# **Smart Contracts - Aposta Par-Ímpar**

Aluno: Danilo Augusto Nunes - 11611BCC021

Na implementação da atividade foi utilizado o Brownie, além disso, a biblioteca de "Commit" disponibilizada pelo professor.

O jogo funciona da seguinte forma, tanto o dono quanto o jogador apostam com um número, que ficará escondido no contrato pelo protocolo de commit, depois de ambos jogarem, os números são revelados, após serem somados, através do resto por 2, são classificados em par ou ímpar, dessa forma, resultado 0 garante que o dono do contrato venceu, consequentemente, o jogador ganha com resultado 1.

A punição do jogo é para o participante que não releva o commit dentro do limite estipulado, no caso, uma quantidade de blocos definida. Sendo assim, o outro participante vence e fica com o dinheiro. Além disso, caso apenas um revele, e mesmo assim, esse valor não esteja correto, ele vence a aposta, já que o outro participante foi punido por não revelar. Caso os dois não revelem, o dinheiro morre no contrato.

# **Contrato: Parimpar**

```
pragma solidity >=0.4.25 <0.6.0;
import "./SimpleCommit.sol";
contract parImpar {
    using SimpleCommit for SimpleCommit.CommitType;
    enum StatesType {aguardaJogador, revelaValores, revelado, fim}
    SimpleCommit.CommitType valorDono;
    SimpleCommit.CommitType valorJogador;
    address payable dono;
    address payable jogador;
    uint256 valorAposta;
    uint256 limiteBloco;
    StatesType myState;
    // saber se os jogadors já reveleram o commit
    bool donoRevealed = false;
    bool jogadorRevealed = false;</pre>
```

```
constructor (bytes32 vD) public payable {
    valorDono.commit( vD);
    valorAposta = msg.value;
    limiteBloco = block.number + 10;
    dono = msg.sender;
    myState = StatesType.aguardaJogador;
function entraJogo(bytes32 _vJ) public payable {
  // Dono precisa ter iniciado o jogo
  require (myState == StatesType.aguardaJogador);
   require (msg.value >= valorAposta, "Aposta tem valor minimo!!!");
  // atualiza estado do contrato
  valorJogador.commit( vJ);
  jogador = msg.sender;
  // armazena vJ e msg.sender
   myState = StatesType.revelaValores;
function donoRevela(byte value, bytes32 nonce) public {
  require (msg.sender == dono);
  require (myState == StatesType.revelaValores);
  valorDono.reveal(nonce, value);
  donoRevealed = true;
function jogadorRevela(byte value, bytes32 nonce) public {
  require (msg.sender == jogador);
  require (myState == StatesType.revelaValores);
  valorJogador.reveal(nonce,value);
  jogadorRevealed = true;
function pagaVencedor() public {
  // jogadores devem pelo menos ter iniciado o jogo (ambos)
  // Não necessariamente terem revelado o commit
  require (myState == StatesType.revelaValores);
```

```
// bloco passou do limite
if (block.number > limiteBloco) {
  // Encerra o jogo
 myState = StatesType.fim;
 // jogador não revelou
 if (donoRevealed && !jogadorRevealed) {
    dono.transfer(address(this).balance);
  // dono não revelou
  else if (!donoRevealed && jogadorRevealed) {
    jogador.transfer(address(this).balance);
else {
 // jogadores precisam ter revelado os commits
 require (donoRevealed && jogadorRevealed);
 // Encerra o jogo
 myState = StatesType.fim;
  // dono não revelou corretamente
   if (!(valorDono.isCorrect()) && valorJogador.isCorrect()) {
   jogador.transfer(address(this).balance);
  // jogador não revelou corretamente
  else if (valorDono.isCorrect() && !(valorJogador.isCorrect())) {
    dono.transfer(address(this).balance);
  // ambos revelaram correto
  else if (valorDono.isCorrect() && valorJogador.isCorrect()){
   uint soma;
    soma = uint8(valorDono.getValue())+uint8(valorJogador.getValue());
   // dono ganha com par
   if (soma % 2 == 0) {
      dono.transfer(address(this).balance);
    // jogador ganha com impar
      jogador.transfer(address(this).balance);
```

