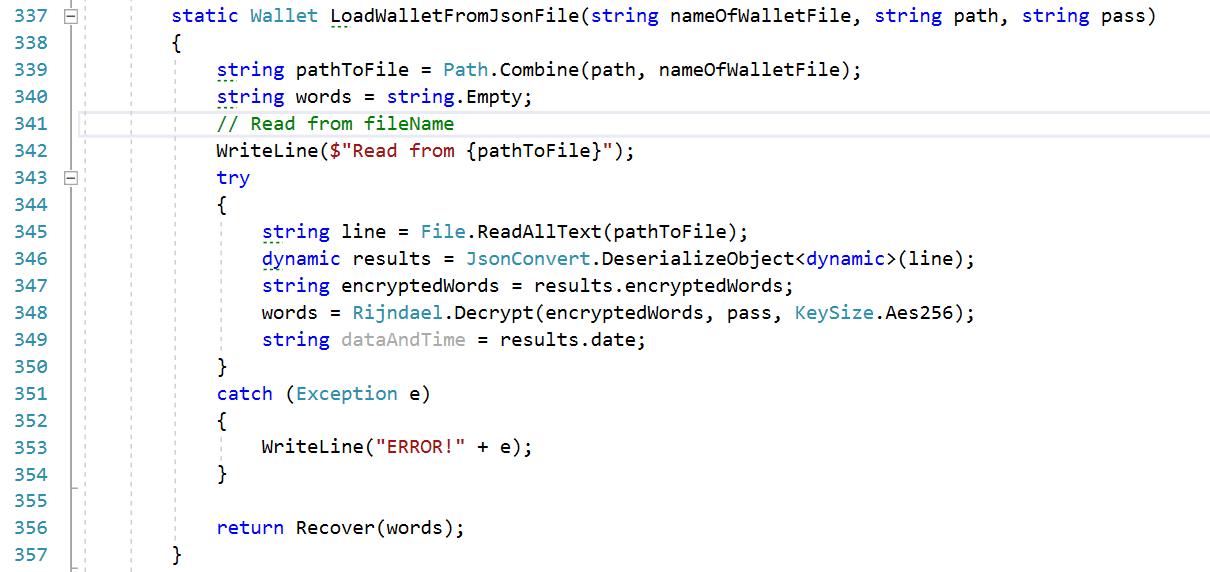
1. Printing of the addresses and private keys is separated in method **PrintAddressesAndKeys**. Method simple **receives** **wallet** and **print** his **addresses** and **private keys**.

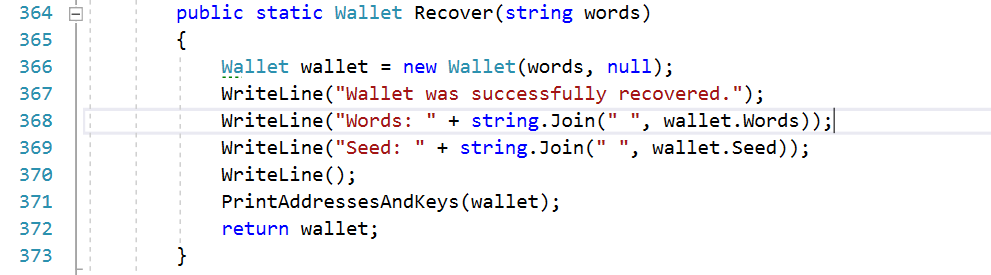


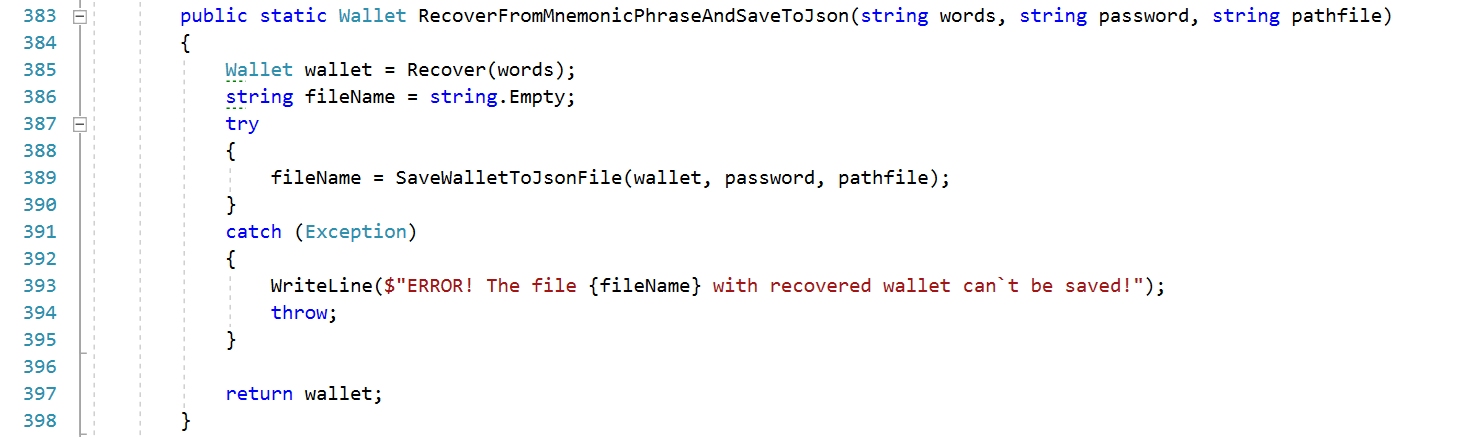
1. The next method is **SaveWalletToJsonFile**. This function **receives** **wallet**, **password** for encryption of the mnemonic words and **path** where the json file containing encrypted mnemonic phrase and date will be saved. For this example, we will use **Rijndael** library and **Aes256** algorithm. For the work with json we will use **Newtonsoft.Json** library. The program will generate automatically **file name** by concatenating string “EthereumWallet\_” and time variables from year to seconds, in the end of the filename we will put the random number in diapason (0,10000). It will guarantee that the file name will be unique. Method **return** filename.



