**Adding more tests**

**Expanding our test coverage**

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

We should test if the check upkeep returns false if the contract has no balance. Open your RaffleTest.t.sol and write the following:

function testCheckUpkeepReturnsFalseIfItHasNoBalance() public {

// Arrange

vm.warp(block.timestamp + interval + 1);

vm.roll(block.number + 1);

// Act

(bool upkeepNeeded, ) = raffle.checkUpkeep("");

// Assert

assert(!upkeepNeeded);

}

We use warp and roll to set the block.timestamp in the future. We call checkUpkeep and record its return in memory. We check it returned false.

**Note:** !upkeepNeeded means not upkeepNeeded meaning if upkeepNeeded is false that expression would read not false and not false is true.

Run the test using forge test --mt testCheckUpkeepReturnsFalseIfItHasNoBalance.

It passes, amazing!

What else? We should test if the check upkeep function returns false if the raffle is not Open. Paste the following inside RaffleTest.t.sol:

function testCheckUpkeepReturnsFalseIfRaffleIsntOpen() public {

// Arrange

vm.prank(PLAYER);

raffle.enterRaffle{value: entranceFee}();

vm.warp(block.timestamp + interval + 1);

vm.roll(block.number + 1);

raffle.performUpkeep("");

Raffle.RaffleState raffleState = raffle.getRaffleState();

// Act

(bool upkeepNeeded, ) = raffle.checkUpkeep("");

// Assert

assert(raffleState == Raffle.RaffleState.CALCULATING);

assert(upkeepNeeded == false);

}

We start by pranking the PLAYER. Then we enter the raffle using the correct entranceFee. After that, we use warp and roll to set block.timestamp in the future. We call performUpkeep. This will modify the RaffleState into CALCULATING. We then call checkUpkeep and record its return in memory. We check it returned false. We also check that the RaffleState is indeed CALCULATING.

Run the test using: forge test --mt testCheckUpkeepReturnsFalseIfRaffleIsntOpen.

It passes, great!

So testing goes amazing, but how do we know what's left to test? Let's run the following command in the CLI:

forge coverage --report debug > coverage.txt

We are interested in the Raffle.sol file for now. You can search for that and see an output like this:

Uncovered for src/Raffle.sol:

- Function "" (location: source ID 37, line 53, chars 1729-2253, hits: 0)

- Line (location: source ID 37, line 54, chars 1913-1940, hits: 0)

- Statement (location: source ID 37, line 54, chars 1913-1940, hits: 0)

- Line (location: source ID 37, line 55, chars 1950-1971, hits: 0)

- Statement (location: source ID 37, line 55, chars 1950-1971, hits: 0)

- Line (location: source ID 37, line 56, chars 1981-2014, hits: 0)

- Statement (location: source ID 37, line 56, chars 1981-2014, hits: 0)

- Line (location: source ID 37, line 57, chars 2024-2056, hits: 0)

- Statement (location: source ID 37, line 57, chars 2024-2056, hits: 0)

- Line (location: source ID 37, line 59, chars 2067-2127, hits: 0)

- Statement (location: source ID 37, line 59, chars 2067-2127, hits: 0)

- Line (location: source ID 37, line 60, chars 2137-2156, hits: 0)

- Statement (location: source ID 37, line 60, chars 2137-2156, hits: 0)

- Line (location: source ID 37, line 61, chars 2166-2199, hits: 0)

- Statement (location: source ID 37, line 61, chars 2166-2199, hits: 0)

- Line (location: source ID 37, line 62, chars 2209-2246, hits: 0)

- Statement (location: source ID 37, line 62, chars 2209-2246, hits: 0)

- Branch (branch: 2, path: 0) (location: source ID 37, line 97, chars 3717-3918, hits: 0)

- Branch (branch: 2, path: 1) (location: source ID 37, line 97, chars 3717-3918, hits: 0)

- Line (location: source ID 37, line 98, chars 3750-3907, hits: 0)

[...]

You can follow the locations indicated to find the lines not covered by tests. For example, in my Raffle.sol the code block starting on line 97 is this:

function performUpkeep(bytes calldata /\* performData \*/) external override {

(bool upkeepNeeded, ) = checkUpkeep("");

// require(upkeepNeeded, "Upkeep not needed");

@> if (!upkeepNeeded) {

@> revert Raffle\_\_UpkeepNotNeeded(

@> address(this).balance,

@> s\_players.length,

@> uint256(s\_raffleState)

@> );

}

And the output is right, we never tested this if + revert block inside performUpkeep.

But beware! This is not entirely accurate. For example, checkUpkeep doesn't appear in the report anymore, but we didn't test every single line out of it. We never tested if the upkeep returns false if enough time hasn't passed, we also never checked if the upkeep returns true when everything is alright.

Try writing these two tests yourself and then compare them against what [Patrick wrote](https://github.com/Cyfrin/foundry-smart-contract-lottery-f23/blob/d106fe245e0e44239dae2479b63545351ed1236a/test/unit/RaffleTest.t.sol).

Great job! Let's keep going!