한 번에 끝내는 블록체인 개발 A to Z

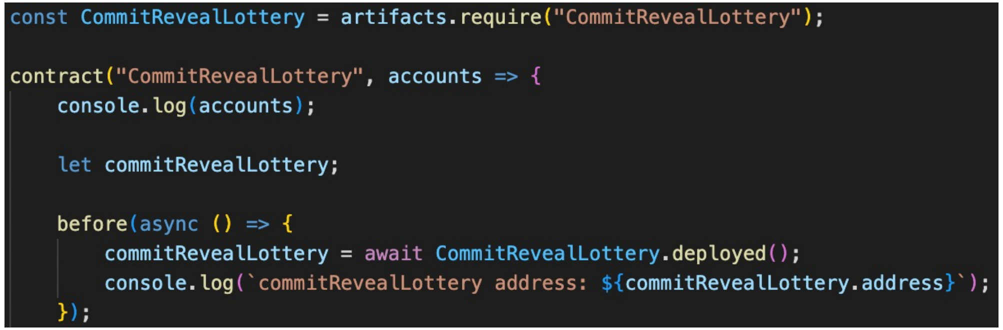
Chapter 3

Lottery 컨트랙트 v1 개발

Chapter 3 Lottery 컨트랙트 v1 개발

CommitRevealLottery 컨트랙트 테스트하기 - truffle test

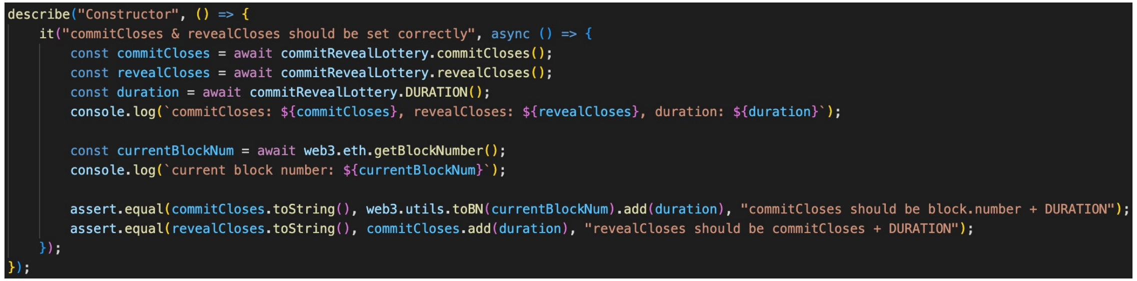
컨트랙트 연결하기



```
const CommitRevealLottery = artifacts.require("CommitRevealLottery");

contract("CommitRevealLottery", accounts => {
    console.log(accounts);

    let commitRevealLottery;

    before(async () => {
        commitRevealLottery = await CommitRevealLottery.deployed();
        console.log(`commitRevealLottery address: ${commitRevealLottery.address}`);
    });
```

배포 후 생성자 값 체크

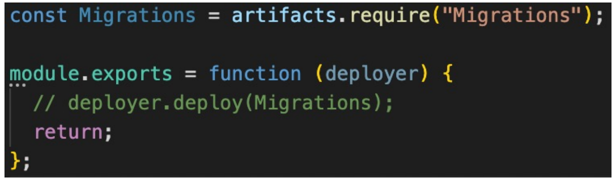
- 이 테스트의 목적은 배포 후 생성자 값이 잘 세팅되었는지 체크하는 것
- 즉, 이 테스트의 제목, 목차는 Constructor라고 지을 수 있음 → describe(“Constructor”)
- commitCloses가 현재 블록 넘버보다 DURATION 만큼 더 후 인지 체크
- revealCloses가 commitCloses보다 DURATION 만큼 더 후 인지 체크



```javascript
describe("Constructor", () => {
    it("commitCloses & revealCloses should be set correctly", async () => {
        const commitCloses = await commitRevealLottery.commitCloses();
        const revealCloses = await commitRevealLottery.revealCloses();
        const duration = await commitRevealLottery.DURATION();
        console.log(`commitCloses: ${commitCloses}, revealCloses: ${revealCloses}, duration: ${duration}`);

        const currentBlockNum = await web3.eth.getBlockNumber();
        console.log(`current block number: ${currentBlockNum}`);

        assert.equal(commitCloses.toString(), web3.utils.toBN(currentBlockNum).add(duration), "commitCloses should be block.number + DURATION");
        assert.equal(revealCloses.toString(), commitCloses.add(duration), "revealCloses should be commitCloses + DURATION");
    });
});
```