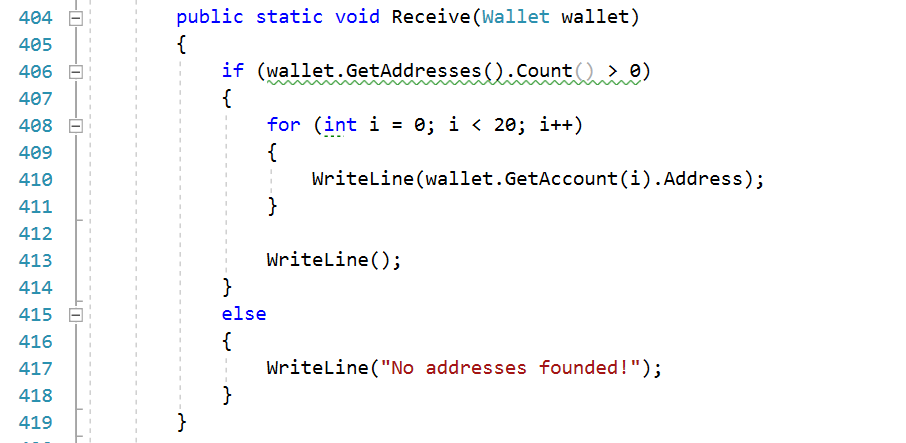
1. Now we will write the **LoadWalletFromJsonFile** method. Method receives parameters: **file name**, **path to file** and **password** to decrypt the mnemonic words from json. Function read the file content, load the encrypted mnemonic words from json and then decrypt it with password. Then this method calls **Recover** function with mnemonic phrase as parameter and **return** recovered wallet.



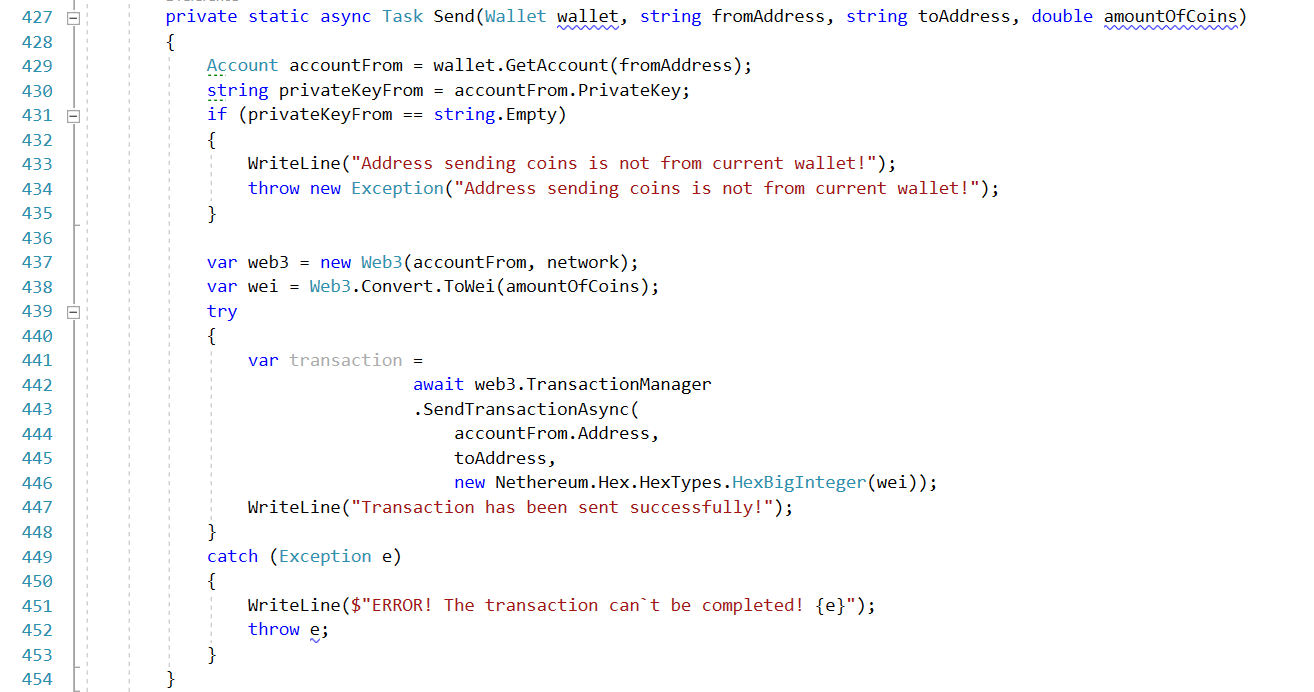
1. Existing wallet can be recovered from mnemonic phrase (words). For this purpose, we will create **Recover** method. This receives **mnemonic phrase** as **parameter** and generated wallet from **words**. Method print addresses and keys and return recovered **walle**
2. When program wants to recover **wallet,** it calls Method **RecoverFromMnemonicPhraseAndSaveToJson.** This functionfirst calls method **Recover** and recover **wallet** with **words.** Then **create** **new json file** with **password** and **path to file**. Function **return** recovered wallet.



