contracts/ej_6.sol

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

contract Loteria {
    address public owner;
```

Para que drawWinner lo corra el "owner", ahora tiene que ser definido.

```solidity
    uint256 public ticketPrice;
    uint256 public totalTickets;
    uint256 public ticketsSold;
    mapping(address => uint256) public ticketsBought;
    address[] public participants;
```

Ahora conviene llevar un segundo registro de los participantes. (Queremos evitar iterar sobre el mapping a la hora de escribir drawWinner

```solidity
    modifier onlyOwner() {
        require(msg.sender == owner, "Only owner can call this function");
        _;
    }
```

el modifier y constuctor ahora los necesitamos para dar exclusividad al Owner de correr la función drawWinner

```solidity
    constructor() {
        owner = msg.sender;
    }

    function buyTickets(uint256 numberOfTickets) public payable {
        require(numberOfTickets > 0, "Must buy at least one ticket");
        require(ticketsBought[msg.sender] + numberOfTickets <= totalTickets, "Not enough tickets available");
        require(msg.value == ticketPrice * numberOfTickets, "Incorrect amount sent");

        if (ticketsBought[msg.sender] == 0) {
            participants.push(msg.sender);
        }
```

Escanea la cantidad de ticketsBought de la dirección, si es 0, lo agrega a "participants"

```solidity

        ticketsBought[msg.sender] += numberOfTickets;
        ticketsSold += numberOfTickets;
    }

    function getMyTickets() public view returns (uint256) {
        return ticketsBought[msg.sender];
    }

    function getTotalTicketsSold() public view returns (uint256) {
        return ticketsSold;
    }

    function drawWinner() public onlyOwner {
        require(ticketsSold > 0, "No tickets sold yet");
        require(participants.length > 0, "No participants");
```

Requisitos para no correr la función con errores conceptuales

```solidity

        // Generate a random index using block data
        uint256 randomIndex = uint256(keccak256(abi.encodePacked(
            block.timestamp,
            block.prevrandao,
            block.number
```

Generador de Número Random

```
        ))) % participants.length;

        address winner = participants[randomIndex];
        uint256 prize = address(this).balance;

        // Reset the lottery
        for (uint256 i = 0; i < participants.length; i++) {
            ticketsBought[participants[i]] = 0;
        }
        delete participants;
        ticketsSold = 0;

        // Transfer the prize to the winner
        (bool success, ) = winner.call{value: prize}("");
        require(success, "Transfer failed");
    }
}
```

Selección del Ganador

Marcamos el valorque recibirá el ganador

Opcional (y quizá poco recomendable)
Transferimos al ganador