Migrations 배포 코드 스킵하기

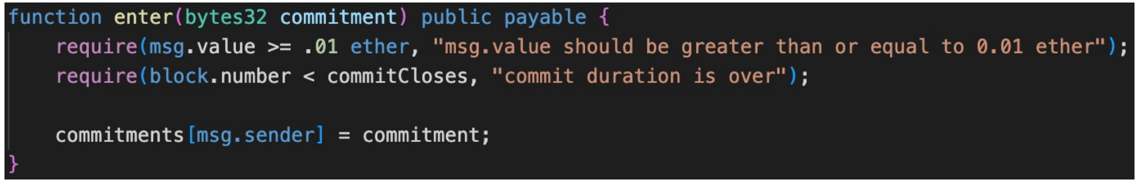
- contract() 함수 실행시, migrations 디렉토리 밑에 있는 모든 배포코드가 실행됨
- Migrations 배포시, 이후의 컨트랙트 배포될 때마다 Migrations의 setCompleted() 함수가 호출됨. 의도치 않은 트랜잭션 생성을 방지하기 위해 Migrations 배포는 스킵하기
    - commitCloses에서 block.number를 비교하는 부분을 제대로 테스트해보고자
- 나중에 Truffle을 이용해 컨트랙트 배포시 Migrations는 같이 배포하여 이용하는게 좋음 → truffle migrate 명령어 실행시 기존에 배포된 컨트랙트는 스킵해줌



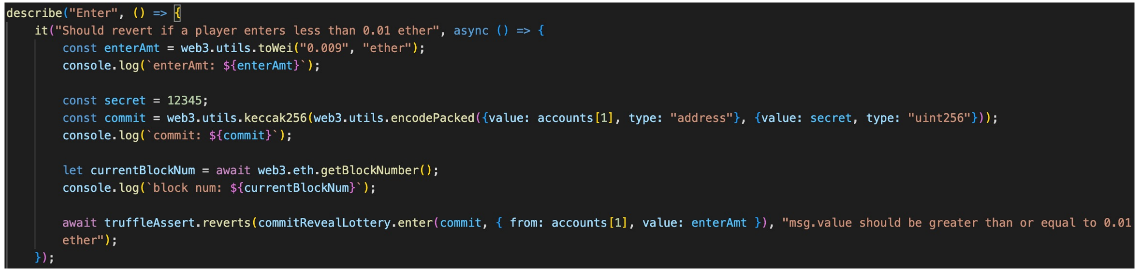
```
const Migrations = artifacts.require("Migrations");

module.exports = function (deployer) {
  // deployer.deploy(Migrations);
  return;
};
```

enter() 테스트

- 이 테스트의 목적은 enter() 기능을 테스트하는 것
- 즉, 이 테스트의 제목, 목차는 Enter라고 지을 수 있음 → describe(“Enter”)
- 먼저 require() 구문이 잘 동작하는지 체크
- revert가 잘 되는지 확인하기 위해 truffle-assertion npm 모듈 사용
- 사용자가 보낸 ETH가 0.01 이상인지 체크



