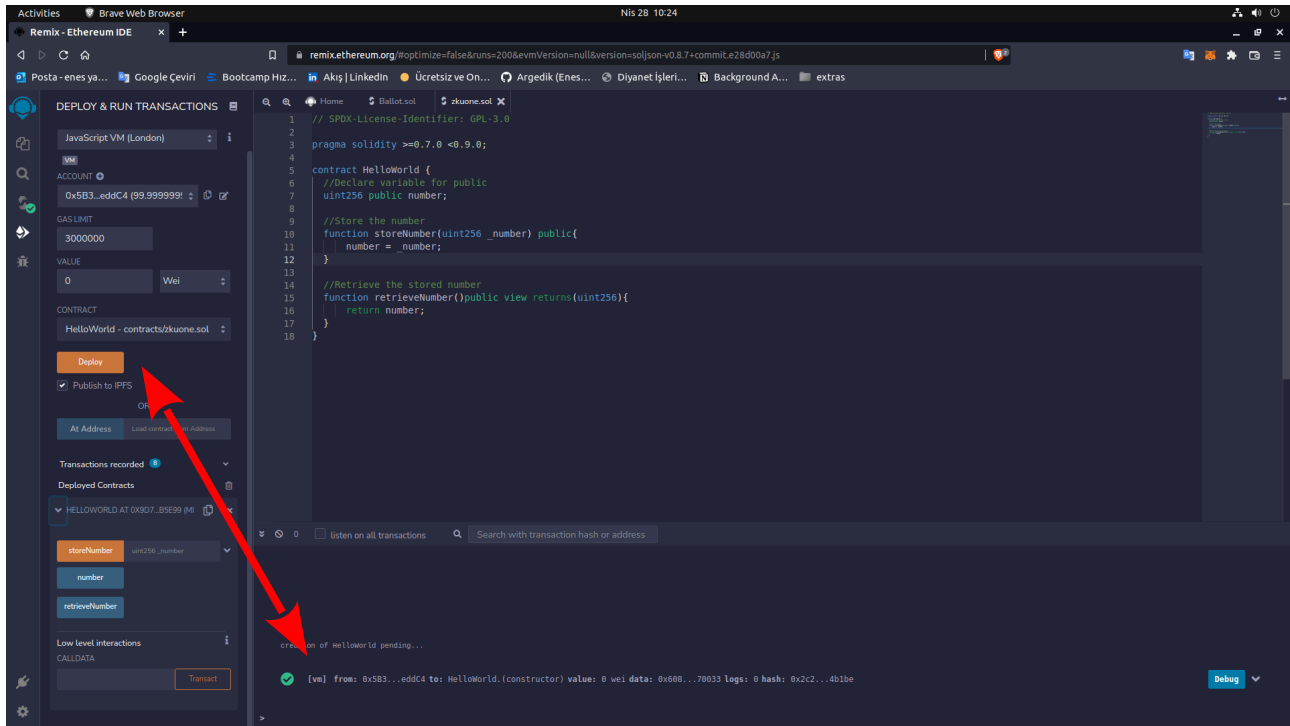
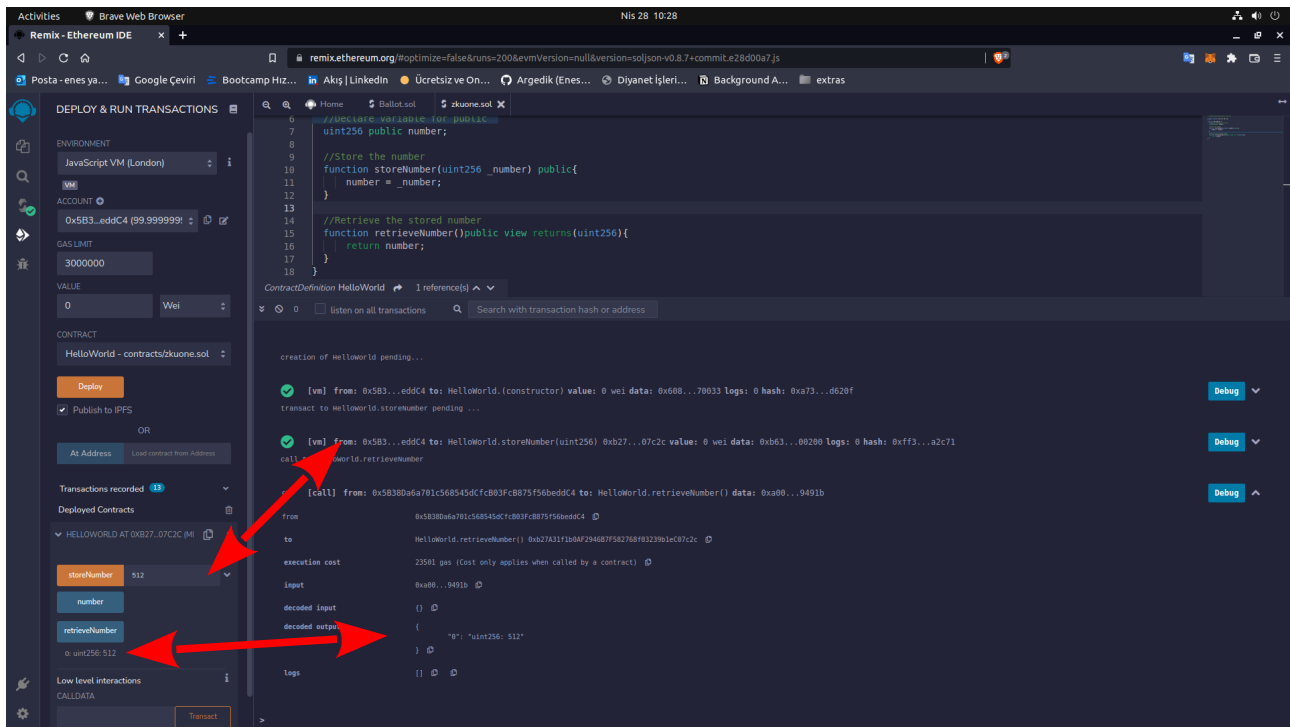


