Design and analysis

(by Tim Abdiukov, z5214048)

The implementation done within this assignment's is done in accordance to the views on how the requirements could be satisfied within the blockchain environment.

The code is structured as follows. There are the following main contracts ('modules', if you like),

Contract	Description
Concert	Main contract. The majority of code related to the contract placed here
Ticket	The OOP class for manipulating tickets information
RefundRequest	The OOP class for manipulating refund requests information
Tickets_FIFO	The Solidity FIFO queue implementation for Tickets, derived from the code avail. online
RefundRequests_FIFO	The Solidity FIFO queue implementation for Refund Requests, derived from the code avail. online

Within the main Concert contract, within the functions themselves, there has been placed the custom 'decorations' (so to speak) that check, where necessary, the timing and the validity of the concert and its sales, as well as the calling user's status. All that is done in order to prevent malicious attempts on abusing the contract functions, both from the fans, and owner of the concert.

Once sales start, the fans can buy tickets. If the ticket quota is exceeded, then the interested users will be able to buy a place in the FIFO 'rush' tickets waiting list. The rush tickets automatically transfer to the trivial tickets once there are places available. Also one cannot buy a rush ticket if a trivial counterpart is available for security reasons.

If one requests a refund within the sales timeframe, they must provide with the documentation (blob) as well as the description of their situation (desc). The requests gets added to the corresponding FIFO queue where the requests can be reviewed by the owner and decided if valid or not. With that said, if owner himself acts rogue, all the refund requests are refunded automatically. Note that because f the linear lookup time within the FIFO rush tickets queue, the rush tickets cannot be refunded manually, and are automatically refunded as the concert begins, or ends without beginning.

If the concert initiates, the rush tickets are refunded automatically. However, to prevent from the owner's rogue activity, if the concert does not begin at all, all the rush tickets (automatically) as well as the normal tickets (manually, upon the user's request) are refunded without the owner's input. If the owner goes rogue, they cannot withdraw the money.

For the users to attend concert, they must check in.

Below come the possible thought-of bad situations that can happen.

Situation	Response
"Someone stole my ticket/wallet!"	None of our responsibility.
"The owner goes rogue and does not give a concert"	Users can, and should withdraw their money, while the owner cannot
"Someone tries to refund a fraudulent ticket"	(security checks)
(User tries to access functions they are not supposed to access)	the custom 'decorations'