Cheatsheet: Creating mocks with Prophecy in PHPUnit

prophecy

Available in test cases

Name of the class to mock (= to subclass)

```
$prophecy = $this->prophesize(Coffee::class);

/** @var Coffee|ProphecySubjectInterface $mock */
$mock = $prophecy->reveal();
```

revealed mock

Why mock a method or object?
▷ To "disable" an object (to not write to the DB, or to not launch a cruise missile).
▷ To have the method return a particular return value.
▷ To test that the method gets called in a certain way.

Cheatsheet: Using mocks with Prophecy in PHPUnit

name of the method
that should be called

fixed behaviour:
willReturn(true)
willReturnArgument(0)
willThrow(new \InvalidArgumentException())

```
$prophecy->fill(2)->willReturn(true)->shouldBeCalled();
```

required parameters

All methods will automatically be mocked!

With Prophecy, partial mocks are not possible.

expected calls:
shouldBeCalled()
shouldNotBeCalled
shouldBeCalledTimes(3)
shouldHaveBeenCalled()
shouldHaveBeenCalledTimes(2)