Blockchain Bootcamp Day 7(18 minutes)

* Exception handling in solidity.
* Require statements in solidity are for input validation.require(true condition, else throw this exception message)
* Assert statement in solidity are used for validating variable state(true condition, if true stop execution.).
* Uint64 datatype can only store up to 18 ether.
* Assert checks to make internal state are not violated.
* Throw, require, assert, and revert
* Transactions are atomic, they are state reverting. Everything happened before will be reverted. Does not cascade for low-level functions such as address.send.
* Errors are “state reverting”
* Throw depreciated, require, assert or revert instead.
* In the past, use a if/else, in the else block throw except**i**on,
* Catching is not possible in solidity, you can’t react to a thrown exception.
* Require -> returns remaining gas.(0xfd)
* Assert -> consume all gas(oxfe)
* Both use different byte code and trigger different byte code. The edm will react differently to those.
* In revert operation(require statement) the remaining gas will be returned to user.
* In invalid operation(assert statement) consumes all gas.
* Assert used to invalidate invariants, such as funds missing, something unexpected, check sum doesn’t apply. Does NOT handle user input. Such as variables that don’t wrap around and checks don’t work.
* Require used to validate user input
* How is assert is triggered. When trying to reach a array that out of bound. Dividing by 0.
* How is require is triggered. Doing a function call via a message call but it doesn’t finish properly. Not passing payable to method, or if transfer fails.
* Revert statement is just like require statement but it’s false and in if statement.
* **UINT64 data type can only store a max of 18.4467 ether or 2^64-1 before it automatically rolls over to zero**