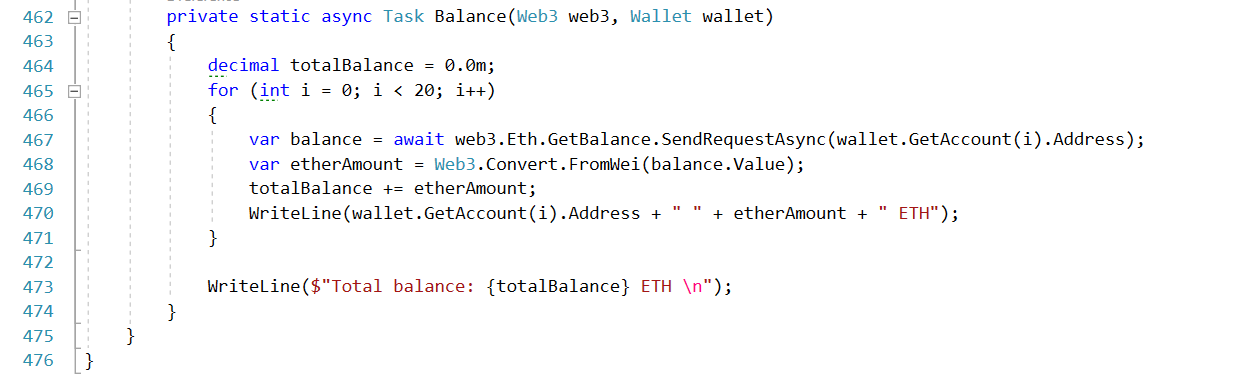
1. When the wallet is ready for use we can receive coins on it. The method **Receive** takes the **wallet as parameter** and printthe addresses in which the coins can be received.



1. When we want to **send** coins this is a little more complicated. Method **Send** receives as parameters **wallet**, **address sending** coins, **address receiving** coins and **amount**. Then we created **Account** sending coins. We take it from **wallet** using **address**. Then **get private key** from **account**. Then we **create new Web3** variable with **account** and **network**. Then **convert** amount of **ethers** to **wei** and make a transaction.



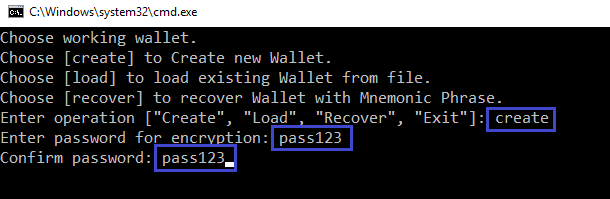
1. We can check the **balance** of our wallet any time with method **Balance**. It takes **Web3** and **wallet** as parameters. Connect to network and give as the **amount of ethers** of each **address** and **total balance** of the **wallet**.



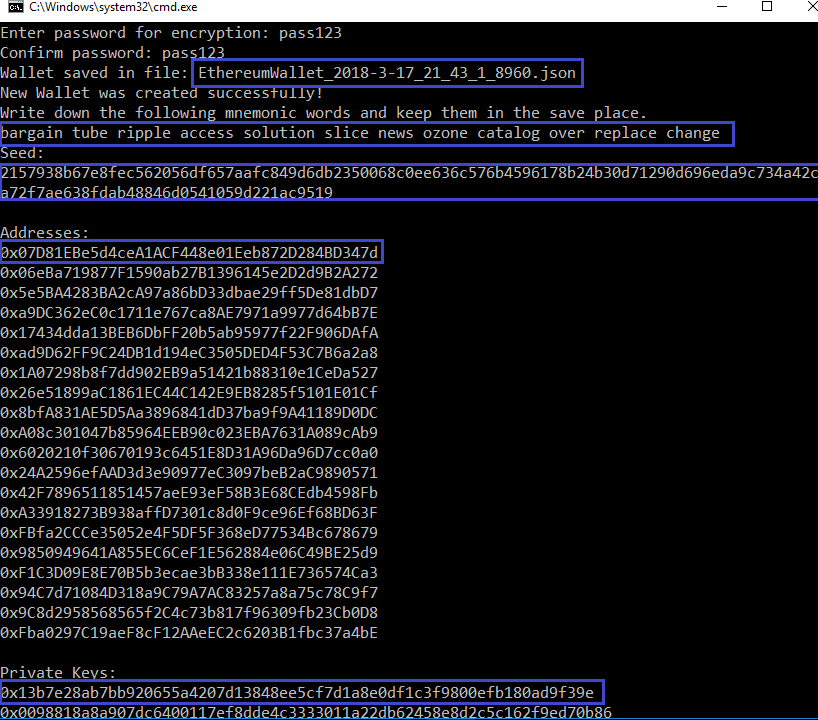
This is the whole code of our program. Now let`s see how all it works.

