**实验6 学生管理系统合约**

**【实验介绍】**

通过实验5，我们熟悉了如何使用Python的web3j库调用合约函数，对以太坊的数据进行读写操作，改变合约的状态，并且对Solidity的文件结构、数据类型、控制结构与表达式有了一些初步的理解。本次实验的内容是根据提供的Python测试脚本编写相应的合约，**完成一个模拟的学生管理系统。**

在进行实验操作之前，我们首先要了解典型学生管理系统的系统架构，学习以太坊虚拟机及其提供的事件与日志机制的相关知识。

1. 学生管理系统

学生管理系统是大部分语言的入门类业务系统，总体任务是将学生、院系、教师、课程等相关信息系统化，信息化和规范化，满足学校管理人员对学生各种信息的日常管理。从工程角度上看，完成一个学生管理系统，需要对系统内的各种数据的结构进行定义，并实现对数据的基本操作，即增加、删除、查询和修改。

实际应用的大型信息管理系统还要求实现日志功能，记录用户的操作记录和系统的运行历史。在真实的项目中，在调试、错误定位和数据分析等环节上都离不开日志。日志本质上是程序输出的一行文本信息，一般包含时间点、输出定位以及输出内容。那么没有输出功能的Solidity智能合约能够实现日志功能吗？

1. 事件与日志机制

虽然Solidity智能合约不能像其他语言的print函数那样输出信息到屏幕上，但是以太坊虚拟机EVM（Ethereum Virual Machine）提供了事件与日志机制。事件是能方便地调用EVM日志功能的接口。在以太坊中，开发者能够在合约内部使用event关键字来定义事件，在合约外部调用事件并访问EVM日志来实现一些交互功能，如通知前端UI或返回函数调用结果等。下表1-1为用户向合约捐赠以太币的例子，使用event关键字定义Donate事件，接受两个参数，address类型的sender表示调用donate方法的账户地址，即捐赠者；uint类型的amount参数表示捐赠的金额。在实际应用中，当donate函数被调用时，就会触发Donate事件，开发者可以在上层应用订阅该事件，进行相关操作。

表1-1 案例\_捐赠以太币

|  |
| --- |
| pragma solidity >=0.4.15 <0.9.0;  contract SimpleEvent {  event Donate(address sender, uint amount); // 定义事件  function donate() public payable {  // ...  emit Donate(msg.sender, msg.value); // 触发事件  }  } |

1. EVM

EVM是以太坊智能合约的运行环境，类似于JVM（Java Virtual Machine），EVM旨在提供与操作系统和硬件环境无关的运行环境，保证程序兼容性和结果确定性。EVM层次结构如图6-1所示，自顶向下，第一层，开发人员将智能合约编写、编译为字节码后进行部署；第二层，EVM解释执行上一层的具体合约代码；第三层，以太坊节点，用来作为EVM的载体。第四层，是实际执行机器码的硬件。



图6-1 EVM层次结构

**【实验要求】**

1. 根据测试脚本编写Solidity合约，实现一个学生管理系统
2. 设计合适的数据结构，编写业务代码逻辑，并尽可能优化相关函数的时间复杂度
3. 理解EVM及事件日志机制的工作原理

**【实验准备】**

**1. 环境配置**

本次实验开发环境和运行环境与实验5保持一致。

**2. 导入代码**

解压附件中的实验6，使用vscode打开student文件夹。其中student\_test.py为测试脚本，它会调用web3\_deploy.py来编译部署合约，并调用student\_util.py生成学生信息。

**【实验过程】**

1. **创建合约**

实验代码框架如下图所示，根据红框中的合约名创建合约。在脚本文件web3\_deploy.py中，我们编写了编译和部署的接口。读者编写好合约，启动ganache测试链工具后，就可以使用compile\_and\_deploy函数完成编译与部署，如图6-2红色方框所示。

