Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Лабораторна робота №2

Програмування смарт-контрактів

Тема: Інтерфейси в мові Solidity

Виконав
студент групи ІП-11:
Панченко С. В.

Перевірив:
Яланецький В.А.

Київ 2025

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Ознайомитися з інтерфейсами, реалізувати функціонал інтерфейсів задля взаємодії з методами смар-контракту.

2 ЗАВДАННЯ

Створити інтерфейси смарт-контракту згідно індивідуального варіанту, протестувати його роботу.

1. Додати до першого смарт-контракту відповідний інтерфейс.
2. Створити новий смарт-контракт і викликати методи першого смарт-контракту.

3 ВИКОНАННЯ

Для виконання лабораторної написав інтерфейс IPanchenko та контракт PanchenkoCaller, що приймає в конструктур адресу Panchenko, а далі кастить її до IPanchenko.

```
interface IPanchenko {
    function PanchenkoSerhiiVitaliyovichContribute() payable external;
    function PanchenkoSerhiiVitaliyovichGetPrice(string memory priceString) external pure returns (uint256);
    function PanchenkoSerhiiVitaliyovichGetWeather(int8 temperature) external pure returns (string memory);
    function PanchenkoSerhiiVitaliyovichWithdraw() external;
    function PanchenkoSerhiiVitaliyovichGetBalance() external view returns (uint256);
}

contract PanchenkoCaller {
    IPanchenko private _panchenko;

    constructor(address panchenko) {
        _panchenko = IPanchenko(panchenko);
    }

    function doWork() external payable {
        _panchenko.PanchenkoSerhiiVitaliyovichContribute{value: msg.value}();
        _panchenko.PanchenkoSerhiiVitaliyovichGetWeather(25);

_panchenko.PanchenkoSerhiiVitaliyovichGetPrice("ETH=123456$");
    }
}
```

Створимо об’єкт Panchenko та 3 об’єкти PanchenkoCaller, викличимо в кожного об’єкта PanchenkoCaller метод doWork з 6 одиницями ефіру, далі викличемо в Panchenko PanchenkoSerhiiVitaliyovichWithdraw на рисунку 3.1.

