```
pragma solidity ^0.4.23;

contract Bank {
    // 此合約的擁有者
    address private owner;
    uint256 private contractValue;//儲存定存金額用的變數
    uint256 private contractPeriods;//儲存定存期數用的變數
    // 儲存所有會員的餘額
    mapping (address => uint256) private balance;

    // 事件們，用於通知前端 web3.js
    event DepositEvent(address indexed from, uint256 value,
uint256 timestamp);
    event WithdrawEvent(address indexed from, uint256 value,
uint256 timestamp);
    event TransferEvent(address indexed from, address indexed to,
uint256 value, uint256 timestamp);

    event contractExpiredEvent(address indexed from, uint256
timestamp);
    event buyContractDepositEvent(address indexed from, uint256
value, uint256 periods, uint256 timestamp);
    event terminateContractEvent(address indexed from, uint256
periods, uint256 timestamp);

    modifier isOwner() {
        require(owner == msg.sender, "you are not owner");
        _;
    }

    // 建構子
    constructor() public payable {
        owner = msg.sender;
    }

    // 存錢
    function deposit() public payable {
        balance[msg.sender] += msg.value;

        emit DepositEvent(msg.sender, msg.value, now);
}

    // 提錢
    function withdraw(uint256 etherValue) public {
        uint256 weiValue = etherValue * 1 ether;

        require(balance[msg.sender] >= weiValue, "your balances
are not enough");
```

```
        msg.sender.transfer(weiValue);

        balance[msg.sender] -= weiValue;

        emit WithdrawEvent(msg.sender, etherValue, now);
    }

    // 轉帳
    function transfer(address to, uint256 etherValue) public {
        uint256 weiValue = etherValue * 1 ether;

        require(balance[msg.sender] >= weiValue, "your balances
are not enough");

        balance[msg.sender] -= weiValue;
        balance[to] += weiValue;

        emit TransferEvent(msg.sender, to, etherValue, now);
    }

/*// 購買定存
etherValue、periods為呼叫的參數，依照etherValue值決定要定存多少錢，決定要
定存多久periods，設置require判斷餘額夠不夠和有沒有已經買定存了，若不夠則拋
出異常，並設置buyContractDepositEvent監聽這個function
*/
    function buyContractDeposit(uint256 etherValue,uint256
periods) public {
        uint256 weiValue = etherValue * 1 ether;
        require(contractPeriods == 0, "You already have a Contract
Deposit");
        require(balance[msg.sender] >= weiValue, "your balances
are not enough");
        contractValue = weiValue;
        contractPeriods = periods;
        balance[msg.sender] -= weiValue;


        emit buyContractDepositEvent(msg.sender, etherValue,
periods, now);
    }
    //
```

```
/*//合約期滿
設置require判斷是否有定存金額與期數，若無有定存金額與期數則拋出異常，並設置
contractExpiredEvent監聽這個function
並將銀行內的金額加上原來金額和利息
*/

    function contractExpired() public {

        require(contractValue > 0, "No money contract deposit");
        require(contractPeriods > 0, "period <= 0");

        balance[msg.sender] += contractValue + contractValue *
contractPeriods /100;
        contractValue = 0;
        contractPeriods = 0;

        emit contractExpiredEvent(msg.sender, now);
}
/*//提前解約
設置require判斷是否有定存金額與期數和是否提前結束期數大於原本期數，若無金額與
期數或提前結束期數大於原本期數則拋出異常，並設置terminateContractEvent監聽
這個function
並將銀行內的金額加上原來金額和應得利息
*/

    function terminateContract(uint256 periods) public {

        require(contractValue > 0, "No money contract deposit");
        require(periods > 0, "period <= 0");
        require(contractPeriods > periods, "period >=
contractPeriods");

        balance[msg.sender] += contractValue + contractValue *
periods /100;
        contractValue = 0;
        contractPeriods = 0;

        emit terminateContractEvent(msg.sender, periods, now);
    }

    // 檢查銀行帳戶餘額
    function getBankBalance() public view returns (uint256) {
        return balance[msg.sender];
    }

    function kill() public isOwner {
        selfdestruct(owner);
```

```
}
}
```