```
pragma
solidity
^0.6.6;


            contract BankContract {

                struct client_account{
                    int client_id;
                    address client_address;
                    uint client_balance_in_ether;
                }

                client_account[] clients;

                int clientCounter;
                address payable manager;
                mapping(address => uint) public interestDate;

                modifier onlyManager() {
                    require(msg.sender == manager, "Only
            manager can call this!");
                    _;
                }

                modifier onlyClients() {
                    bool isclient = false;
                    for(uint i=0;i<clients.length;i++){
                        if(clients[i].client_address ==
            msg.sender){
                            isclient = true;
                            break;
                        }
                    }
                    require(isclient, "Only clients can call
            this!");
                    _;
                }
```

```
    constructor() public{
        clientCounter = 0;
    }

    receive() external payable { }

    function setManager(address managerAddress)
public returns(string memory){
        manager = payable(managerAddress);
        return "";
    }

    function joinAsClient() public payable
returns(string memory){
        interestDate[msg.sender] = now;
clients.push(client_account(clientCounter++,
msg.sender, address(msg.sender).balance));
        return "";
    }

    function deposit() public payable onlyClients{
payable(address(this)).transfer(msg.value);
    }

    function withdraw(uint amount) public payable
onlyClients{
        msg.sender.transfer(amount * 1 ether);
    }

    function sendInterest() public payable
onlyManager{
        for(uint i=0;i<clients.length;i++){
            address initialAddress =
clients[i].client_address;
            uint lastInterestDate =
interestDate[initialAddress];
            if(now < lastInterestDate + 10
seconds){
```

```
                revert("It's just been less than
10 seconds!");
            }
            payable(initialAddress).transfer(1
ether);
            interestDate[initialAddress] = now;
        }
    }

    function getContractBalance() public view
returns(uint){
        return address(this).balance;
    }
}
```