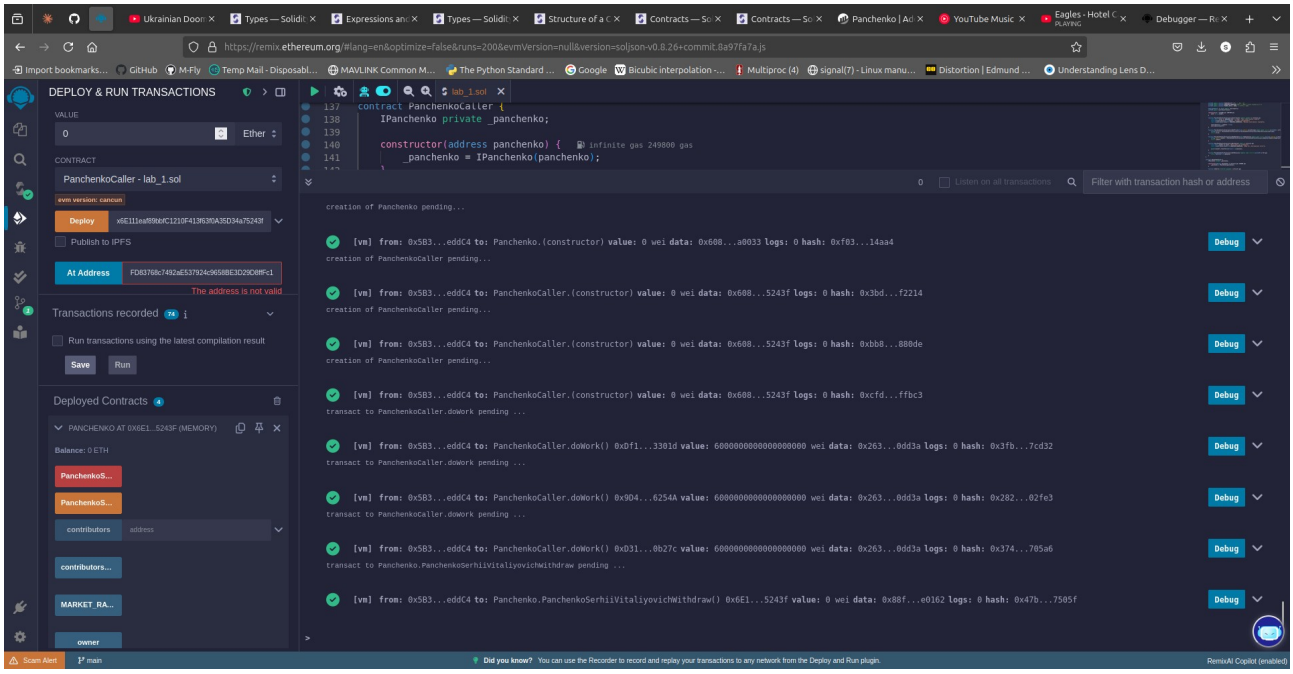

Рисунок 3.1 — виконання операцій з інтерфейсом IPanchenko

[vm]

from: 0x5B3...eddC4

to: Panchenko.(constructor)

value: 0 wei

data: 0x608...a0033

logs: 0

hash: 0xf03...14aa4

creation of PanchenkoCaller pending...

[vm]

from: 0x5B3...eddC4

to: PanchenkoCaller.(constructor)

value: 0 wei

data: 0x608...5243f

logs: 0

hash: 0x3bd...f2214

---

creation of PanchenkoCaller pending...

[vm]

from: 0x5B3...eddC4

to: PanchenkoCaller.(constructor)

value: 0 wei

data: 0x608...5243f

logs: 0

hash: 0xbb8...880de

creation of PanchenkoCaller pending...

[vm]

from: 0x5B3...eddC4

to: PanchenkoCaller.(constructor)

value: 0 wei

data: 0x608...5243f

logs: 0

hash: 0xcfd...ffbc3

transact to PanchenkoCaller.doWork pending ...

[vm]

from: 0x5B3...eddC4

to: PanchenkoCaller.doWork() 0xDf1...3301d

value: 6000000000000000000 wei

data: 0x263...0dd3a

logs: 0

hash: 0x3fb...7cd32

transact to PanchenkoCaller.doWork pending ...

[vm]

from: 0x5B3...eddC4

to: PanchenkoCaller.doWork() 0x9D4...6254A

value: 6000000000000000000 wei

data: 0x263...0dd3a

logs: 0

hash: 0x282...02fe3

transact to PanchenkoCaller.doWork pending ...

[vm]

from: 0x5B3...eddC4

to: PanchenkoCaller.doWork() 0xD31...0b27c

value: 6000000000000000000 wei

data: 0x263...0dd3a

logs: 0

hash: 0x374...705a6

transact to Panchenko.PanchenkoSerhiiVitaliyovichWithdraw pending ...

[vm]

from: 0x5B3...eddC4

to: Panchenko.PanchenkoSerhiiVitaliyovichWithdraw() 0x6E1...5243f

value: 0 wei

data: 0x88f...e0162

logs: 0

hash: 0x47b...7505f

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду

(Найменування програми (документа))

Жорсткий диск

(Вид носія даних)

(Обсяг програми (документа), арк.)

Студента групи ІП-11 4 курсу

Панченка С. В

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

// Library named after the full name (PIP)
library PanchenkoSerhiiVitaliyovich {
    // Function to parse ETH price from a string
    function PanchenkoSerhiiVitaliyovichParsePrice(string memory priceString) internal pure returns (uint256) {
        bytes memory priceBytes = bytes(priceString);

        // Find the position of '='
        uint256 equalPosition = 0;
        for (uint256 i = 0; i < priceBytes.length; i++) {
            if (priceBytes[i] == '=') {
                equalPosition = i + 1;
                break;
            }
        }

        // Parse the number
        uint256 price = 0;
        for (uint256 i = equalPosition; i < priceBytes.length; i++) {
            if (priceBytes[i] >= '0' && priceBytes[i] <= '9') {
                price = price * 10 + uint256(uint8(priceBytes[i]) - 48);
            }
            if (priceBytes[i] == '$') {
                break;
            }
        }

        return price;
    }

    // Function to generate weather comment
    function PanchenkoSerhiiVitaliyovichWeatherComment(int8
```

