Lesson 12

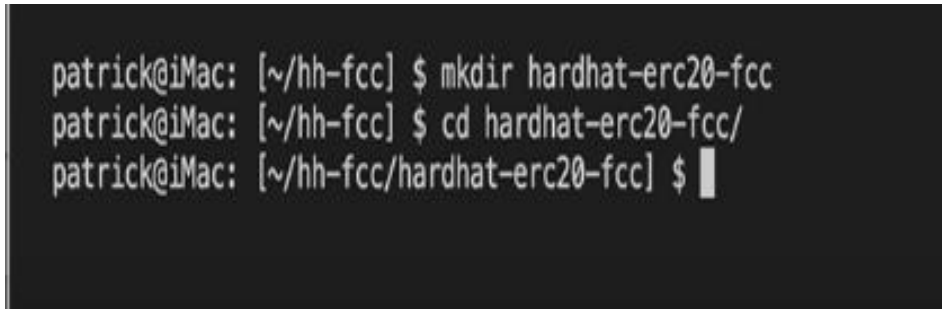
HARDHAT ERC20S

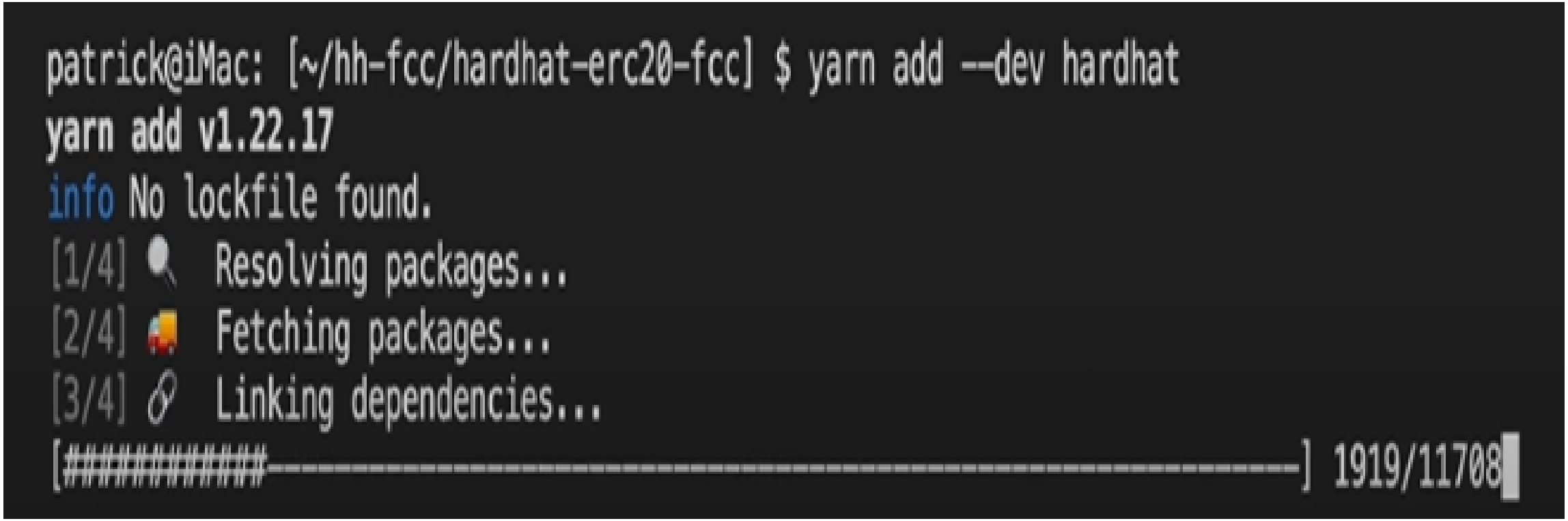
Hikmah Nisya - 1103184094
Radzis Araaf Jaya Jamaludin - 1103184234
Raudhatul Rafiqah Assyahiddini - 1103180225

Pertama harus membuat directory terlebih dahulu,

Disini kita menggunakan “hardhat-erc20-fcc” , lalu

File tersebut akan di buka dengan”cd(nama dicertorynya)”





Selanjutnya akan membuat project hardhat nya,
dengan cara “yarn add –dev hardhat” lalu enter

Setelah itu buka hardhat yang sebelumnya di buat lalu install hardhat nya dan akan muncul tampilan seperti ini,

