WEEKEND PROJECT 3 REPORT – GROUP 6

ENCODE SOLIDITY BOOTCAMP Q3 2023

This week's project involved deploying and interacting with a Tokenized Ballot contract (*TokenizedBallot.sol*) and the corresponding Token contract (*ERC20Votes.sol*) used in the Ballot. To perform the interactions, we had to write scripts (*typescript*) that would allow the group to interact with the contracts, deploying, minting tokens, transferring, delegating voting power, voting, and querying info/results.
Instead of breaking each interaction (function call) into a separate script file we opted to only have one file, for each contract, with all the functions and have a conditional statement to ensure that each function (including the deploy function) could be called from the terminal with arguments if needed.
Unlike what happened with the simple Ballot in the previous Weekend Project 2 where each account simply could or could not vote on one proposal, this Tokenized Ballot considers that the voting power is equivalent to the number of Tokens held by the account and delegated to itself before voting. This way also allows the voter to spread the voting power between the different proposals as desired.
Firs step was to deploy the Token contract and distribute tokens to the various individual accounts, after which some transfers and delegations followed. When everything was set, and at a determined block number, the Tokenized Ballot contract was deployed, and voting begun.
As an example, the following shows how to deploy the Tokenized Ballot contract and how to vote on the second proposal (Batman) with 50 units of voting power:

yarn run ts-node --files ./scripts/TokenizedBallot.ts deployBallot Spiderman Batman Superman

yarn run ts-node --files ./scripts/TokenizedBallot.ts vote 2 50

1 – ERC20Votes.ts (Token contract interaction)

1.1 – Contract deployment

The contract was deployed from the script file using the *deployToken* function.

```
● → Weekend_project_3 git:(main) ✗ yarn run ts-node --files ./scripts/ERC20Votes.ts deployToken

Deploying G6Token contract...
Group 6 Token contract deployed at 0x9805944Da4F69978dffc4c02eA924911D668d81a
```

Contract address: 0x9805944Da4F69978dffc4c02eA924911D668d81a
Transaction hash: 0x678c64de5dab2133de71a088ef7351866853607f6b56aa6f60885ed4d044f97d

To start the interaction with the Token and Tokenized Ballot contracts, using the script files, we had to update the variable *tokenAddress* with the newly created contract address. From here the first interactions with the Token contract come from the contract creator's account (who gets *MINTER_ROLE*) since he is the only one who can mint tokens to the other accounts.

1.2 – mint(address *to*, uint256 *amount*)

This function mints a number of tokens (*amount*) to the receiver address (*to*).

Transaction hash: [0x3a315e94ba662c5f2b1fca1dcd4026e7db0d27bd6ff5f81e922af82e7716c56e]

When another account that does not have the *MINTER_ROLE* tries to mint tokens, the transaction is reverted with the error: *"AccessControl: account 0x27…d9 is missing role"*.

1.3 – delegate(address *to*)

This function allows the caller to delegate the voting power (equivalent to the tokens held) to another account (*to*) or to itself (passing own account) to get voting rights.

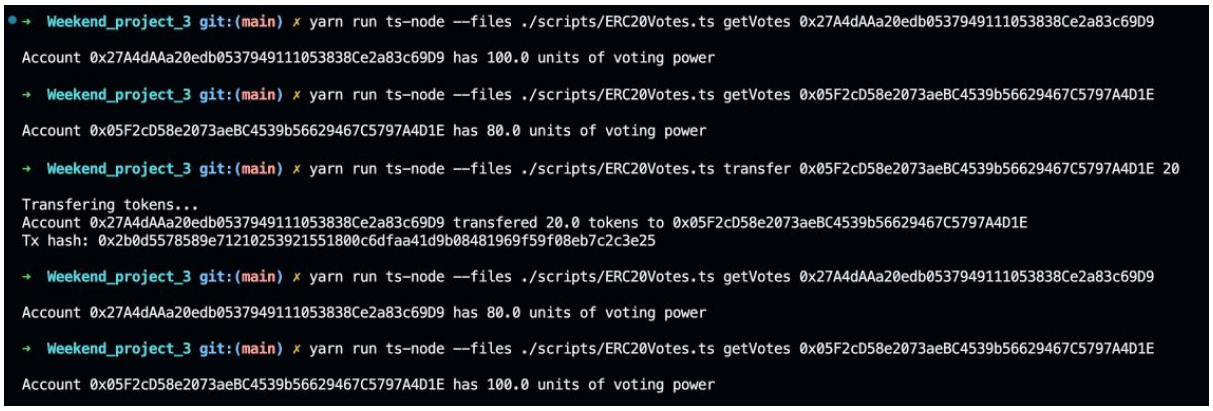
Transaction hash: [0xb24b4f814f00b44649c735aee5cbdc000061097de289d492828ecdca7dc17261]

1.4 – transfer(address *to*, uint256 *amount*)

This function allows the caller to transfer a number of tokens (*amount*) to another account (*to*).

