法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

关注 小象学院

区块链编程：Solidity以太坊智能合约

王亮

第四课 solidity编程:智能合约实现

4.18 合约与继承

合约

- Solidity中合约类似面向对象语言中的类。
- 合约可以继承。
- 一个合约可以调用另外一个合约。
- 一个合约中可以创建另外一个合约。
- 合约操作另外一个合约，一般都需要知道其代码。

合约间调用

```
contract OwnedToken {
    // TokenCreator是一个合约类型
    // 未初始化前，为一个引用
    TokenCreator creator;
    address owner;//状态变量
    bytes32 name; //状态变量

    //构造函数
    function OwnedToken(bytes32 _name) public {
        owner = msg.sender;
        creator = TokenCreator(msg.sender);//另外一个合约
        name = _name;
    }
```

合约中创建合约

- 一个合约可以通过new关键字来创建一个合约。
- 要创建合约的完整代码，必须提前知道。

创建合约实例

```
contract TokenCreator {
    function createToken(bytes32 name) public returns (OwnedToken)
    {
        //创建一个新的合约，name为构造函数所需变量
        OwnedToken tokenAddress = new OwnedToken(name);
        return tokenAddress;
    }
```

继承(Inheritance)

- Solidity通过复制包括多态的代码来支持多重继承。基本的继承体系与python类似。
- 当一个合约从多个其它合约那里继承，在区块链上仅会创建一个合约，在父合约里的代码会复制来形成继承合约。
- 派生的合约需要提供所有父合约需要的所有参数。

继承

```
contract Owned {
    function owned() { owner = msg.sender; }
    address owner;
}


// 使用`is` 来继承另外一个合约.
// 子合约可以使用所有的非私有变量，包括内部函数和状态变量
contract Mortal is Owned {
    function kill() {
        if (msg.sender == owner) selfdestruct(owner);
    }
}
```

几种特殊的合约

- 抽象合约（Abstract Contracts）
- 接口（Interfaces）
- 库（Libraries）

抽象合约(Abstract Contracts)

- 合约包含抽象函数，也就是没有函数体的函数。
- 这样的合约不能通过编译，即使合约内也包含一些正常的函数。
- 抽象合约一般可以做为基合约被继承。

抽象合约

```
contract Feline {
    //函数没有函数实现，为抽象函数，对应的合约为抽象合约
    function utterance() returns (bytes32);

    function getContractName() returns (string) {
        return "Feline";
    }
}


contract Cat is Feline {
    //继承抽象合约，实现函数功能
    function utterance() returns (bytes32) { return "miaow"; }
}
```

接口（interface）

- 接口与抽象合约类似，与之不同的是，接口内没有任何函数是已实现的，同时还有如下限制。
- 不能继承其它合约，或接口。
- 不能定义构造器
- 不能定义变量
- 不能定义结构体
- 不能定义枚举类
- 接口基本上限制为合约ABI定义可以表示的内容。

接口

```
//接口
interface Token {
    function transfer(address recipient, uint amount);
}


//接口可以被
contract MyToken is Token {
    function transfer(address recipient, uint amount) {
        //函数实现
    }
}
```

库（Libraries）

- 库与合约类似，但它的目的是在一个指定的地址，且仅部署一次，然后通过EVM的特性来复用代码。
- 使用库合约的合约，可以将库合约视为隐式的父合约(base contracts)，不会显式的出现在继承关系中。
- 调用库函数的方式非常类似，如库L有函数f()，使用L.f()即可访问。

库

```
library Set {
    struct Data { mapping(uint => bool) flags; }

    // 第一个参数的类型为"storage reference"，仅存储地址，而不是
    // 这个是库的特别特征.
    // 按照一般语言的惯例，`self`代表第一个参数
    function insert(Data storage self, uint value)
        public
        returns (bool)
    {
        if (self.flags[value])
            return false; // already there
        self.flags[value] = true;
        return true;
    }
```

```
contract C {
    Set.Data knownValues;

    function register(uint value) public {
        // 库可以直接调用，而无需使用this
        require(Set.insert(knownValues, value));
    }
}
```

流程演示

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：**小象学院**