## Create New Wallet

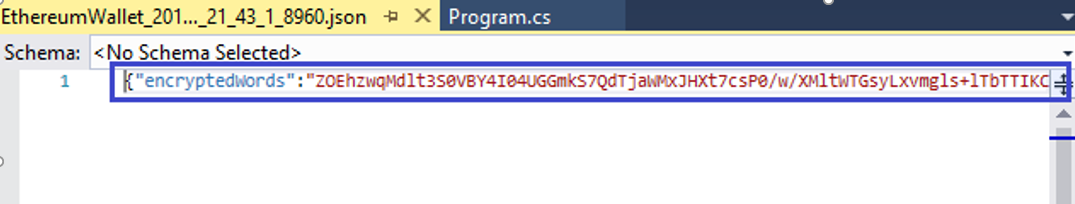
1. Start the program. Push [**ctrl + F5].**  In console write command **create** and press [**Enter**]**.** Program will ask you for the **password** and **password confirmation**.



1. If passwords are equal the **new wallet will be created**. First the program informs you about the **name of the json file**. Then shows the **mnemonic words**. **You must keep them in the save place!** Then you can see **seed**, **addresses** and **private keys**. **Keep the private keys in private and don`t show them to anybody! With private keys you can sigh transactions and arrange your coins!**

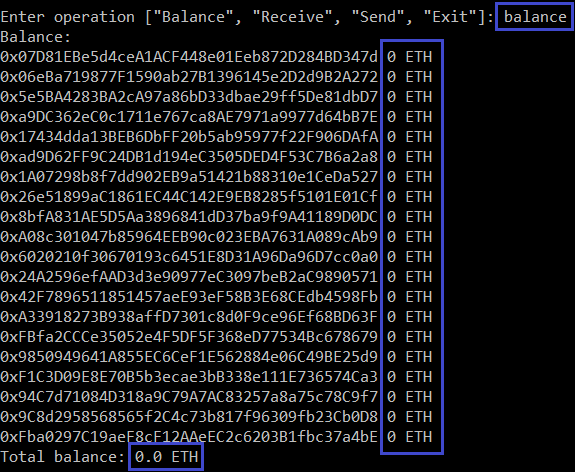


1. With wallet`s creation, in the “**Wallets**” directory the program **creates the json file** with **encrypted** wallet`s **pneumonic phrase** and **current time**.