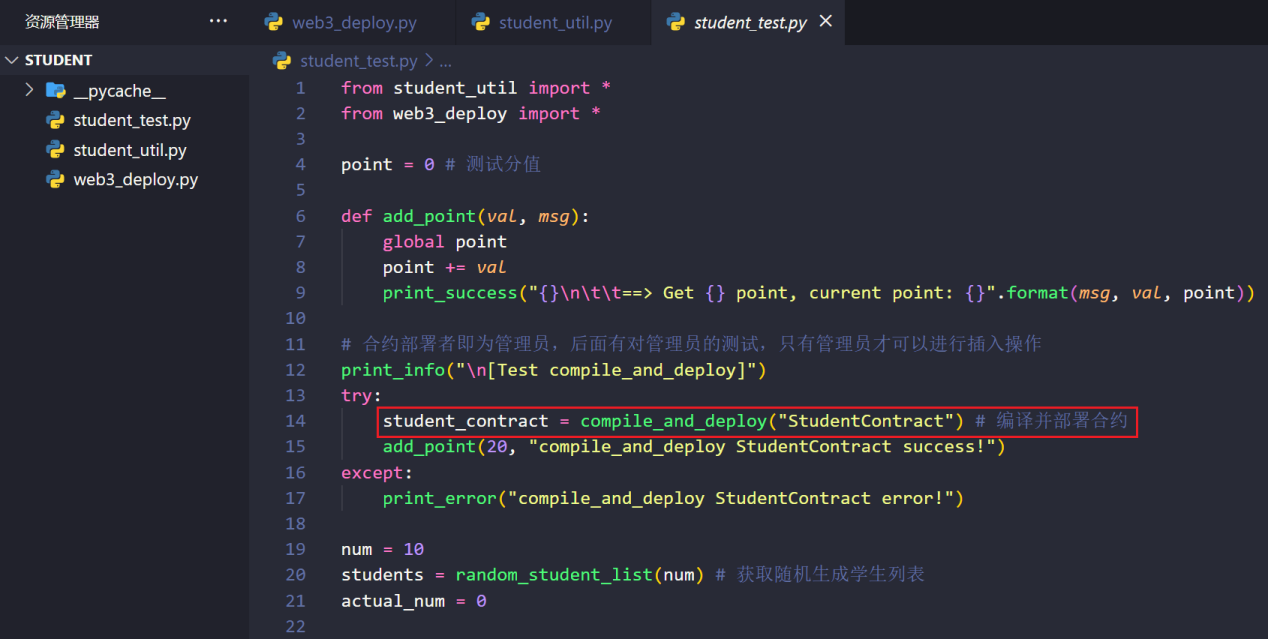


图6-2 代码框架

阅读student\_test.py中的测试代码，了解该合约所需实现的合约函数，如下图6-3白底所示。

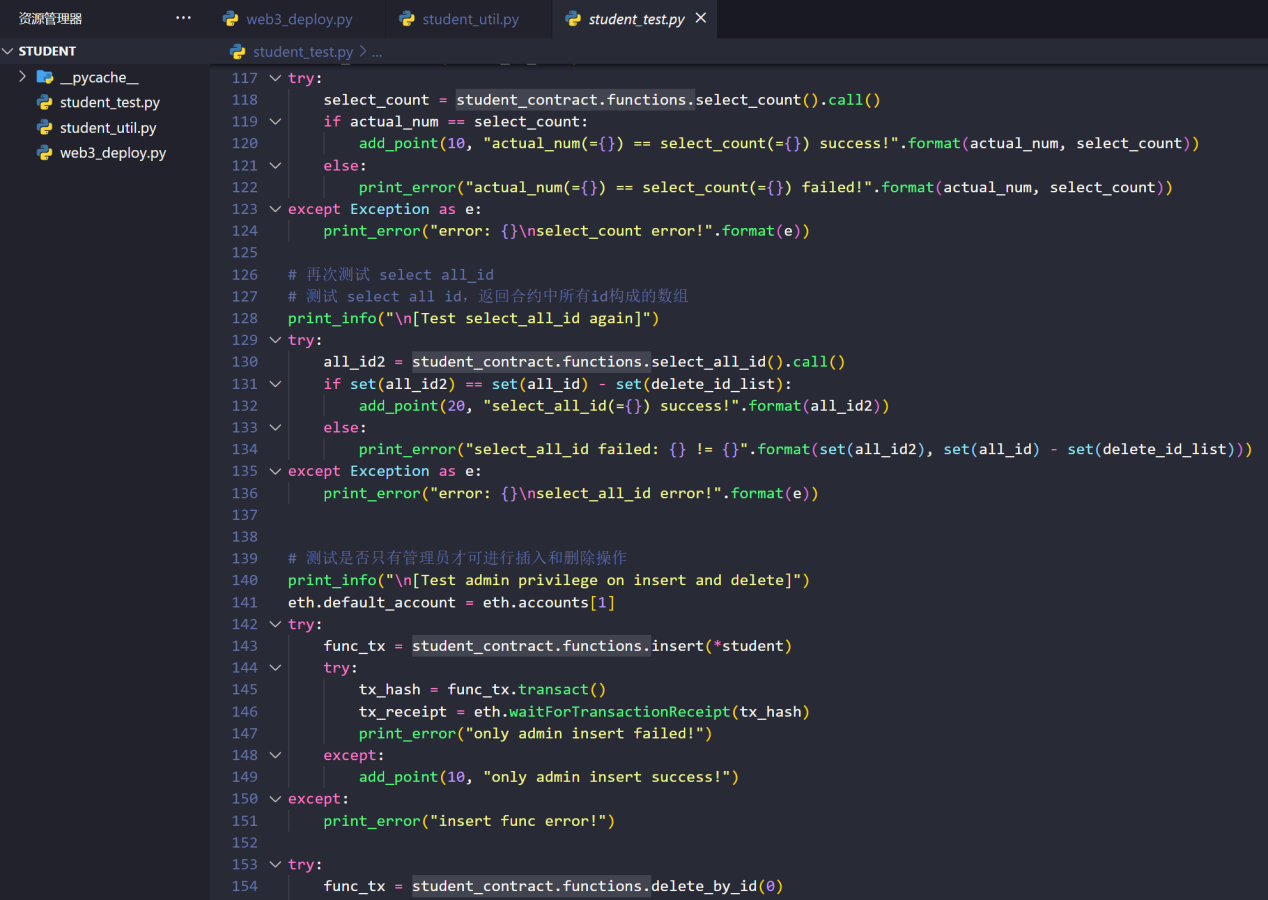


图6-3 测试脚本

表6-2给出了学生管理系统合约的简单框架，同学们也可以根据自己的想法和编码习惯来从零开始编写合约。

表6-2 StudentContract.sol

|  |
| --- |
| // SPDX-License-Identifier: GPL-3.0  pragma solidity >=0.4.15 <0.9.0;  contract StudentContract {  struct Student {  uint256 id;  string name;  string sex;  uint256 age;  string dept;  }  address admin;  constructor() {  admin = msg.sender;  }  function insert(  uint256 \_id,  string memory \_name,  string memory \_sex,  uint256 \_age,  string memory \_dept  ) public {  // TODO:插入一条学生记录  // 插入完成后，触发Insert事件  emit Insert(\_id);  }  event Insert(uint256 id);  function exist\_by\_id(uint256 \_id) public view returns (bool isExist) {  // TODO:查找系统中是否存在某个学号  }  function select\_count() public view returns (uint256 \_count) {  // TODO:查找系统中的学生数量  }  function select\_all\_id() public view returns (uint256[] memory \_ids) {  // TODO:查找系统中所有的学号  }  function select\_id(uint256 \_id) public view returns (Student memory) {  // TODO:查找指定学号的学生信息  }  function delete\_by\_id(uint256 \_id) public {  // TODO:删除指定学号的学生信息  }  } |

