VTI Academy

WAY TO ENTERPRISE

Basic Programming

Lesson 07

VTI Academy

Unit Test

Unit Testing Fundamentals

A Unit is a Small Piece of Code

A method or function

A module or class

A small group of related classes

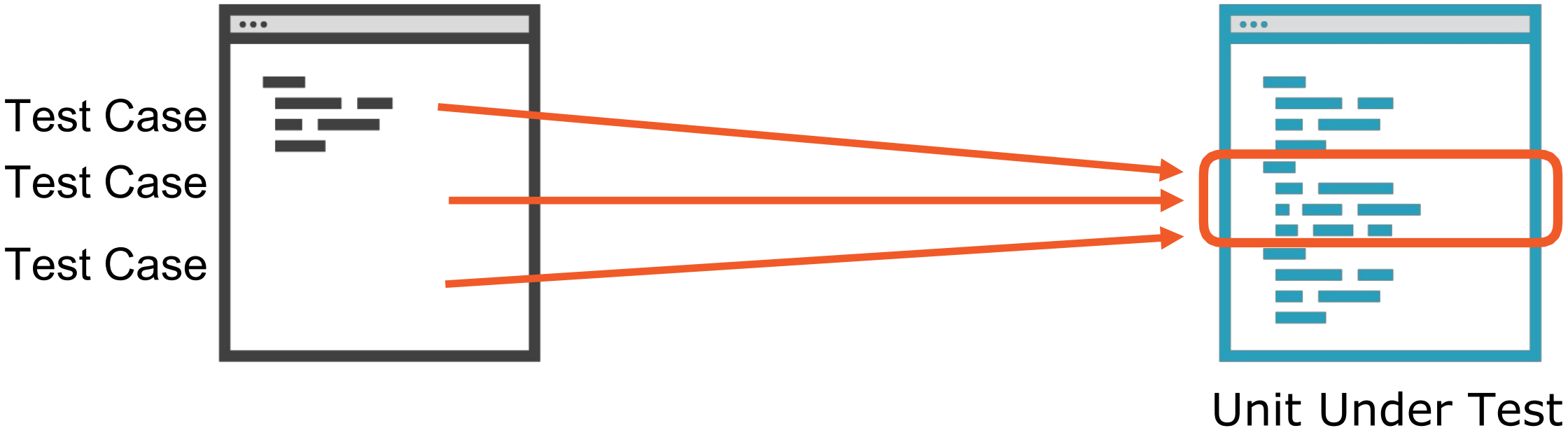
An Automated Unit Test

Is designed by a human

Runs without intervention

Reports either ‘pass’ or ‘fail’

Unit Test Vocabulary: Test Case



A suite

Unit Test Vocabulary: Test Runner

Test Case

Unit Under Test

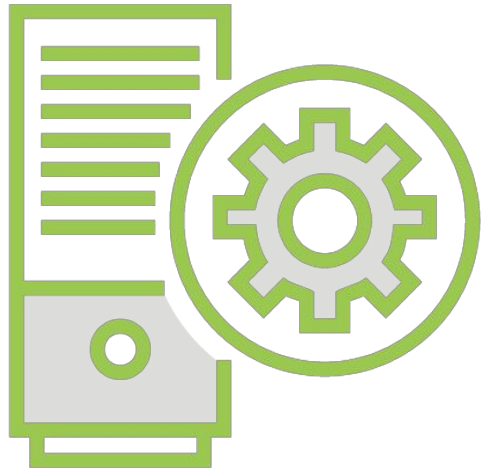
```
.
----------------------
Ran 1 test in 0.001s

OK
```

Test Runner

Choosing a Test Runner

Working Interactively

An IDE like VSCode



Continuous Integration

A Command Line Test Runner

Unit Test Vocabulary: Test Suite

Test Case

Test Case

Test Case

Test Suite

Units Under Test

```
.......
----------------------------------
Ran 7 tests in 0.000s

OK
```

Test Runner

Test Fixture: Order of Execution

Test Fixture: Order of Execution

```
setUp()
```

```
TestCaseMethod()
```

```
tearDown()
```

Test Fixture: Order of Execution

```
setUp()
```

Test Fixture for Strict Unit Tests

```
setUp()
```

```
TestCaseMethod()
```

```
class PhoneBookTest(unittest.TestCase):

    def setUp(self):
        self.phonebook = PhoneBook()

    def tearDown(self):
        pass

    def test_lookup_by_name(self):
        self.phonebook.add("Bob", "12345")
        number = self.phonebook.lookup("Bob")
        self.assertEqual("12345", number)

    def test_missing_name(self):
        with self.assertRaises(KeyError):
            self.phonebook.lookup("missing")

    def test_add_exists(self):
```