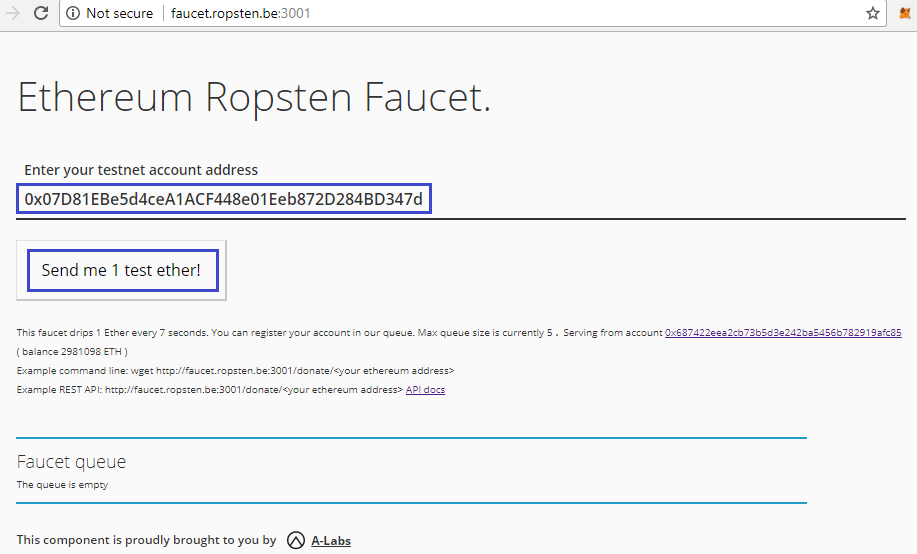
## Check balance

1. Now when the wallet is ready we can check the **balance**. Write command **balance** and press [**Enter**]

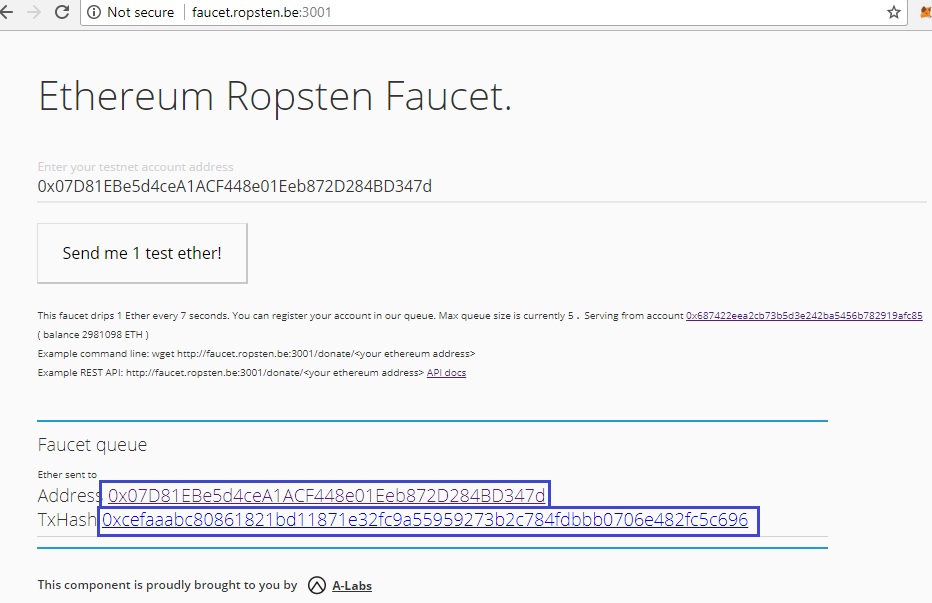


## Take coins from faucet

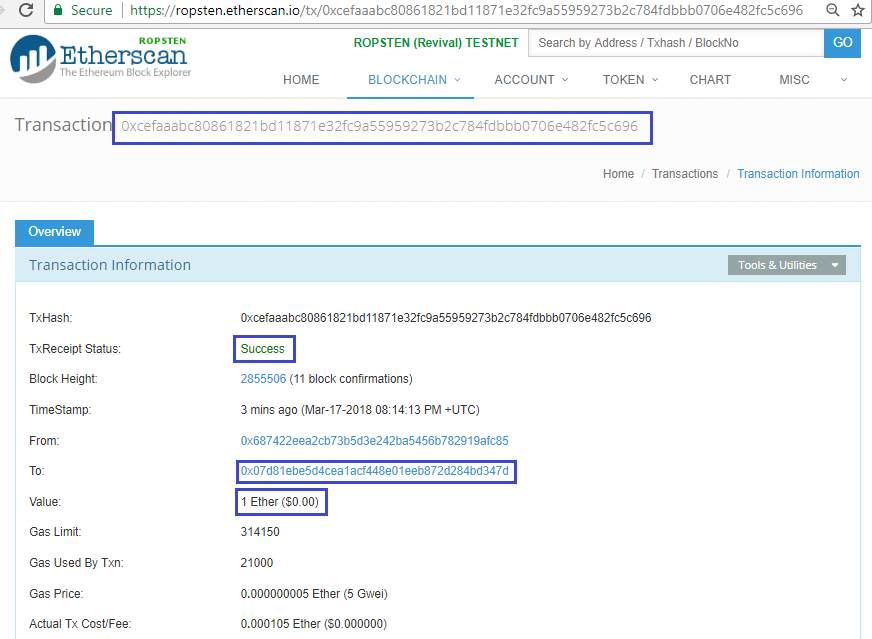
1. Of course the wallet is empty. We can feed it with some coins from the Ropsten ethereum faucet in <http://faucet.ropsten.be:3001/>Copy one of the addresses, paste it to the site and **ask for 1 test ether**.



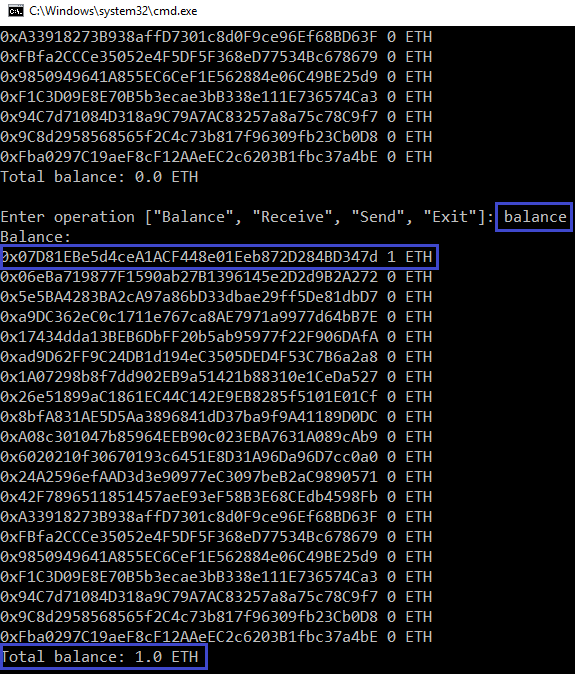
1. The **transaction** is on the way.