Lalu pilih “create an empty hardhat.config.js” maka akan secara otomati terbentuk “hardhat.config.js”

```
contract ManualToken {
    uint256 initialSupply;
    mapping(address => uint256) public balanceOf;
    mapping(address => mapping(address => uint256)) public allowance;
```

contracts > ManualToken.sol

```
1   // SPDX-License-Identifier: MIT
2   pragma solidity ^0.8.7;
3
4   contract ManualToken {
5
6
7
8   }
9
```

Selanjutnya kita membuat “ManualToken.Sol”, lalu buat awalan seperti di ini, dan kita akan membuat mapping untuk untuk membalancekan dari seluruh orang yang menggunakan, dan seberapa banyak yang mereka punya, mapping berikutnya di gunakan untuk mengizinkan penggunakan untuk mentranfer, dan tambahkan initialsuply yaitu “uint256” sebagai inisialdari token yang di miliki,

```
function transfer(
    address from,
    address to,
    uint256 amount
) public {
    balanceOf[from] = balanceOf[from] - amount;
    balanceOf[to] += amount;
}
```

```
function transferFrom(
    address _from,
    address _to,
    uint256 _value
) public returns (bool success) {
    require(_value <= allowance[_from][msg.sender]);
    allowance[_from][msg.sender] -= _value;
    _transfer(_from, _to, _value);
    return true;
}
```

langkah selanjutnya adalah membuat function "tranfer" dan “tranferfrom”, seperti di gambar, lalu" TransferFrom" di gunakan untuk fucntion mengiring dan function "tranfer" di gunakan untuk membalance data yang di terima

```
    uint256 _value
) internal {
    require(_to != address(0x0));
    require(balanceOf[_from] >= _value);
    require(balanceOf[_to] + _value >= balanceOf[_to]);
    uint256 previousBalances = balanceOf[_from] + balanceOf[_to];
    balanceOf[_from] -= _value;
    balanceOf[_to] += _value;
    emit Transfer(_from, _to, _value);
    assert(balanceOf[_from] + balanceOf[_to] == previousBalances);
}


function transfer(address _to, uint256 _value) public returns (bool success) {
    _transfer(msg.sender, _to, _value);
    return true;
}

function transferFrom(
    address _from,
    address _to,
    uint256 _value
) public returns (bool success) {
    require(_value <= allowance[_from][msg.sender]); // Check allowance
    allowance[_from][msg.sender] -= _value;
    _transfer(_from, _to, _value);
    return true;
}

function approve(address _spender, uint256 _value)
    public
    returns (bool success)
{
    allowance[msg.sender][_spender] = _value;
    emit Approval(msg.sender, _spender, _value);
    return true;
}

function approveAndCall(
    address _spender,
    uint256 _value,
    bytes memory _extraData
) public returns (bool success) {
    tokenRecipient spender = tokenRecipient(_spender);
    if (approve(_spender, _value)) {
        spender.receiveApproval(msg.sender, _value, address(this), _extraData);
        return true;
    }
}
```

langkah selanjutnya adalah menambahkan function “approve”, Approval”,dsb yang tersedia dalam EIP-20, atau kita biasa mengcopy langsung pada Code yang disediakan melalui github, Hasilnya akan seperti gambar di samping

```
patrick@iMac: [~/hh-fcc/hardhat-erc20-fcc] $ yarn add --dev @openzeppelin/contracts
yarn add v1.22.17
warning package.json: No license field
warning No license field
[1/4] 🔍  Resolving packages...
[2/4] 🚚  Fetching packages...
[##############################################################----------] 253/293
```

```
1   // SPDX-License-Identifier: MIT
2
3   pragma solidity ^0.8.7;
4
5   import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
6
7   contract OurToken is ERC20 {
8
9   }
10
```

Selanjutnya membuat “OutToken.sol”, dengan cara membuat file baru lalu buat contract dengan nama “ OutToken.sol” seperti membuat “ManualToken.sol”.

Lalu hubungkan visualstudio code dengan openzippelin contracts, dengan cara seperti gambar. Dengan cara ini bisa mempercepat waktu pengerjaan.

Setelah itu import openzeppelin tersebut ke dalam “outToken.sol” dengan cara seperti ini.

```
contract OurToken is ERC20 {
    // initial supply is 50 <- 50 WEI
    // initial supply 50e18
    // 50 * 10**18
    constructor(uint256 initialSupply) ERC20("OurToken", "OT") {
        _mint(msg.sender, initialSupply);
    }
}
```

Maka selanjutnya adalah membuat inisial dari token dan juga symbol seperti gambar di samping,

Dengan ini Hardhat ERC20S Selesai