Para os testes, foram utilizadas as funções de hash prontas disponibilizadas pelo professor no exemplo de Commit. Tais funções criam o commit concatenando o nonce ao valor.

```
#!/usr/bin/python3

from brownie import *
import brownie
import hashlib

def createNonce(s):
    n = hashlib.sha256()
    n.update(s)
    return n.digest()

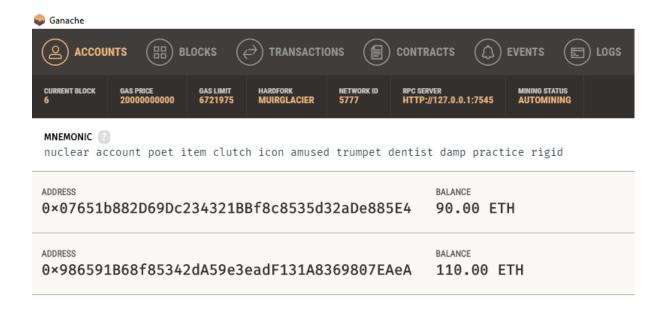
def doCommit(n,v):
    c = hashlib.sha256()
    c.update(n)
    c.update(v)
    return c.digest()
```

Como forma de testes, foram criados diversos cenários que mostram o comportamento do contrato diante de diferentes situações.

accounts[0] é o endereço do dono. accounts[1] é o endereço do jogador.

```
def main():
    # deploy da biblioteca de commit
    dono = accounts[0]
    SimpleCommit.deploy({'from':dono})
   valor_dono = b'2'
    nonce dono = createNonce(b'nonce1')
    commit dono = doCommit(nonce dono, valor dono)
    # deploy do jogo
   parImpar.deploy(commit dono, {"from":dono, "value":"10 ether"})
    address = parImpar[0].address
    contrato = parImpar.at(address)
    jogador = accounts[1]
   valor_jogador = b'3'
    nonce_jogador = createNonce(b'nonce2')
    commit jogador = doCommit(nonce_jogador,valor_jogador)
    contrato.entraJogo(commit jogador,{"from":jogador,'value':"20 ether"})
    contrato.donoRevela(valor_dono,nonce_dono,{"from":dono})
    contrato.jogadorRevela(valor jogador,nonce jogador,{"from":jogador})
    contrato.pagaVencedor()
```

Ambos revelaram correto Soma = 2 + 3 = 5 5 % 2 = 1 Jogador venceu

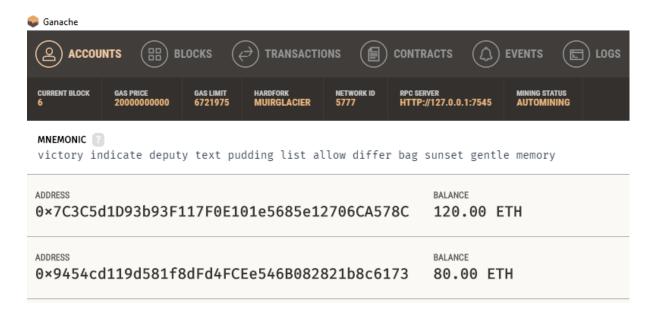


```
def main():
    # deploy da biblioteca de commit
   dono = accounts[0]
    SimpleCommit.deploy({'from':dono})
   valor_dono = b'2'
   nonce dono = createNonce(b'nonce1')
   commit dono = doCommit(nonce dono, valor dono)
   # deploy do jogo
   parImpar.deploy(commit dono, {"from":dono, "value":"10 ether"})
   address = parImpar[0].address
    contrato = parImpar.at(address)
   jogador = accounts[1]
   valor jogador = b'3'
   nonce jogador = createNonce(b'nonce2')
   commit_jogador = doCommit(nonce_jogador,valor_jogador)
   contrato.entraJogo(commit jogador,{"from":jogador,'value':"5 ether"})
   contrato.donoRevela(valor dono, nonce dono, {"from":dono})
    contrato.jogadorRevela(valor jogador,nonce jogador,{"from":jogador})
    contrato.pagaVencedor()
```