1. After a couple of time the transaction will be completed and we receive **1 ether**.

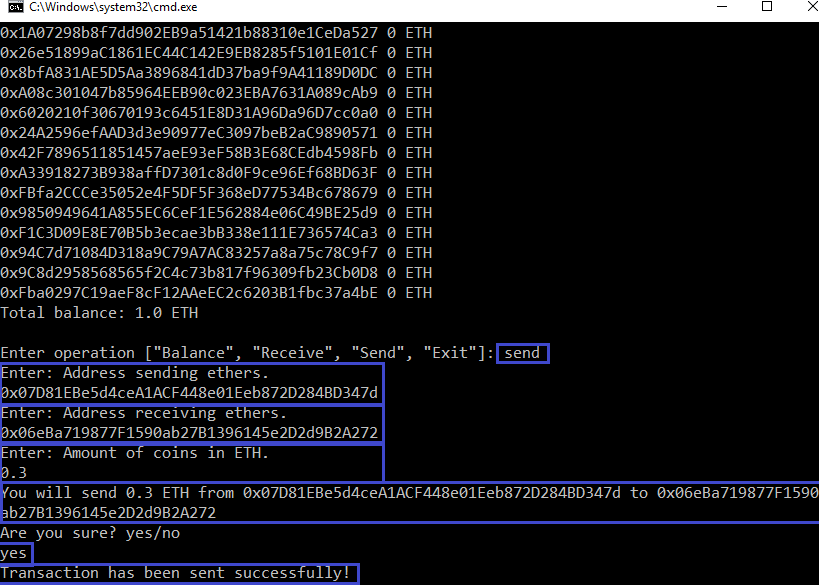


1. Let`s check the **balance** again. Great! We have one test ether in our wallet!

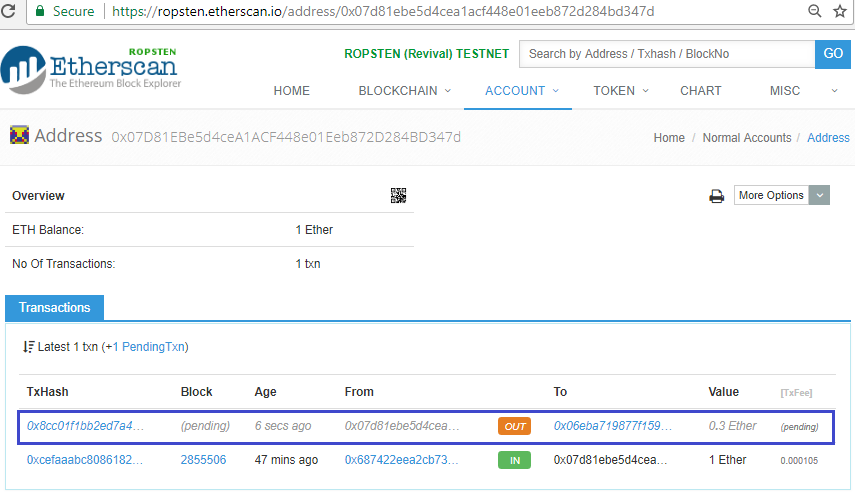


## Send coins

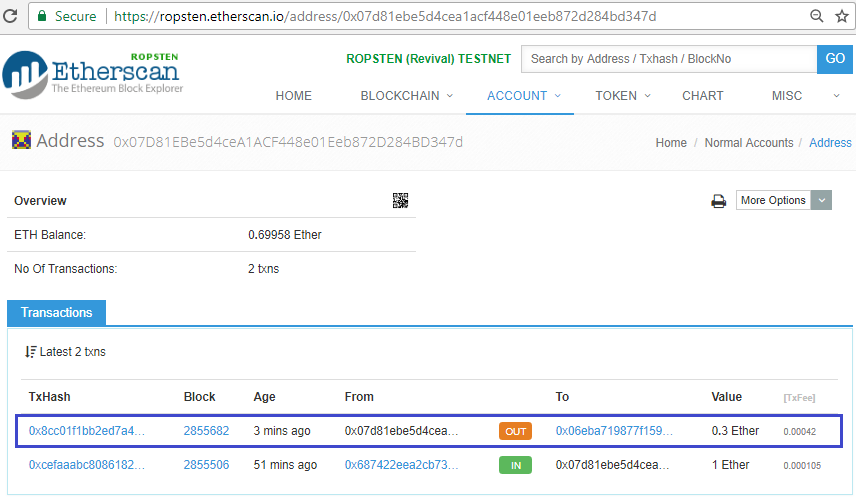
1. Now let`s send some coins to another address. Type command **send**. First copy and paste the **address** which **sending coins** and the **address receiving coins**. Then type the **amount of coins** in ethers.  
   Then the program will ask for **confirmation**, type “**yes**” and press [**Enter**].  
   The message "**Transaction has been sent successfully!**" will appear.



