CMPE 483 Assignment I Documentation

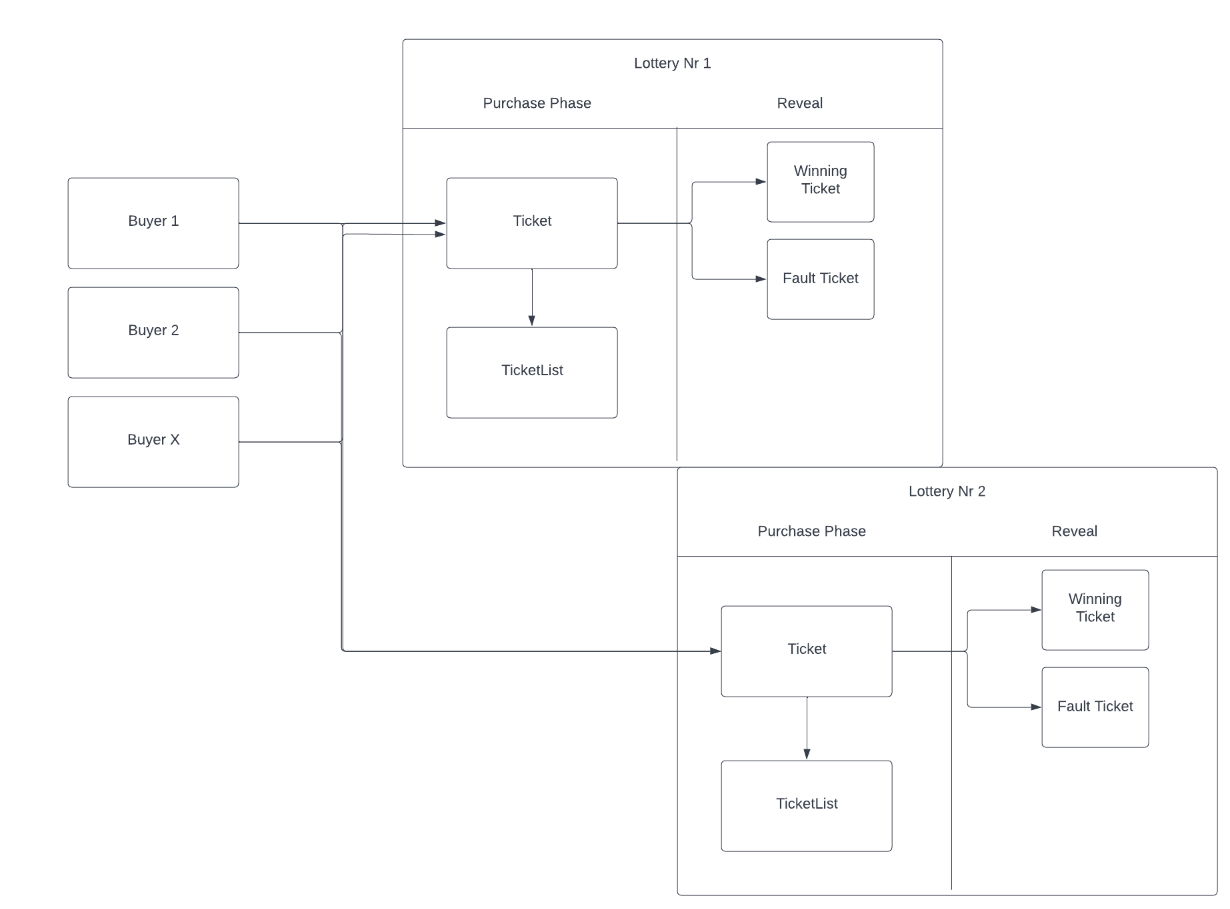
Can

Marcel Namyslo

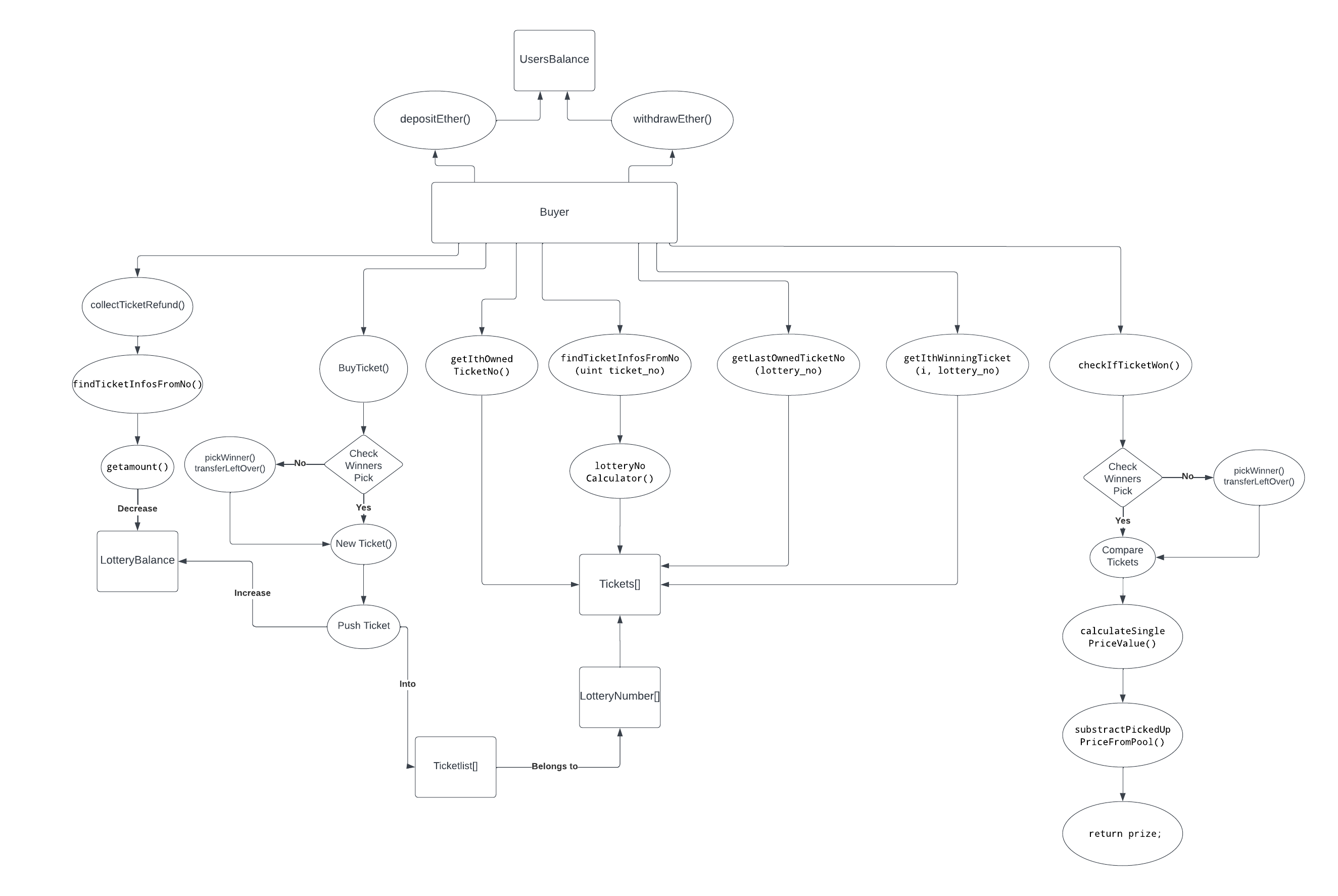
Introduction

This documentation gives a briefly insight about our implementation of an autonomous decentralized lottery system within a Blockchain environment. Herefore we used Solidity and the Geth developer environment to deploy our contract on a ficticious node within a locally runned Ethereum Blockchain.

System overview



Functions Overview



**Contract**

Our contract, which is called „Lottery“ has four properties:

* uint ticketNoCounter;
* uint lotteryDeploymentTime = block.timestamp;
* uint lotteryBalance;

We count the total number of tickets bought in ticketNoCounter,

the start timestamp when the lottery system is started in lotteryDeploymentTime and the total amount of money excisting in the Lottery in lotteryBalance.

The contract can then be deployed by it`s constructor

**Datastructure**

In order to storage our data and find them later by reference, we used three Solidity Mappings structures :

**TicketNosInArray**

FirstMapping safes for every single ticket number a reffering user address. This will be always the adress which bought the ticket with the cetain ticket number.