O jogador tentou fazer uma aposta de 5 ether, que é um valor menor do que o mínimo definido pelo dono. No caso, a aposta deveria ser maior ou igual a 10 ether.

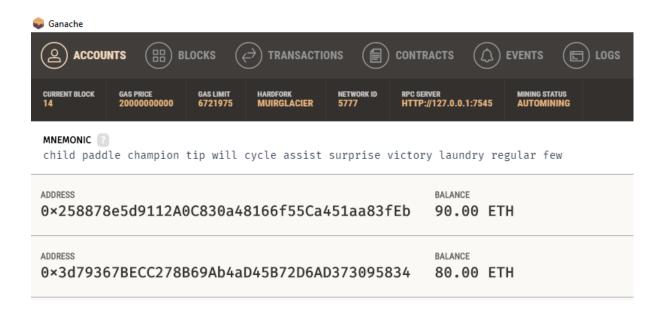
```
def main():
    # deploy da biblioteca de commit
    dono = accounts[0]
    SimpleCommit.deploy({'from':dono})
    valor_dono = b'2'
    nonce_dono = createNonce(b'nonce1')
    commit dono = doCommit(nonce dono, valor dono)
    # deploy do jogo
    parImpar.deploy(commit dono, { "from ": dono, "value ": "10 ether "})
    address = parImpar[0].address
    contrato = parImpar.at(address)
    jogador = accounts[1]
    valor jogador = b'3'
    nonce jogador = createNonce(b'nonce2')
    commit_jogador = doCommit(nonce_jogador,valor_jogador)
    contrato.entraJogo(commit jogador,{"from":jogador,'value':"20 ether"})
    contrato.donoRevela(valor dono, nonce dono, {"from":dono})
    contrato.jogadorRevela(b'1',nonce_jogador,{"from":jogador})
    contrato.pagaVencedor()
```

Ambos revelaram o commit, mas o jogador revelou incorreto Nesse caso, dono ganhou



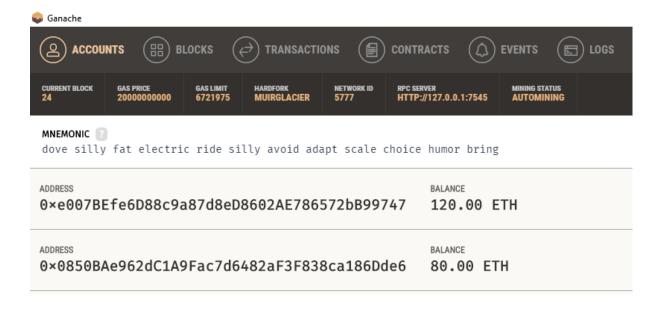
```
def main():
    # deploy da biblioteca de commit
    dono = accounts[0]
    SimpleCommit.deploy({'from':dono})
   valor_dono = b'2'
    nonce dono = createNonce(b'nonce1')
    commit dono = doCommit(nonce dono, valor dono)
    # deploy do jogo
   parImpar.deploy(commit dono, {"from":dono, "value":"10 ether"})
    address = parImpar[0].address
    contrato = parImpar.at(address)
    jogador = accounts[1]
   valor_jogador = b'3'
    nonce_jogador = createNonce(b'nonce2')
    commit jogador = doCommit(nonce_jogador,valor_jogador)
    contrato.entraJogo(commit jogador,{"from":jogador,'value':"20 ether"})
    for _ in range(20):
        contrato.pagaVencedor()
```