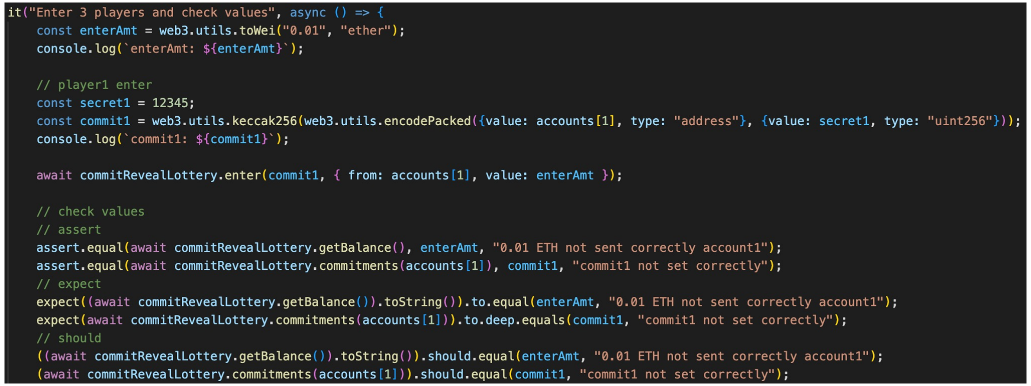
```
function enter(bytes32 commitment) public payable {
    require(msg.value >= .01 ether, "msg.value should be greater than or equal to 0.01 ether");
    require(block.number < commitCloses, "commit duration is over");

    commitments[msg.sender] = commitment;
}
```



```
describe("Enter", () => {
    it("Should revert if a player enters less than 0.01 ether", async () => {
        const enterAmt = web3.utils.toWei("0.009", "ether");
        console.log(`enterAmt: ${enterAmt}`);

        const secret = 12345;
        const commit = web3.utils.keccak256(web3.utils.encodePacked({value: accounts[1], type: "address"}, {value: secret, type: "uint256"}));
        console.log(`commit: ${commit}`);

        let currentBlockNum = await web3.eth.getBlockNumber();
        console.log(`block num: ${currentBlockNum}`);

        await truffleAssert.reverts(commitRevealLottery.enter(commit, { from: accounts[1], value: enterAmt }), "msg.value should be greater than or equal to 0.01
        ether");
    });
```

enter() 테스트

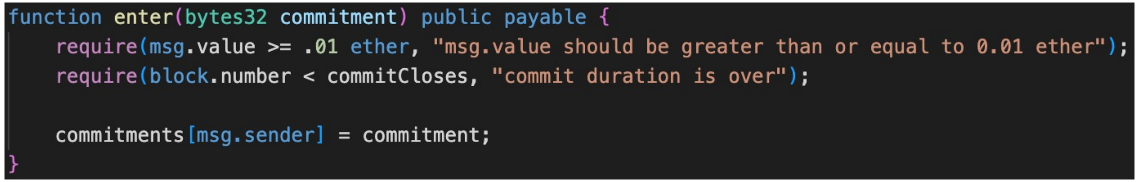
- 3명의 player가 enter 하는 상황 테스트
- player 별로 직접 commit 값을 생성하여 enter
- 한 명씩 enter 할 때마다 컨트랙트의 ETH balance가 의도한대로 느는지, commitments 매핑에 각 commit 값이 잘 저장됐는지 체크



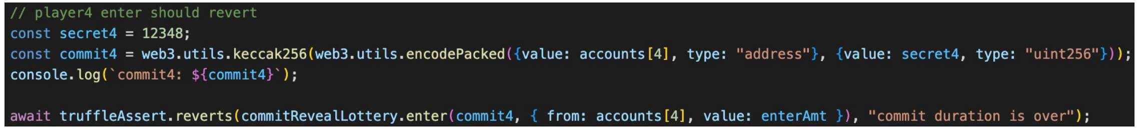
```
it("Enter 3 players and check values", async () => {
    const enterAmt = web3.utils.toWei("0.01", "ether");
    console.log(`enterAmt: ${enterAmt}`);

    // player1 enter
    const secret1 = 12345;
    const commit1 = web3.utils.keccak256(web3.utils.encodePacked({value: accounts[1], type: "address"}, {value: secret1, type: "uint256"}));
    console.log(`commit1: ${commit1}`);

    await commitRevealLottery.enter(commit1, { from: accounts[1], value: enterAmt });

    // check values
    // assert
    assert.equal(await commitRevealLottery.getBalance(), enterAmt, "0.01 ETH not sent correctly account1");
    assert.equal(await commitRevealLottery.commitments(accounts[1]), commit1, "commit1 not set correctly");
    // expect
    expect((await commitRevealLottery.getBalance()).toString()).to.equal(enterAmt, "0.01 ETH not sent correctly account1");
    expect(await commitRevealLottery.commitments(accounts[1])).to.deep.equals(commit1, "commit1 not set correctly");
    // should
    ((await commitRevealLottery.getBalance()).toString()).should.equal(enterAmt, "0.01 ETH not sent correctly account1");
    (await commitRevealLottery.commitments(accounts[1])).should.equal(commit1, "commit1 not set correctly");
```

enter() 테스트

- 3명의 player가 enter 하는 상황 테스트
- 4번째 사용자가 enter 시도하는 순간, “commit duration is over” require 구문에서 정상적으로 revert 되는지 체크
    - commitCloses: 컨트랙트 배포시의 블록 넘버(2번 블록) + DURATION(4블록) = 6번 블록
    - 배포 직후 블록 넘버는 2 → 3명 enter하고 나면 블록 넘버는 5 → 4번째 enter시 revert 되는 것



