Lecture 14 - Start on Solidity

News

1. 14 countries are moving to a blockchain based currency. https://qz.com/1810727/central-banks-are-researching-digital-currencies-to-replace-cash/
2. Mashreq accomplishes this by removing the existing paper-based KYC procedure and replacing it with a blockchain-centric digital one. https://cointelegraph.com/news/dubai-welcomes-global-businesses-with-first-blockchain-kyc-platform

Solidity

Version of the compiler

```
pragma solidity >=0.4.25 <0.7.0;
```

Library we will be using

```
import "openzeppelin-solidity/contracts/token/ERC20/StandardToken.sol";
```

Comments

```
/* Some Comment */ // Single line comment
```

Classes and Inheritance

```
contract StandardToken is ERC20, BasicToken {
```

Constructor

```
    constructor() public {
```

Declare a string

```
    string public constant name = "SimpleToken"; // solium-disable-line uppercase
```

Test for true statments

```
  require(_to != address(0));
```

if

```
  if (_subtractedValue > oldValue) {
    allowed[msg.sender][_spender] = 0;
  } else {
    allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
  }
```

Function

```
function transferFrom( address _from, address _to, uint256 _value
) public returns (bool)
{
```

Events

```
event Approval( address indexed owner, address indexed spender,
  uint256 value );
```

```
  emit Approval(msg.sender, _spender, _value);
```

Example with a Loop

```
pragma solidity >=0.4.25 <0.7.0;

import "openzeppelin-solidity/contracts/ownership/Ownable.sol";

contract Loop is Ownable {
  struct StudentStruct {
    uint256 grade;        // grade is a percentage * 1000
    uint256 pin;        // Unique PIN code for student to get grade
  }

  mapping(address => StudentStruct) addressToStudent;
  address [] studentList;

  event LogStudentGrade(address student, uint256 studentGrade);

  function appendStudentGrade(address student, uint256 studentGrade,
  uint256 pin) public onlyOwner {
    studentList.push(student);
    addressToStudent[student].grade = studentGrade;
    addressToStudent[student].pin = pin;
  }

  function getStudentCount() public view returns(uint) {
    return studentList.length;
  }

  function getNthStudentGrade(uint i)
  public view onlyOwner returns(uint256) {
    require(i >= 0);
    require(i < studentList.length);
    return(addressToStudent[studentList[i]].grade);
  }

  function getNthStudentAddress(uint i)
  public view onlyOwner returns(address) {
    require(i >= 0);
    require(i < studentList.length);
    return(studentList[i]);
  }
```

```
37
38    function getAGrade(address student, uint256 pin)
      public view returns(uint256) {
39      require(msg.sender == address);
40      require(pin == addressToStudent[student].pin);
41      return(addressToStudent[student].grade);
42    }
43
44    function emitGrades () public onlyOwner {
45      for (uint i=0; i<studentList.length; i++) {
46        emit LogStudentGrade(studentList[i],
          addressToStudent[studentList[i]].grade);
47      }
48    }
49  }
```