Transaction hash: [0x75b4866d1ec1d55430d474214b58604b0171b129dd0bba99c522b08d44dcf44e]

If an account has voting power and afterwards decides to transfer some number of tokens to another account, it loses the equivalent units of voting power. These tokens add up to the receiver's account increasing its voting power:



If an account tries to transfer more tokens than the ones held, the transaction is reverted with the error: "*ERC20: transfer amount exceeds balance*"

1.5 – balanceOf(address *account*)

This function represents a query function that will return the number of tokens held by the passed *account* address.

```
→ Weekend_project_3 git:(main) ✗ yarn run ts-node --files ./scripts/ERC20Votes.ts balanceOf 0x05F2cD58e2073aeBC4539b56629467C5797A4D1E

Account 0x05F2cD58e2073aeBC4539b56629467C5797A4D1E has 80.0 tokens
```

1.6 – getVotes(address *account*)

This function represents a query function that will return the units of voting power held by the passed account address (*account*).

```
→ Weekend_project_3 git:(main) ✗ yarn run ts-node --files ./scripts/ERC20Votes.ts getVotes 0x27A4dAAa20edb0537949111053838Ce2a83c69D9

Account 0x27A4dAAa20edb0537949111053838Ce2a83c69D9 has 80.0 units of voting power
```

1.7 – getPastVotes(address *account*, uint256 *timepoint*)

This function represents a query function that will return the units of voting power held by the passed address (*account*) at a given block, set by passing the block number (*timepoint*). As an improvement we can also see the amount at the latest block for reference.

```
→ Weekend_project_3 git:(main) ✗ yarn run ts-node --files ./scripts/ERC20Votes.ts getPastVotes 0xbdC1B3Beb4Ee43Ff40629B0371b808C903E2227f 4080337

Account 0xbdC1B3Beb4Ee43Ff40629B0371b808C903E2227f had 100.0 units of voting power at block 4080337
Account 0xbdC1B3Beb4Ee43Ff40629B0371b808C903E2227f has 120.0 units of voting power at latest block 4082801
```

2 – TokenizedBallot.ts (Tokenized Ballot contract interaction)

2.1 – Contract deployment

The contract was deployed from the script file, passing in 4 arguments, the *deployBallot* function and the names for 3 *proposals* (Spider-Man, Batman and Superman).

Before deployment some variables had to be updated. The variable *tokenContract* with the previously deployed Token contract address and the variable *targetBlockNumber* with the block number as a limit to consider the present voting rights (at the given block). By using this target block it is guaranteed that any interaction happening after the deployment of the contract won't affect the end result of the Ballot.

```
→ Weekend_project_3 git:(main) ✗ yarn run ts-node --files ./scripts/TokenizedBallot.ts deployBallot Spider-Man Batman Superman

Deploying TokenizedBallot contract...
Proposals:
Proposal N. 1: Spider-Man
Proposal N. 2: Batman
Proposal N. 3: Superman
Contract deployed at address 0x86194b8C24DB66Ef9ACFA70b4c2fc837F0684961
```

Contract address: 0x86194b8C24DB66Ef9ACFA70b4c2fc837F0684961
Transaction hash: 0x5e4bee8ac9caeb6852d6b61b8a42c93bea0edbca4f2206a769bf704f3815461e

After deployment and to begin interactions using the script file, we had to update the variable *tokenizedBallotAddress* with the deployed Tokenized Ballot address. At this point we can start to interact with the contract using the script.

2.2 – vote(uint *proposal*, uint256 *amount*)

This function casts a number of votes (*amount*) on the proposal's array index (*proposal*).

Regarding our script we changed it so that the vote will be cast on the proposal's number and not the index. To vote on proposal number 1 (index 0) we call the function with the argument set to 1. After voting we can also see how much voting power we have left to use.

```
● → Weekend_project_3 git:(main) ✗ yarn run ts-node --files ./scripts/TokenizedBallot.ts vote 1 50

Vote for proposal 1: "Spider-Man" submited...
Remaining voting power: 50.0
Tx hash: 0x7eb7d247cd9dbf6873940abae8940e2abc78d336ea7bde73dd30491ecfa971b0
```

Transaction hash: 0x7eb7d247cd9dbf6873940abae8940e2abc78d336ea7bde73dd30491ecfa971b0

If an account tries to vote with more than the voting power available, the transaction will revert with the error: *"TokenizedBallot: trying to vote more than allowed"*.

If the vote is set to an index outside of the array's range, the transaction will revert.

2.3 – votingPower(address *account*)

Similar to the *getVotes* function from the Token contract, this function represents a query function that will return the units of voting power held by the passed account address (*account*).

```
● → Weekend_project_3 git:(main) ✗ yarn run ts-node --files ./scripts/TokenizedBallot.ts votingPower 0x27A4dAAa20edb0537949111053838Ce2a83c69D9

Account 0x27A4dAAa20edb0537949111053838Ce2a83c69D9 has 20.0 units of voting power
```

2.4 – checkProposals()

This function is exclusive from the script file, and it is a simple way to check the numbers, names and number of votes cast, for each proposal.

```
● → Weekend_project_3 git:(main) ✗ yarn run ts-node --files ./scripts/TokenizedBallot.ts checkProposals

Available Proposals

Number: 1
Name: Spider-Man
Votes: 110.0

Number: 2
Name: Batman
Votes: 75.0

Number: 3
Name: Superman
Votes: 10.0
```

2.5 – winningProposal()

This represents a query function that will return the index of the proposal with the most votes. In the case of our script, it will output the proposal's number and not the index.

In the case that no vote has been cast, when this function is called it will return the first proposal as the winner.

```
● → Weekend_project_3 git:(main) ✗ yarn run ts-node --files ./scripts/TokenizedBallot.ts winningProposal

The winning proposal number is: 1
```

2.6 – winnerName()

This represents a query function that will return the name of the proposal with the most votes. In the case that no vote has been cast, when this function is called it will return the first proposal's name as the winner.