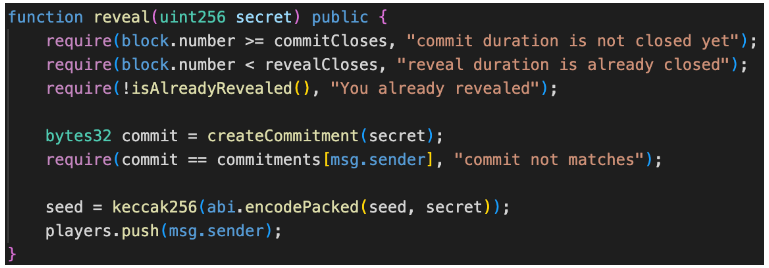
```
function enter(bytes32 commitment) public payable {
    require(msg.value >= .01 ether, "msg.value should be greater than or equal to 0.01 ether");
    require(block.number < commitCloses, "commit duration is over");

    commitments[msg.sender] = commitment;
}
```



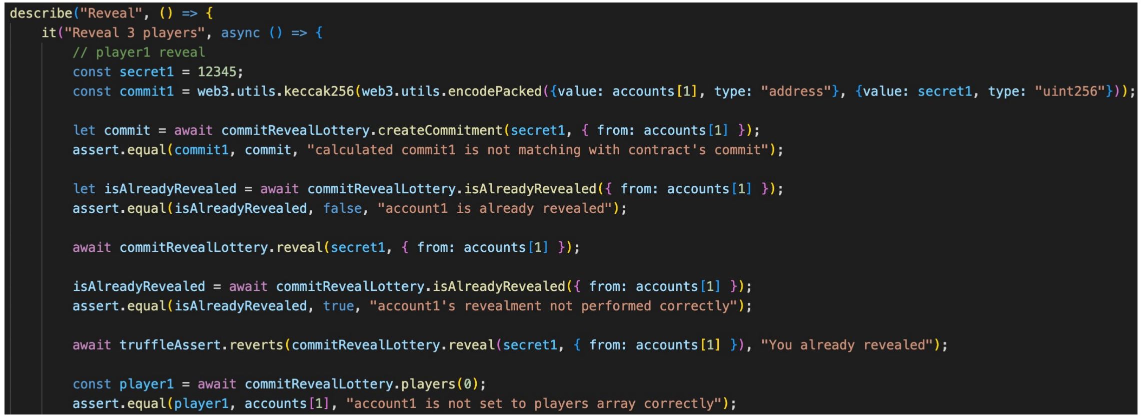
```
// player4 enter should revert
const secret4 = 12348;
const commit4 = web3.utils.keccak256(web3.utils.encodePacked({value: accounts[4], type: "address"}, {value: secret4, type: "uint256"}));
console.log(`commit4: ${commit4}`);

await truffleAssert.reverts(commitRevealLottery.enter(commit4, { from: accounts[4], value: enterAmt }), "commit duration is over");
```

reveal() 테스트

- 각 player 별로 컨트랙트의 createCommitment()를 통한 commit 값과 직접 만든 commit값 같은지 체크
- reveal() 전후로 isAlreadyRevealed()가 false → true로 전환되는지 체크
- 다시 같은 player가 같은 secret 값으로 reveal() 시도하면, “You already revealed” require 구문에서 정상적으로 revert 되는지 체크
- 각 player가 reveal 할 때마다 players 배열에 잘 저장되는지 체크



