**Adding more tests**

**Expanding our test coverage**

Welcome back! Let's continue testing our Raffle contract.

We should test if the Raffle reverts when you don't pay enough money. Open your RaffleTest.t.sol and write the following:

function testRaffleRevertsWhenYouDontPayEnough() public {

// Arrange

vm.prank(PLAYER);

// Act / Accert

vm.expectRevert(Raffle.Raffle\_\_NotEnoughEthSent.selector);

raffle.enterRaffle();

}

We use vm.prank(PLAYER); to mimic our user and to test the revert functionality we use vm.expectRevert();. When we go back to Raffle.sol we can see that the error we should get is Raffle\_\_NotEnoughEthSent. This we can leverage in our expectRevert() function: vm.expectRevert(Raffle.Raffle\_\_NotEnoughEthSent.selector);. Like this, we say, we expect a revert due to the Raffle\_\_NotEnoughEthSent error. We will teach you about the selector key and what functions selectors are later in detail.

To test run: forge test --mt testRaffleRevertsWhenYouDontPayEnough

It passes! Great!

Besides, we want to test if the Raffle updates or more precisely adds players to the s\_players array in Raffle.sol

function testRaffleRecordsPlayersWhenTheyEnter() public {

// Arrange

vm.prank(PLAYER);

// Act

raffle.enterRaffle{value: entranceFee}();

// Assert

address playerRecorded = raffle.getPlayer(0);

assert(playerRecorded == PLAYER);

}

With vm.prank(PLAYER); we pretend to be a player and enter the Raffle raffle.enterRaffle{value: entranceFee}(); with an entrance fee.

Unfortunately, we don't have yet a function to get the players from Raffle.sol. So let's do this! Go to Raffle.sol and add the following function to your getter functions:

function getPlayer(uint256 indexOfPlayer) external view returns (address) {

return s\_players[indexOfPlayer];

}

Then let's use this function in our test. Since, we only have one user we want to get the first player: address playerRecorded = raffle.getPlayer(0);

Run the test using forge test --mt testRaffleRecordsPlayersWhenTheyEnter.

Ohhh, it fails. Let's test again with more verbose: forge test --mt testRaffleRecordsPlayersWhenTheyEnter -vvvv

Ahhh, we are **OutOfFunds**. Thus, let's add some funds in the setup() function: vm.deal(PLAYER, STARTING\_PLAYER\_BALANCE);

function setUp() external {

DeployRaffle deployer = new DeployRaffle();

(raffle, helperConfig) = deployer.deployContract();

HelperConfig.NetworkConfig memory config = helperConfig.getConfig();

entranceFee = config.entranceFee;

interval = config.interval;

vrfCoordinator = config.vrfCoordinator;

gasLane = config.gasLane;

callbackGasLimit = config.callbackGasLimit;

subscriptionId = config.subscriptionId;

vm.deal(PLAYER, STARTING\_PLAYER\_BALANCE);

}

Let's test once more: forge test --mt testRaffleRecordsPlayersWhenTheyEnter -vvvv

It passes, amazing!

Great job! Let's keep going!