User`s address → Ticket number

**Old** (Mapping is a three dimensional Array, which safes every single Ticket in a Ticket Array, which is then safed in an Array of Integeters which represent the Lotterynumber. This Lotterynumber Array can then be referenced to a single Buyers adress. Through this structure, we safe every Ticket bought in each Lottery Period for every Buyer Adress of Buyers → Lottery Numbers → Tickets[…] → single Ticket)

**Balance**

Second Mapping safes the information of the balance for every buyer

**lotteryInfos**

ThirdMappingrepresents the Information of a single Lottery Period. It contains the lottery number, an Unix start timestamp, an array of the winning tickets belonging to this certain Lottery and an Array which safes the ticket number of every ticket belonging to the ticket. In the Array winning tickets, we safe the index of each winning ticket to find it in the ticketNosinLottery

For example: A Lottery in the fourth week has lottery number 4, a timestamp like 20370234810248, 3 winning tickets which has to be determined, and 10 tickets which could have ticket number 88,89,90,91,92. After the three winning tickets has been determined, e.G. 89,90 and 92, we safed the position(index they had in the ticketNosinLottery into the winningtickets array.

lotteryNumber → informations for this lottery

**moneyCollectedForEachLottery**

Fourth Mapping safes the amount of money collected for each single lottery. The Lottery period number is used as Key. By this, we can easily access the amount of money in each lottery by it`s lottery number

lotteryNumber → balance at this lottery

**totalPrizeMoney**

Fifth Mapping safes the prize money computed on all three winning tickets in each single lottery. The Lottery period number is used as Key. Each single price money has to be calculated for each winning ticket and then added up to a sum and safed in this mapping

lotteryNumber → prize at this lottery

**sixth Mapping**

ticketsFromOutside

This mapping safes every Ticket and it`s informations with his own ticket number as key. Every single ticket can then be accessed by its number.

Ticket number → ticket

**Structs:**  
We created a struct/class called „Ticket“ to create a Ticket when it is bought and then push it into our data storage. We safe several informations in each ticket:

* address owner:
* uint ticketNo:
* uint lotteryNo:
* bytes32 ticketHash
* uint8 status; //revealed or not
* bool active; //Todo Can will add modifier to that
* TicketTier ticketTier;

The second struct is the „LotteryInfo“ struct to safe all the attributes of a single lottery. Attributes are:

lotteryNo: period of this lottery

uint startTimestamp: timestamp the lottery has started (UNIX)

uint[] winningTickets: Array of the three winningTickets, safes each ticket number

uint[] ticketNosInLottery Every ticket number bought in this lottery period

**Addition:**

**ticketNosInLottery** is an array which safes the ticket number of each ticket and uses the senders address, on which the ticket has been bought, as the key

**Functions Description**

1. **depositEther**

function depositEther(uint amnt) public payable

This function can be used by a user to add Ether to his own balance. This basic function makes it than possible for him to buy tickets afterwards.

2. **withdrawEther**

function withdrawEther(uint amnt) public

Makes it possible for a user to withdraw money from the contract and his balance.

3. **buyTicket**

function buyTicket(bytes32 hash\_rnd\_number, int tier) public

With buyTicket(), a user is able to purchase a ticket with the form of his choice(full, half, quarter).

It differs between the different types of tickets and checks if the user has eough balance on his account to purchase the requested ticket.

We then add the information about the ticket to the ticketsFromOutside mapping, which safes every ticket and it`s information and uses the ticket number as key.

Then the balance of the user is reduced by the price of the ticket, while the total lottary balance is increased as well as the balance information of the single lottery itself(in moneyCollectedForEachLottery)

4. collectTicketRefund

function collectTicketRefund(uint ticket\_no) public

By this function, user can refund the ticket he has bought before. It needs to check some requirements first, before adding the amount of the ticket price back to the users balance and substracting the money from the lottery balance. It is prerequesite that the revealing phase has not started yet and the user is still at the buying stage while calling the refund function.

5. revealRndNumber

//Could to explain this more briefly pls

function revealRndNumber(uint ticketno, uint rnd\_number) public

By calling this function, a number that the user provided us will be revealed at reveal stage.

The caller has to be the owner of the ticket to call this function and can only reveal it once

6. getLastOwnedTicketNo

function getLastOwnedTicketNo(uint lottery\_no) public view returns(uint,uint8 status)

This function returns the recent/last ticket a user has purchased in a given lottery round.