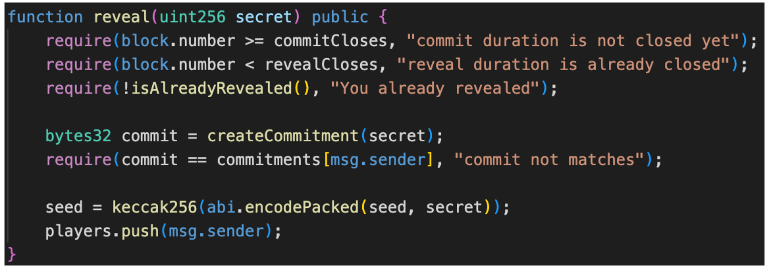
```
function reveal(uint256 secret) public {
    require(block.number >= commitCloses, "commit duration is not closed yet");
    require(block.number < revealCloses, "reveal duration is already closed");
    require(!isAlreadyRevealed(), "You already revealed");

    bytes32 commit = createCommitment(secret);
    require(commit == commitments[msg.sender], "commit not matches");

    seed = keccak256(abi.encodePacked(seed, secret));
    players.push(msg.sender);
}
```



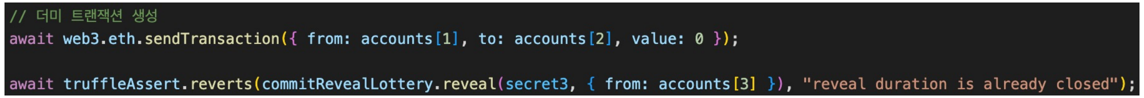
```
describe("Reveal", () => {
    it("Reveal 3 players", async () => {
        // player1 reveal
        const secret1 = 12345;
        const commit1 = web3.utils.keccak256(web3.utils.encodePacked({value: accounts[1], type: "address"}, {value: secret1, type: "uint256"}));

        let commit = await commitRevealLottery.createCommitment(secret1, { from: accounts[1] });
        assert.equal(commit1, commit, "calculated commit1 is not matching with contract's commit");

        let isAlreadyRevealed = await commitRevealLottery.isAlreadyRevealed({ from: accounts[1] });
        assert.equal(isAlreadyRevealed, false, "account1 is already revealed");

        await commitRevealLottery.reveal(secret1, { from: accounts[1] });

        isAlreadyRevealed = await commitRevealLottery.isAlreadyRevealed({ from: accounts[1] });
        assert.equal(isAlreadyRevealed, true, "account1's revealment not performed correctly");

        await truffleAssert.reverts(commitRevealLottery.reveal(secret1, { from: accounts[1] }), "You already revealed");

        const player1 = await commitRevealLottery.players(0);
        assert.equal(player1, accounts[1], "account1 is not set to players array correctly");
```

reveal() 테스트

- 더미 트랜잭션을 하나 생성시켜 현재 블록 넘버가 revealCloses 블록 넘버에 도달하게 만든 후, reveal 시도시, “reveal duration is already closed” require 구문에서 정상적으로 revert 되는지 체크
    - revealCloses: commitCloses(6번 블록) + DURATION(4) = 10번 블록
    - 3명 reveal 후 블록 넘버 9 → 더미 트랜잭션 생성 후 블록 넘버 10 → 이때 reveal시 revert 되는 것



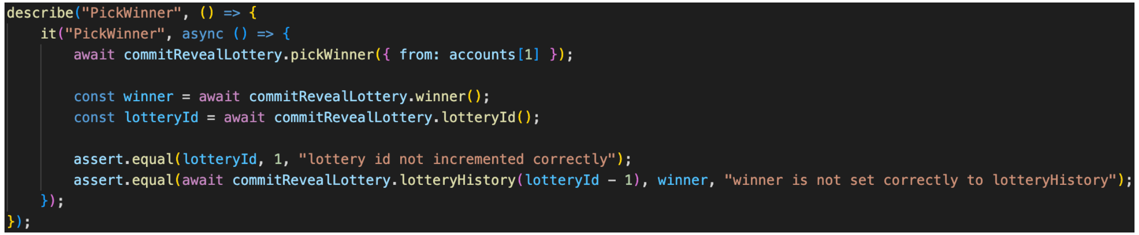
```
function reveal(uint256 secret) public {
    require(block.number >= commitCloses, "commit duration is not closed yet");
    require(block.number < revealCloses, "reveal duration is already closed");
    require(!isAlreadyRevealed(), "You already revealed");

    bytes32 commit = createCommitment(secret);
    require(commit == commitments[msg.sender], "commit not matches");

    seed = keccak256(abi.encodePacked(seed, secret));
    players.push(msg.sender);
}
```



```
// 더미 트랜잭션 생성
await web3.eth.sendTransaction({ from: accounts[1], to: accounts[2], value: 0 });

await truffleAssert.reverts(commitRevealLottery.reveal(secret3, { from: accounts[3] }), "reveal duration is already closed");
```

