Contract: IssuanceStateMachine

Getting started:

1. Go to remix.ethereum.org.

2. Click on the "+" button on the upper left corner of the screen.

3. Name the file name anything appropriate.

4. Copy-paste the code file's contents into the IDE.

Note: Make sure that the compiler version is set to "0.4.24+commit.e67f0147" before trying to run the code, else errors will come.

How to use:

1. Once everything is set up, click on "Start to compile", found on the right side of the IDE.

2. Go to the "Run" tab

3. You'll then see the name of the contract "IssuanceStateMachine" on the right side of the IDE. Click on "Deploy".

4. On the "Deployed Contracts", found on the lower right side of the screen, you'll see the contract. Click on it.

5. You'll start off by inputting "1 Ether" into the Value field on the upper right side of the IDE under the "Run" tab.

6. Click on "buyIssuerRight" to store the value placed in the Value field before interacting with the "issueCertificate" function.

5. On the "issueCertificate" function, click on the "v" button to drop down two forms, "\_cert" and "issued\_to\_address".

6. Input the appropriate name for the certificate in the "\_cert" field.

7. For the address found in the "issued\_to\_address" field, you'll have to go to the "Account" field on the upper right side of the IDE and pick a different account with 100 ether. Then you'll have to copy its address value by clicking on the clipboard button beside it. Once that's done, set the "Account" back to the first default one.

8. Paste the copied address into the "issued\_to\_address" field.

10. Finally, click on the "Transact" button.

11. The code should mine the transaction and succeed in its execution.

12. You may also get the issuer hash by clicking on the "getIssuerHash" function button.

Context: The behavioral pattern State Machine sets the program into "stages" where they'll have to go through a prerequisite before they proceed with the next set of functions. In this case, the issuance of certificates is set to two (Three, if considering the "getIssuerHash" function) stages. First, the user will have to buy the right to issue a certificate before actually doing it. If you didn't pay 1 Eth to it, the process won't proceed further. If you're able to pay for it, you may now move on to certificate issuance.

Contract: Self Destruct

Getting started:

1. Go to remix.ethereum.org.

2. Click on the "+" button on the upper left corner of the screen.

3. Name the file name anything appropriate.

4. Copy-paste the code file's contents into the IDE.

Note: Make sure that the compiler version is set to "0.4.24+commit.e67f0147" before trying to run the code, else errors will come.

How to use:

1. Once everything is set up, click on "Start to compile", found on the right side of the IDE.

2. Go to the "Run" tab

3. You'll then see the name of the contract "IssuanceStateMachine" on the right side of the IDE. Click on "Deploy".

4. On the "Deployed Contracts", found on the lower right side of the screen, you'll see the contract. Click on it.

5. You'll start off by selecting the address you wish to remove.

6. Click on "removeContract" to remove the contract from the selected address.

5. On the "viewAddress" function, this is to view which address will be destroyed or removed.

Context: The behavioral pattern Self Destruct is set to permanetly remove a contract in the blockchain.

Contract: Oracle

Getting started:

1. Go to remix.ethereum.org.

2. Click on the "+" button on the upper left corner of the screen.

3. Name the file name anything appropriate.

4. Copy-paste the code file's contents into the IDE.

Note: Make sure that the compiler version is set to "0.4.24+commit.e67f0147" before trying to run the code, else errors will come.

Use:

The function of the Oracle states that you will pull the data file for the web base source onto the smart contract.

This imported file will be based on the github link and it will show on the smart contract with the bytes32 (your address) and a character/s that you will be input. In this case, the function will be based also on the current contract that you want to expand from the import.

Context: The behavioral pattern Oracle is set to access or fetch other datas from outside the blockchain.

Contract: Generate\_Credit\_Card\_Number

Getting started:

1. Go to remix.ethereum.org.

2. Click on the "+" button on the upper left corner of the screen.

3. Name the file name anything appropriate.

4. Copy-paste the code file's contents into the IDE.

Note: Make sure that the compiler version is set to "0.4.24+commit.e67f0147" before trying to run the code, else errors will come.