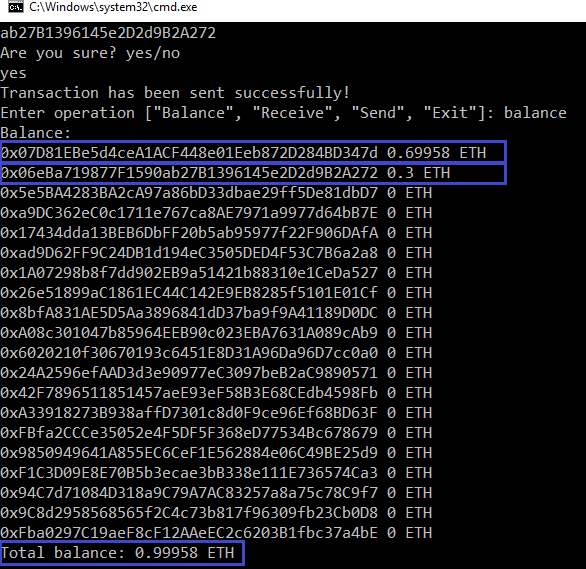
1. Our transaction is **successfully sent** and **pending**.



1. After a couple of time the transaction will be **mined** and new balances will be written in **blockchain**.

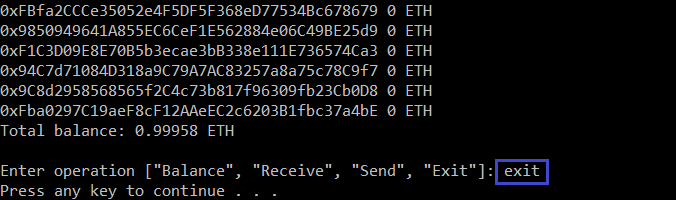


1. Let`s **check the balance** again. The second address was **received 0.3 ethers**. The total balance has lower than 1 ether because of the **fees** payed for the transaction.

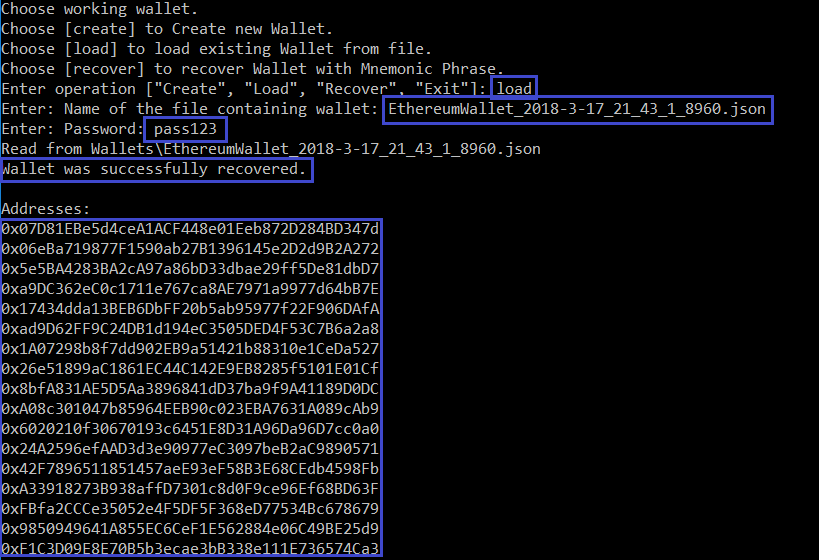


## Load the Wallet

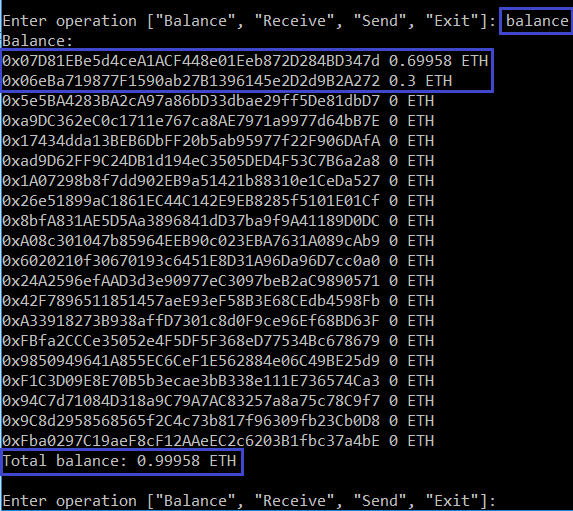
1. It`s time to close the program. Type “**exit**” and **press any key**. Next run the program again.



1. You see the well-known screen, but this time we already have created wallet and will **load** it from **json file**. Type command “**load**”, then write the **name of the file** and enter the **password** for the decryption of the file info.   
   You will see the message “**Wallet was successfully recovered**.” Here is the addresses and private keys from our wallet.