pickWinner() 테스트

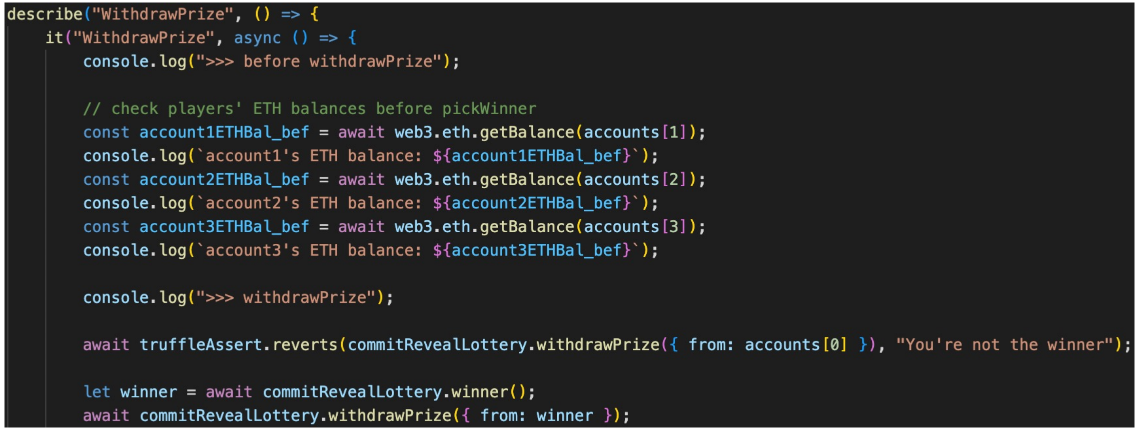
- pickWinner() 호출 후, lotteryId가 1로 잘 증가했는지, 0회차 lotteryHistory에 winner가 잘 저장됐는지 체크



```
describe("PickWinner", () => {
    it("PickWinner", async () => {
        await commitRevealLottery.pickWinner({ from: accounts[1] });

        const winner = await commitRevealLottery.winner();
        const lotteryId = await commitRevealLottery.lotteryId();

        assert.equal(lotteryId, 1, "lottery id not incremented correctly");
        assert.equal(await commitRevealLottery.lotteryHistory(lotteryId - 1), winner, "winner is not set correctly to lotteryHistory");
    });
});
```