How to use:

1. Go to the “Run” tab and click on the “Deploy” button.
2. Click on the deployed contract on the lower right side of the IDE.
3. In the “generateRandom” field, enter any alphabetical word under quotes. It could be a name, a place, etc.
4. Click on the “generateRandom” button. If it works, the transaction should be mined and executed successfully.
5. Click on the “getRandomGenerate” button to see the text you’ve inputted plus the generated number.

Context: The behavioral pattern Randomness is set to generate random numbers in the blockchain.

Contract: Commit-Reveal

Getting started:

1. Go to remix.ethereum.org.

2. Click on the "+" button on the upper left corner of the screen.

3. Name the file name anything appropriate.

4. Copy-paste the code file's contents into the IDE.

Note: Make sure that the compiler version is set to "0.4.24+commit.e67f0147" before trying to run the code, else errors will come.

How to use:

1. Go to the “Run” tab and click on the “Deploy” button.
2. Click on the deployed contract on the lower right side of the IDE.
3. In the “generateRandom” field, enter any alphabetical word under quotes. It could be a name, a place, etc.
4. Click on the “generateRandom” button. If it works, the transaction should be mined and executed successfully.
5. Click on the “getRandomGenerate” button to see the text you’ve inputted plus the generated number.

Context: The behavioral pattern commit-reveal is used to keep secret of private data during a period of time in the blockchain.

Contract: Generate\_Credit\_Card\_Number

Getting started:

1. Go to remix.ethereum.org.

2. Click on the "+" button on the upper left corner of the screen.

3. Name the file name anything appropriate.

4. Copy-paste the code file's contents into the IDE.

Note: Make sure that the compiler version is set to "0.4.24+commit.e67f0147" before trying to run the code, else errors will come.

How to use:

1. On the “Deploy” button under the “Run” tab, you’ll notice how it has a text field beside it. Click on the “v” button for it to drop down.
2. Fill out the following fields:
   1. “\_commitPhaseLengthInSeconds” to 10 seconds or more. If it’s less than 10 seconds, the deployment won’t go through.
   2. “\_choice1” represents the first candidate. Fill it with a name.
   3. “\_choice2” represents the second candidate. Fill it with a name.
3. Once that’s filled out, click on the “Transact” button to deploy the contract.
4. Now, you’ll have to use keccak256 hashes in order to vote and keep track of it.
   1. Go to <https://emn178.github.io/online-tools/keccak_256.html>
   2. On the “Input” field, type in the first candidate by following this format, “1-[Insert first candidate name here]. Make sure it doesn’t contain any spaces if you typed in a full name. (Example: 1-Batman, 1-BruceWayne)
   3. If you’re voting for the second candidate, simply change the value to “2”. (Example: 2-Luthor, 2-LexLuthor)
   4. Below the field, it’ll automatically generate a hash. Copy that and place it in fields that require a “bytes32” input. Make sure to add “0x” in the beginning of the hash. (Example: 0x26c7c5c38eaeaa66c32b8621dd974e78783949751f893471c94b274d746b2357)
5. On the “CommitField” function, click on the “v” button and it’ll expand. In the “\_voteCommit” field, add the hash of the candidate of your choosing in the bytes32 field. Click on “Transact” to proceed. Note, this only commits the data during the committing period that you’ve set before.
6. On the “revealVote” function, click on the “v” button and it’ll expand. In the “\_vote” field, add the name of the candidate of your choosing under the same format as step 4b. For “\_voteCommit”, add the hash of the candidate of your choosing in the bytes32 field. Click on “Transact” to proceed. This will reveal the votes after the committing period is over.
7. Clicking on “choice1” displays the first candidate’s name.
8. Clicking on “choice2” displays the second candidate’s name.
9. Clicking on “commitPhaseEndTime” displays the time you have in long seconds.
10. Clicking on “getWinner” displays the winner with the most votes. It could also display a tie.
11. Clicking on “numberOfVotesCast” displays the overall number of votes placed between the candidates.
12. The function “voteCommits” verifies the vote by using the same hash that you used to vote for the candidate.
13. Clicking on “votesForChoice1” reveals the number of votes for the first candidate.
14. Clicking on “votesForChoice2” reveals the number of votes for the second candidate.

Context: The behavioral pattern commit-reveal is used to keep secret of private data during a period of time in the blockchain.