1. **运行脚本**

编写好合约后，打开命令行，运行 python3 student\_test.py，运行情况如图6-4、6-5、6-6所示。

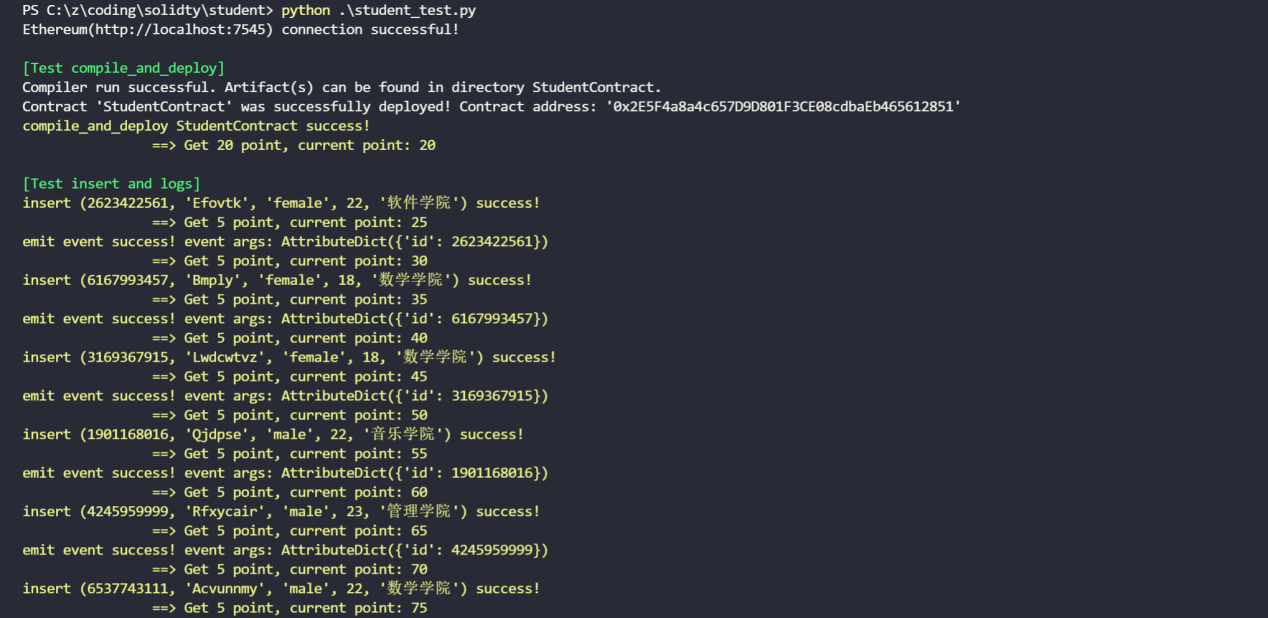


图6-4 测试预期结果1

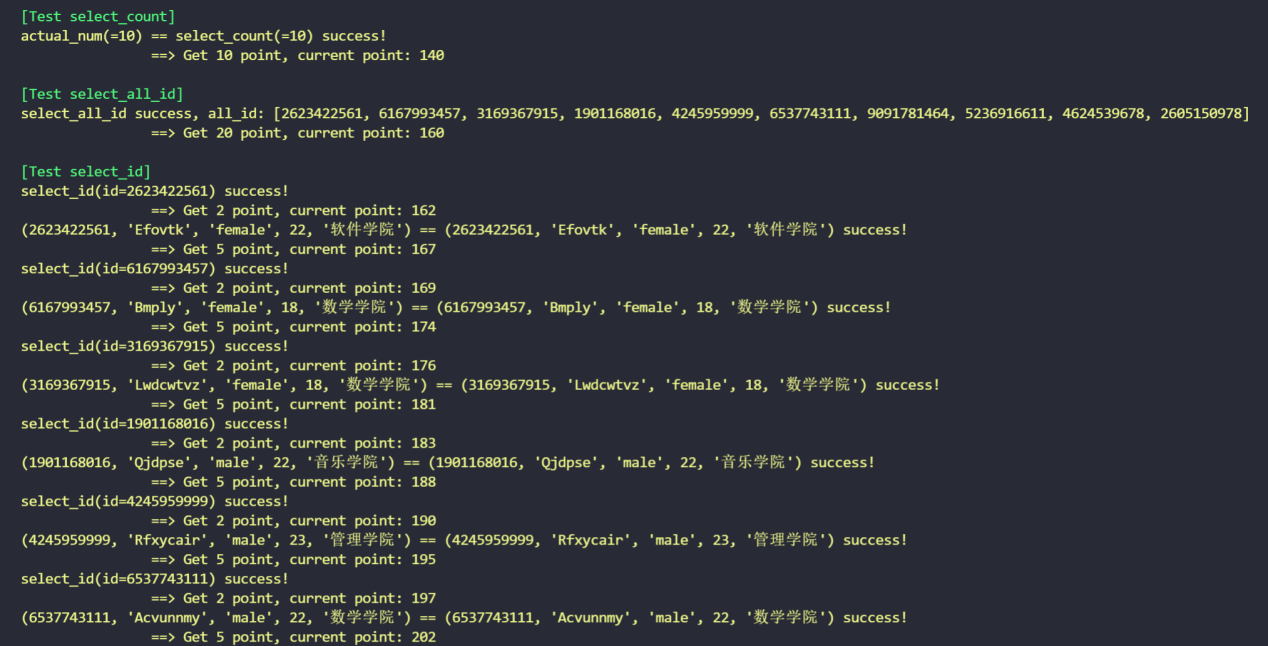


图6-5 测试预期结果2

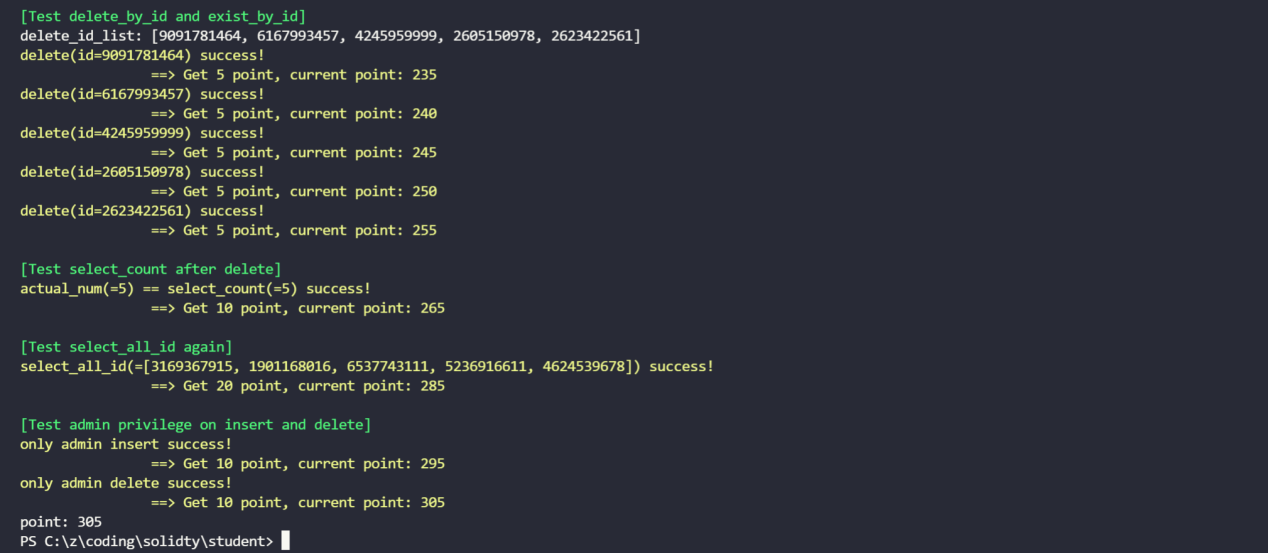


图6-6 测试预期结果3

程序出现错误会打印红色信息，如图6-7所示，可用于诊断和定位bug，因为基本的错误都使用异常进行了捕获，所以读者可以在完成合约的任一函数时进行测试。

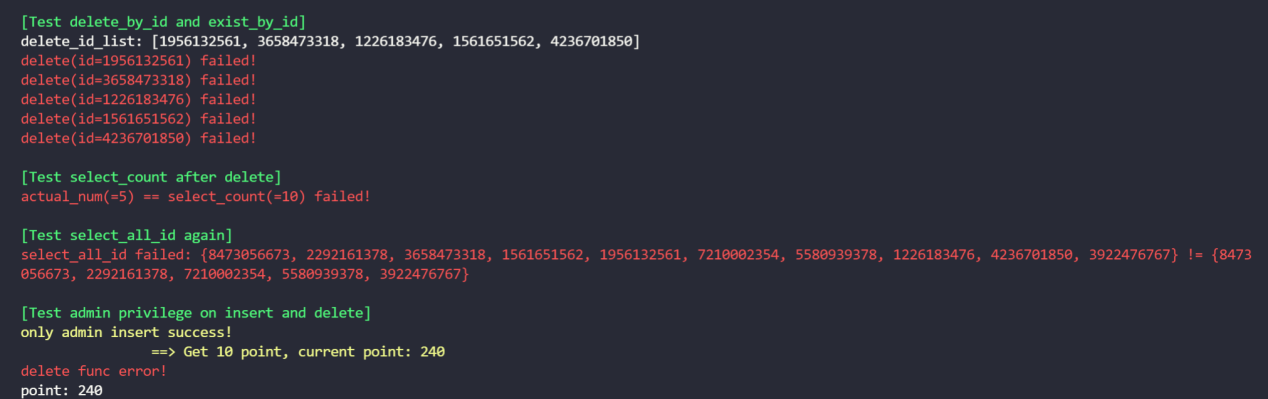


图6-7 错误样例

**【实验小结】**

本实验以经典学生管理系统为例，论述了一般业务系统的总体任务、系统架构以及功能模块。同时，对日志模块进行了简要的阐述，引入事件日志机制并讲解其原理，以深入了解Solidity和EVM。