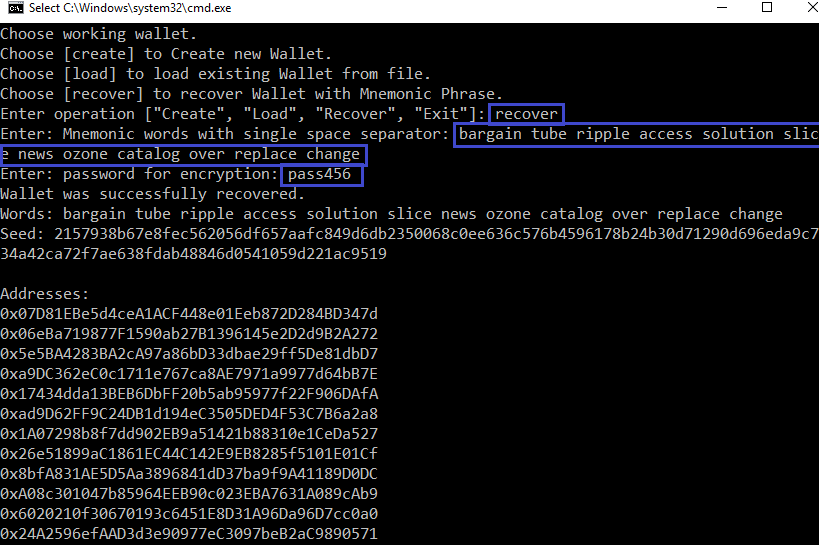


1. Check the **balance** like we did it before. Our test ethers are here.

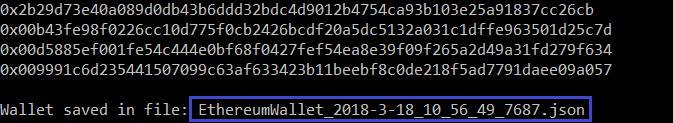


## Recover the Wallet

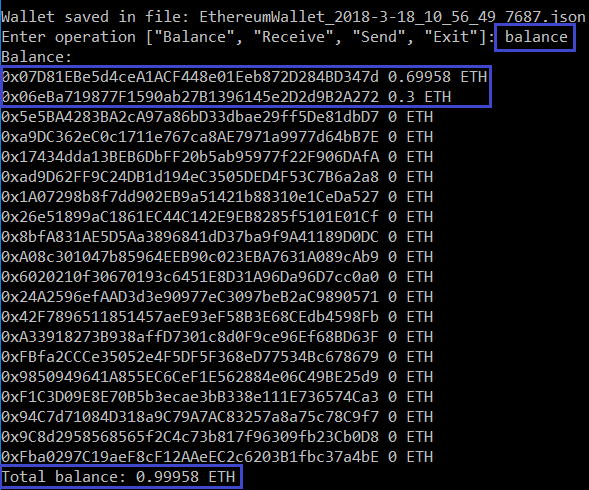
1. Now let`s assume that we want to use our wallet on the new computer. There isn`t exist saved json file and we must recover our wallet from mnemonic phrase. Do you remember your mnemonic words? Did you hide them in the save place? Now take them, you will need to use this words. Start the program and type command “**recover**”. Then write the **words**. The program will ask you for the **password** for json file encryption. Type the **new password** and press [**Enter**]. Wallet will be successfully recovered.



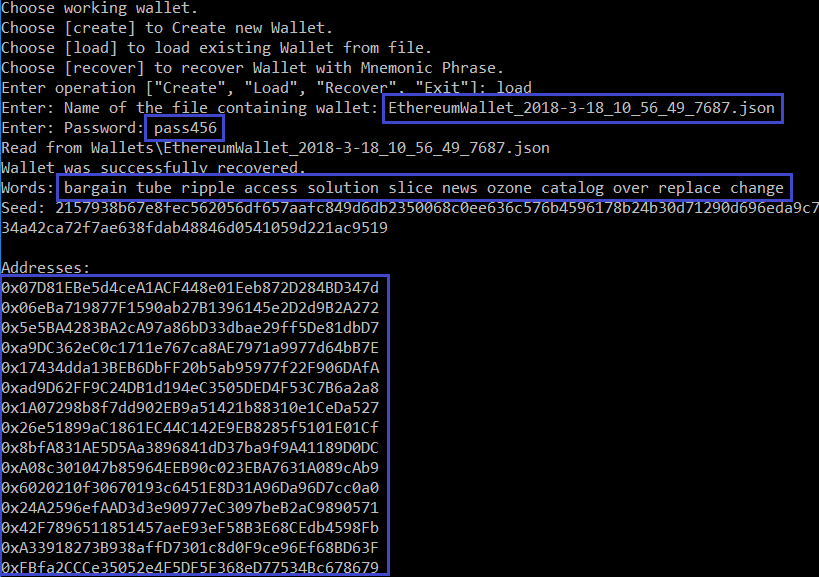
1. In the bottom of the screen is the name of the **new file keeping our wallet** info.



1. For any case check the **balance** again. Our test ethers are here.



1. For exercise load the wallet from new file using the new password.



# What to Submit?

Create a **zip file** (e.g. your-name-create-ethereum-wallet-exercise.zip) holding the source code files and screen shots with your experiments.

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