# Background Assignment

Let's start by deploying the "Hello world" smart contract.



We write the "storeNumber" function to store our unsigned number. For example; "512"  
We write the "retrieveNumber" function to display the unsigned number we have stored.



# Background Assignment

Here we create several new wallet accounts to be able to use the "ballot contract".

```
createBytes.js > ...
1 const ethers = require('ethers');
2
3 async function createBytes(args) {
4   const name = args[0];
5   const bytes = ethers.utils.formatBytes32String(name);
6   console.log("Bytes:", bytes);
7 }
8
9 createBytes(process.argv.slice(2));
```

```
argedik@Argedik:/media/argedik/Hdd/Linux_dekstop/Web3/Solidity_Background_Assignment$ node createBytes.js enes
Bytes: 0x056e573000000000000000000000000000000000000000000000000000000000
argedik@Argedik:/media/argedik/Hdd/Linux_dekstop/Web3/Solidity_Background_Assignment$ node createBytes.js yasin
Bytes: 0x796173696e0000000000000000000000000000000000000000000000000000
argedik@Argedik:/media/argedik/Hdd/Linux_dekstop/Web3/Solidity_Background_Assignment$ node createBytes.js gedik
Bytes: 0x676564696b00000000000000000000000000000000000000000000000000
argedik@Argedik:/media/argedik/Hdd/Linux_dekstop/Web3/Solidity_Background_Assignment$
```

After compiling our "ballot.sol" file, we enter the addresses we created in the "Deploy" section as a list.

Example;

```
["0x656e573000000000000000000000000000000000000000000000000000000000",
"0x796173696e0000000000000000000000000000000000000000000000000000",
"0x676564696b00000000000000000000000000000000000000000000000000"]
```

```
creation of Ballot pending...
[✓] [wei] -> 0xA88...35cb2 to: Ballot.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0xd82...f7c1e
true Transaction mined and execution succeed
0xd820f5160706b8d1acae8938452a5a4b36ee41e96156217459807277c1e
from 0xA0843f649C641eCf90849a6f7d03115835c2
to Ballot.(constructor)
gas 130652 gas
transaction cost 1162219 gas
execution cost 1162219 gas
input 0x608...00000
decoded input {
  "bytes32[] prpocessNames": [
    "0x056e573000000000000000000000000000000000000000000000000000000000",
    "0x796173696e0000000000000000000000000000000000000000000000000000",
    "0x676564696b00000000000000000000000000000000000000000000000000"
  ]
}
```

## Background Assignment

We start our 5-minute period with "startTimer". We can learn when "Timer" starts with ".start" and when it ends with "getTimeLeft".

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is open, displaying a list of transactions. The 'startTimer' transaction is highlighted, showing its details. In the center, the 'STRUCT DEFINITION PROPOSAL' panel shows the function signature for 'startTimer'. On the right, the 'TRANSACTIONS' panel displays a list of transactions, with the 'startTimer' transaction highlighted. Red arrows point from the 'startTimer' transaction in the left panel to the 'startTimer' transaction in the right panel, and from the 'startTimer' transaction in the right panel to the 'startTimer' function signature in the center panel.

If we can't vote within 5 minutes according to the code we wrote in our contract, we will get the "timer is done" error.

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is open, displaying a list of transactions. The 'vote' transaction is highlighted, showing its details. In the center, the 'STRUCT DEFINITION PROPOSAL' panel shows the function signature for 'vote'. On the right, the 'TRANSACTIONS' panel displays a list of transactions, with the 'vote' transaction highlighted. A red arrow points from the 'vote' transaction in the left panel to the 'vote' transaction in the right panel, and from the 'vote' transaction in the right panel to the 'vote' function signature in the center panel. The 'TRANSACTIONS' panel shows an error message: 'Transaction to Ballot.vote errored: VM error: revert. Reason provided by the contract: "timer is done".' This indicates that the transaction failed because the timer had expired.

# Background Assignment

Let's vote in less than 5 minutes :)

The screenshot displays the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows a list of transactions. A red box highlights the 'chairperson' transaction, which is the first in the list. The transaction details show the address 0x58380a6a701c368545dcf803fc8b75f56beddc4 and the function 'getTimelock'. The transaction status is 'success'.

The main panel shows the smart contract code and the transaction logs. The logs show the execution of the 'chairperson' transaction, including the 'startTimer' function call and the 'vote' function call. A red box highlights the 'vote' function call, which is the last transaction in the list. The transaction details show the address 0x58380a6a701c368545dcf803fc8b75f56beddc4 and the function 'vote'. The transaction status is 'success'.

<https://github.com/Argedik/ZKU-ONE-Background-Assignment>