之后，给出了捐赠以太币的合约案例来展示如何在代码中定义并使用Solidity事件。通过本实验的理论学习与工程实践，读者应该对以太坊及Solidity智能合约有了更深入的理解。

**【习题】**

1. 在Solidity中，Event指的是什么？有什么作用？

2. 在实验完成的基础上，实现函数get\_id\_by\_min\_age()，功能为查找系统中年龄最小的学生学号（相同年龄的情况不考虑），并在测试脚本中加以验证。

3. 在实验完成的基础上，实现函数update\_dept\_by\_id(uint256 \_id,string memory \_dept)，功能为修改指定学号学生的院系。要求抛出Update事件，并在测试脚本中加以验证。

**【参考文献】**

1. 欧阳丽炜，王帅，袁勇等.智能合约：架构及进展[J].自动化学报,2019,45(03):445-457.
2. 闫莺. 以太坊技术详解与实战[M]. 机械工业出版社, 2018.
3. Buterin V. A next-generation smart contract and decentralized application platform[J]. white paper, 2014, 3(37).
4. Cong L W, He Z. Blockchain disruption and smart contracts[J]. The Review of Financial Studies, 2019, 32(5): 1754-1797.
5. Solidity Community. Solidity(OL). (2020-10-27) [2020-12-11]. <https://github.com/ethereum/solidity/tags>.
6. Ethereum Community. Ethereum(OL). (2020-06-08) [2020-07-12]. <https://github.com/ethereum/go-ethereum>.