```
temperature) internal pure returns (string memory) {
        string memory comment;

        if (temperature < 0) {
            comment = "cold: ";
        } else if (temperature < 15) {
            comment = "cool: ";
        } else if (temperature < 25) {
            comment = "warm: ";
        } else {
            comment = "hot: ";
        }

        // Convert temperature to string
        string memory tempStr =
PanchenkoSerhiiVitaliyovichIntToString(temperature);

        return string(abi.encodePacked(comment, tempStr, " C"));
    }

    function PanchenkoSerhiiVitaliyovichIntToString(int8 value)
internal pure returns (string memory) {
        if (value == 0) {
            return "0";
        }
        uint8 digits = 0;
        {
            int16 tempVal = value;
            while (tempVal != 0) {
                digits++;
                tempVal /= 10;
            }
        }
        bool isNegative = value < 0;
        bytes memory buffer = new bytes(isNegative ? digits + 1 :
digits);
        if (isNegative) {
```

```
            buffer[0] = '-';
        }
        int16 index = int16(int256(buffer.length) - 1);
        {
            uint8 tempVal = uint8(isNegative ? -value : value);
            while (tempVal != 0) {
                buffer[uint16(index)] = bytes1(uint8(48 + (tempVal
% 10)));
                index--;
                tempVal /= 10;
            }
        }
        return string(buffer);
    }
}

interface IPanchenko {
    function PanchenkoSerhiiVitaliyovichContribute() payable
external;
    function PanchenkoSerhiiVitaliyovichGetPrice(string memory
priceString) external pure returns (uint256);
    function PanchenkoSerhiiVitaliyovichGetWeather(int8
temperature) external pure returns (string memory);
    function PanchenkoSerhiiVitaliyovichWithdraw() external;
    function PanchenkoSerhiiVitaliyovichGetBalance() external view
returns (uint256);
}

contract Panchenko is IPanchenko {
    using PanchenkoSerhiiVitaliyovich for *;

    address public owner;
    uint256 public constant REQUIRED_ETH = 6; // [DD] = 06
    uint256 public constant REQUIRED_ADDRESSES = 3; // [MM]/2 =
03/2 rounded up to 3
    uint256 public constant MARKET_RATE = 2004; // [YYYY] = 2004
```

```
    mapping(address => bool) public contributors;
    uint256 public contributorsCount;

    constructor() {
        owner = msg.sender;
    }

    function PanchenkoSerhiiVitaliyovichContribute() public
payable {
        require(msg.value == REQUIRED_ETH * 1 ether, "Must send
exactly 6 ETH");
        require(!contributors[msg.sender], "Already contributed");
        require(contributorsCount < REQUIRED_ADDRESSES, "Maximum
contributors reached");

        contributors[msg.sender] = true;
        contributorsCount++;
    }

    function PanchenkoSerhiiVitaliyovichGetPrice(string memory
priceString) public pure returns (uint256) {
        uint256 price =
PanchenkoSerhiiVitaliyovich.PanchenkoSerhiiVitaliyovichParsePrice(
priceString);
        return price;
    }

    function PanchenkoSerhiiVitaliyovichGetWeather(int8
temperature) public pure returns (string memory) {
        string memory comment =
PanchenkoSerhiiVitaliyovich.PanchenkoSerhiiVitaliyovichWeatherComm
ent(temperature);
        return comment;
    }

    function PanchenkoSerhiiVitaliyovichWithdraw() public {
        require(msg.sender == owner, "Only owner can withdraw");
```

```
        require(contributorsCount == REQUIRED_ADDRESSES, "Need all
contributions first");

        payable(owner).transfer(address(this).balance);
    }

    function PanchenkoSerhiiVitaliyovichGetBalance() public view
returns (uint256) {
        return address(this).balance;
    }
}

contract PanchenkoCaller {
    IPanchenko private _panchenko;

    constructor(address panchenko) {
        _panchenko = IPanchenko(panchenko);
    }

    function doWork() external payable {
        _panchenko.PanchenkoSerhiiVitaliyovichContribute{value:
msg.value}();
        _panchenko.PanchenkoSerhiiVitaliyovichGetWeather(25);

_panchenko.PanchenkoSerhiiVitaliyovichGetPrice("ETH=123456$");
    }
}
```