University of Basel

Group Project: Smart Contracts
Tuition Fee

Cedric Schröter, Philip Steiger und Jonas Löffler 10.12.18

Background & Idea 1/2

1. **Main essence of smart contracts:** Transaction consensus

1. **Disrupt the fixed tuition based university systems**
   - Free entry, no tuition payable up front
   - Tuition payment is based on in-class performance of professors
   - Incentive based payment system
     - 50% goes to professor
     - 40% university
     - 10% best professor
   - We ignore any game theoretical implications

3. **Problem**: Universities may give false information about incoming fees
4. **Solution**: Smart contracts and consensus

Background & Idea 2/2

5. **Students pay tuition through smart contract**
   - Accessible for professors and students
   - Code is law

6. **Course List**
   - `addclass('0x545c2Fbd2eca50dD9510482B57aB05FB709232a5  ', "Bitcoin, Blockchain and Kryptoassets – Fabian Schär");`
   - `addclass('0xd0c5d178a1b4174799eE0E17129B0dE413394903 ', "Blockchain, Consensus Protocols and Smart Contracts – Alexander Berentsen");`

7. **The Code**

Code 1/3

```
Pragma solidity ^0.4.25;
contract TuitionFee {
```

Declare state variables

```
    using SafeMath for uint256;
    uint public endTime;
    address public owner;
    bool public active = true;
    mapping (bytes32 => uint) public CourseListMapping;

    struct CourseList {
        uint amount;
        address addr;
        bytes32 title;
    }

    CourseList[] public CourseList;
```

Define events

```
    modifier notEnded() { require(true == active);
    _;
    }

    modifier onlyOwner() { require(message.sender ==owner);
    _;
    }
```

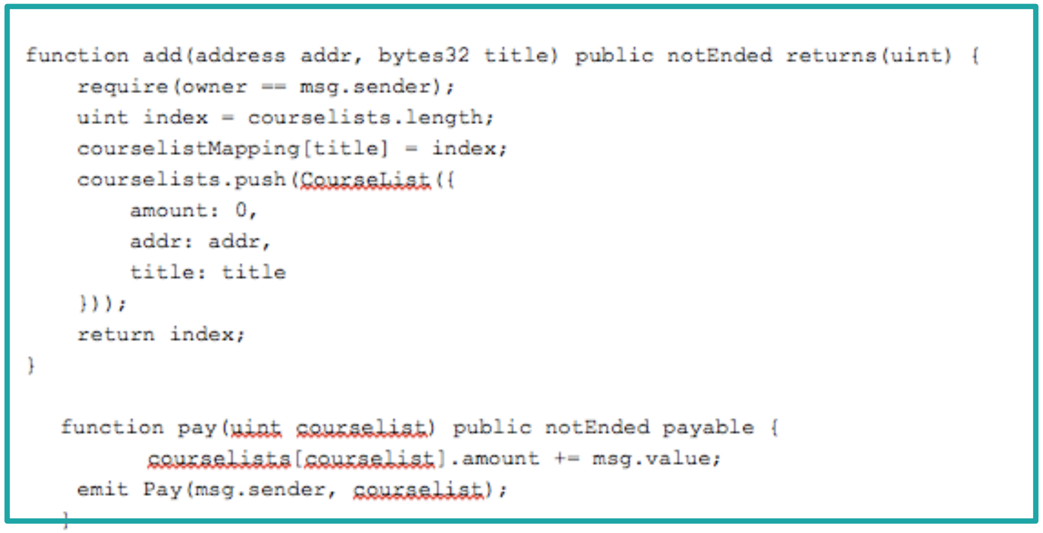
Define modifiers

```
        event Pay(address indexed _from, uint256 indexed _CourseList);
    event Ended();


        constructor(uint end) public {
        endTime = end;
        owner = msg.sender;
        }

    function getCourseListCount() public constant returns(uint) {
        return CourseList.length;
    }
```

Code 2/3

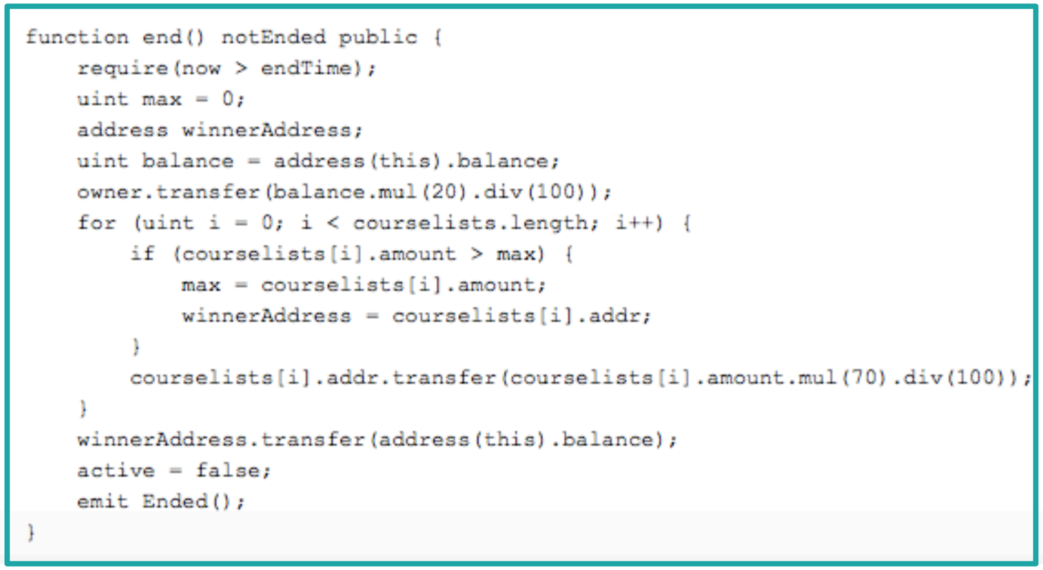
Add Address

Pay



```
function add(address addr, bytes32 title) public notEnded returns(uint) {
    require(owner == msg.sender);
    uint index = courselists.length;
    courselistMapping[title] = index;
    courselists.push(CourseList({
        amount: 0,
        addr: addr,
        title: title
    }));
    return index;
}

   function pay(uint courselist) public notEnded payable {
         courselists[courselist].amount += msg.value;
    emit Pay(msg.sender, courselist);
   }
```

End



```
function end() notEnded public {
    require(now > endTime);
    uint max = 0;
    address winnerAddress;
    uint balance = address(this).balance;
    owner.transfer(balance.mul(20).div(100));
    for (uint i = 0; i < courselists.length; i++) {
        if (courselists[i].amount > max) {
            max = courselists[i].amount;
            winnerAddress = courselists[i].addr;
        }
        courselists[i].addr.transfer(courselists[i].amount.mul(70).div(100));
    }
    winnerAddress.transfer(address(this).balance);
    active = false;
    emit Ended();
}
```

University
of Basel

Thank you
for your attention.