Nenhum dos jogadores revelaram o commit Os dois perderam dinheiro



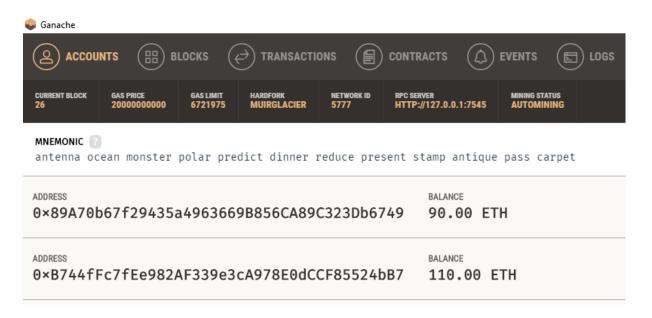
```
def main():
    # deploy da biblioteca de commit
    dono = accounts[0]
    SimpleCommit.deploy({'from':dono})
   valor_dono = b'2'
    nonce dono = createNonce(b'nonce1')
    commit dono = doCommit(nonce dono, valor dono)
    # deploy do jogo
   parImpar.deploy(commit dono, {"from":dono, "value":"10 ether"})
    address = parImpar[0].address
    contrato = parImpar.at(address)
    jogador = accounts[1]
   valor_jogador = b'3'
    nonce_jogador = createNonce(b'nonce2')
    commit jogador = doCommit(nonce_jogador,valor_jogador)
    contrato.entraJogo(commit jogador,{"from":jogador,'value':"20 ether"})
    contrato.donoRevela(valor_dono,nonce_dono,{"from":dono})
    for _ in range(20):
        contrato.pagaVencedor()
```

Somente o dono revelou o commit, ganhando a aposta



```
def main():
    # deploy da biblioteca de commit
    dono = accounts[0]
    SimpleCommit.deploy({'from':dono})
    valor_dono = b'2'
    nonce_dono = createNonce(b'nonce1')
    commit dono = doCommit(nonce dono, valor dono)
    # deploy do jogo
    parImpar.deploy(commit dono, { "from":dono, "value": "10 ether" })
    address = parImpar[0].address
    contrato = parImpar.at(address)
    jogador = accounts[1]
    valor_jogador = b'3'
    nonce_jogador = createNonce(b'nonce2')
    commit jogador = doCommit(nonce_jogador,valor_jogador)
    contrato.entraJogo(commit jogador,{"from":jogador,'value':"20 ether"})
    contrato.jogadorRevela(valor_jogador,nonce_jogador,{"from":jogador})
    for in range (20):
        contrato.pagaVencedor()
    contrato.donoRevela(valor_dono,nonce_dono,{"from":dono})
    contrato.pagaVencedor()
```

Dono quis revelar commit após o limite estipulado, nesse caso, o jogador já tinha ganhado.



```
def main():
    # deploy da biblioteca de commit
    dono = accounts[0]
    SimpleCommit.deploy({'from':dono})
    valor_dono = b'2'
    nonce dono = createNonce(b'nonce1')
    commit dono = doCommit(nonce dono, valor dono)
    # deploy do jogo
   parImpar.deploy(commit dono, {"from":dono, "value":"10 ether"})
    address = parImpar[0].address
    contrato = parImpar.at(address)
    jogador = accounts[1]
   valor_jogador = b'3'
    nonce_jogador = createNonce(b'nonce2')
    commit jogador = doCommit(nonce_jogador,valor_jogador)
    contrato.entraJogo(commit jogador,{"from":jogador,'value':"20 ether"})
    contrato.jogadorRevela(b'1',nonce_jogador,{"from":jogador})
    for in range (20):
        contrato.pagaVencedor()
```

Mesmo o jogador tendo revelado incorretamente o commit, ele ganha a aposta, já que o dono não revelou.