7. getIthOwnedTicketNo

function getIthOwnedTicketNo(uint i,uint lottery\_no) public view returns(uint,uint8 status)

This function finds the ticket the user is looking for by giving the lottery number the ticket should be in and the i-th ticket he is searching for. Since we safed every ticket number in TicketNosArray and used the buyers address as key, we can easily access the i-th index and by this the i-th ticket and it`s number and status.

8. checkIfTicketWon

function checkIfTicketWon(uint lottery\_no, uint ticket\_no) public view returns (uint amount)

Checks if a ticket given by it`s ticket number and the lottery number it belongs to has won.

First, severel requirements regarding ownership, timing and status of the ticket has to be passed first.

Then the ticket number of the three winning tickets has to be determined. Those can be found in the winningTicket Array, which safes the ticket number of the three winning tickets and is located in the lotteryInfos Array for each lottery.

If then one of the three winning ticket`s numbers matches with the given ticket number, the amount of prize money has to be calculated with the calculateSinglePrizeValue helpers function. (Look up in the doc for calculateSinglePrizeValue). If the ticket won, the prize money will be returned, otherwise it will be 0.

9. collectTicketPrize

function collectTicketPrize(uint lottery\_no, uint ticket\_no) public

By calling collectTicketPrize, the user can add the prize won by his ticket into his balance. After giving the lottery number and ticket number and passing the requirements (ownership of ticket, existence of ticket, lottery existence and reveal status), we search for a matching winning ticket. For this, we get the winning tickets index by iterating through the winningtickets array (lotteryInfos[lottery\_no].winningTickets[i]) and find the ticket number in ticketNosInLottery.

If this ticket number matches with one of the winning ticket we calculate the prize with the calculateSinglePriceValue helpers function again.

At the end, lottery balance is reduced and owners balance is increased.

10. getIthWinningTicket

function getIthWinningTicket(uint i, uint lottery\_no) public view returns (uint ticket\_no,uint amount)

Returns the i-th winning ticket of a given lottery period. As we safed every lottery winners ticket, we can easily access the winners ticket index through the winnersTicket array and search for the ticket number in ticketNosinLottery by using the index.

Again, we use calculateSinglePriceValue to calculate the value of the winning ticket.

11. getLotteryNos

function getLotteryNos(uint unixtimeinweek) public view returns (uint lottery\_no, uint lottery\_no)

12. getTotalLotteryMoneyCollected

function getTotalLotteryMoneyCollected(uint lottery\_no) public view returns (uint amount)

This functions calculates the money collected in a given lottery period. This information can be easily found in the moneyCollectedForEachLottery by just using the lottery number as index.

Additional functions:

-function lotteryNoCalculator()

**Problems** we had to face and how we solved them:

1. Problem

Three winners tickets, the type of the tickets and the amount of winnable money has to be calculate, before the next lottery rounds starts. The amount of money which is left has then to be transfered to the next lottery round.

Since we cannot trigger functions after a certain time,

we have to trigger them inside other functions.

The function we have to trigger in order to compute the winners ticket and the amount of money to be substracted from the total lottery money amount is the **pickWinner function.**

It will find the three winners tickets and the amount of money we have to send to the winners.

This function has to be triggered:

* before someone in the next round can buy a new ticket and change the total lottery money amount
* in the getIthWinningTicket function when someone wants to get his i-th ticket
* in the collectTicketPrice function when the user wants to collect his prize
* as well as when some of the ticket owners asks for if his ticket won (checkifticketwon).

2. Problem

Define how to transfer prize money and handle leftover money

After the winnersticket are selected (By calling the BuyTicket(), the collectTicketPrice(), the getIthWinningTicket (), or checkIfTicketWon() function), the index of the three winners ticket will be calculated and safed in Array.

E.g. Ticket A,B,C with Index 66, 80, 85 are safed in Array

By this, we can determine the type of the ticket and also calculate the amount of money the winner gets on each ticket.

SUM of Prices = Winner1 amount + winner2 amount + winner3 amount

After the Winners are picked, we have to take SUM of Prices out of the total money Pool, and also from the single specific lottery pool. The left over money in the singele specific lottery has then to be transferred into the next Lottery