withdrawPrize() 테스트

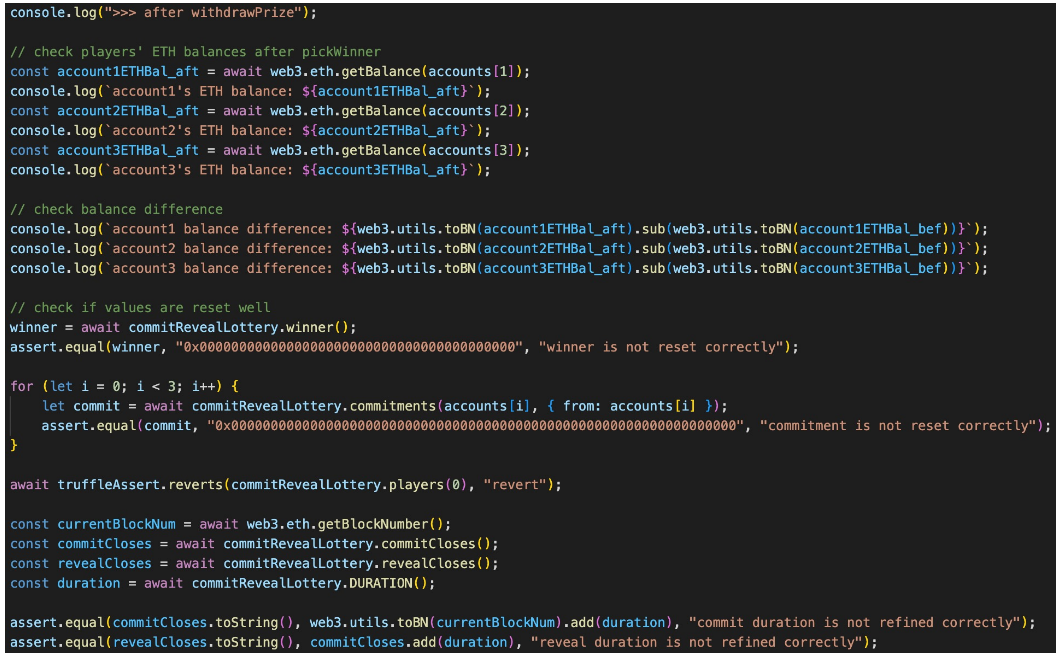
- withdrawPrize() 호출 전, player 3명의 ETH balance 체크
    - withdrawPrize 후, winner의 ETH balance가 0.03 ETH 늘었는지 체크하기 위함
- winner가 아닌 계정으로 withdrawPrize() 호출하면, “You’re not the winner” require 구문에서 정상적으로 revert 되는지 체크
- winner 계정으로 withdrawPrize() 호출



```
describe("WithdrawPrize", () => {
    it("WithdrawPrize", async () => {
        console.log(">>> before withdrawPrize");

        // check players' ETH balances before pickWinner
        const account1ETHBal_bef = await web3.eth.getBalance(accounts[1]);
        console.log(`account1's ETH balance: ${account1ETHBal_bef}`);
        const account2ETHBal_bef = await web3.eth.getBalance(accounts[2]);
        console.log(`account2's ETH balance: ${account2ETHBal_bef}`);
        const account3ETHBal_bef = await web3.eth.getBalance(accounts[3]);
        console.log(`account3's ETH balance: ${account3ETHBal_bef}`);

        console.log(">>> withdrawPrize");

        await truffleAssert.reverts(commitRevealLottery.withdrawPrize({ from: accounts[0] }), "You're not the winner");

        let winner = await commitRevealLottery.winner();
        await commitRevealLottery.withdrawPrize({ from: winner });
```

withdrawPrize() 테스트

- withdrawPrize() 호출 후, player 3명의 ETH balance 체크
- withdrawPrize() 호출 전후로 0.03 ETH 늘어난 계정 체크
- 변수들 모두 잘 리셋됐는지 체크



```
console.log(">>> after withdrawPrize");

// check players' ETH balances after pickWinner
const account1ETHBal_aft = await web3.eth.getBalance(accounts[1]);
console.log(`account1's ETH balance: ${account1ETHBal_aft}`);
const account2ETHBal_aft = await web3.eth.getBalance(accounts[2]);
console.log(`account2's ETH balance: ${account2ETHBal_aft}`);
const account3ETHBal_aft = await web3.eth.getBalance(accounts[3]);
console.log(`account3's ETH balance: ${account3ETHBal_aft}`);

// check balance difference
console.log(`account1 balance difference: ${web3.utils.toBN(account1ETHBal_aft).sub(web3.utils.toBN(account1ETHBal_bef))}`);
console.log(`account2 balance difference: ${web3.utils.toBN(account2ETHBal_aft).sub(web3.utils.toBN(account2ETHBal_bef))}`);
console.log(`account3 balance difference: ${web3.utils.toBN(account3ETHBal_aft).sub(web3.utils.toBN(account3ETHBal_bef))}`);

// check if values are reset well
winner = await commitRevealLottery.winner();
assert.equal(winner, "0x0000000000000000000000000000000000000000", "winner is not reset correctly");

for (let i = 0; i < 3; i++) {
    let commit = await commitRevealLottery.commitments(accounts[i], { from: accounts[i] });
    assert.equal(commit, "0x0000000000000000000000000000000000000000000000000000000000000000", "commitment is not reset correctly");
}

await truffleAssert.reverts(commitRevealLottery.players(0), "revert");

const currentBlockNum = await web3.eth.getBlockNumber();
const commitCloses = await commitRevealLottery.commitCloses();
const revealCloses = await commitRevealLottery.revealCloses();
const duration = await commitRevealLottery.DURATION();

assert.equal(commitCloses.toString(), web3.utils.toBN(currentBlockNum).add(duration), "commit duration is not refined correctly");
assert.equal(revealCloses.toString(), commitCloses.add(duration), "reveal duration is not refined correctly");
```