- Declare a class containing tests
- Set up fixture method
- Tear down fixture method
- First test case
- Second test case

Unit Test Vocabulary

Test Case

```
def setUp(self):
    pass

def tearDown(self):
    pass
```

Test Fixture

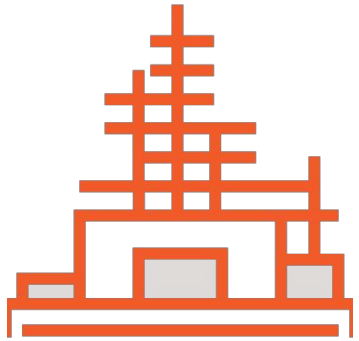
Unit Under Test

Test Suite

```
.......
--------------------------------
Ran 7 tests in 0.000s

OK
```

Test Runner

The Three Parts of a Test



Arrange

Set up the object to be tested, and collaborators

Act

Exercise the unit under test

Assert

Make claims about what happened

```
def test_lookup_by_name(self):
    self.phonebook.add("Bob", "12345")
    number = self.phonebook.lookup("Bob")
    self.assertEqual("12345", number)
```

- Test Case
- Name Arrange
- Act
- Assert

```
def test_lookup_by_name(self):
    self.phonebook.add("Bob", "12345")
    number = self.phonebook.lookup("Bob")
    self.assertEqual("12345", number)

def test_is_consistent(self):
    self.phonebook.add("Bob", "12345")
    self.assertTrue(
           self.phonebook.is_consistent())
    self.phonebook.add("Anna", "012345")
    self.assertTrue(
           self.phonebook.is_consistent())
    self.phonebook.add("Sue", "12345")
    self.assertFalse(
            self.phonebook.is_consistent())
    self.phonebook.add("Sue", "123")
    self.assertFalse(
            self.phonebook.is_consistent())
```

- Test Case Name
  - Arrange
  - Act
  - Assert
- Test Case Name
  - Act
  - Assert
  - Act
  - Assert
  - Act
  - Assert
  - Act
  - Assert

Summary

Vocabulary:

- Test Case
- Test Runner
- Test Suite
- Test Fixture

Test Case Design:

- Test name
- Arrange - Act - Assert

VTI Academy

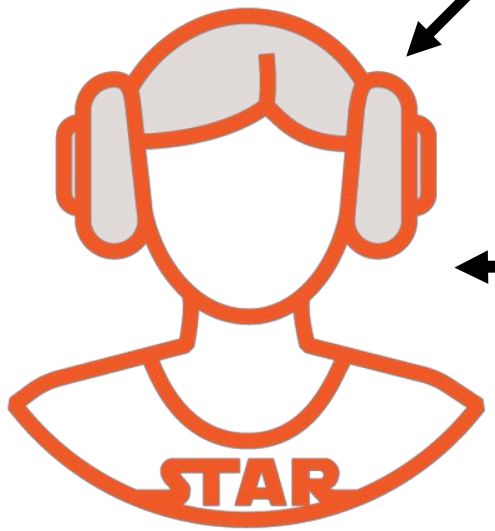
Test Doubles

Test Double - Like a Stunt Double

The real actor

Famous actress

The stunt double

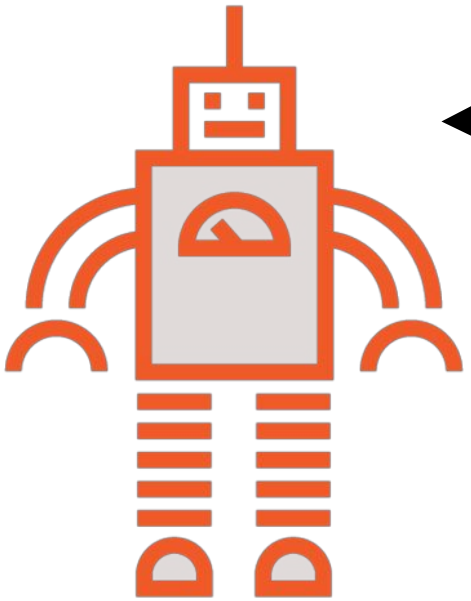


Wig

Som woman called Tracey

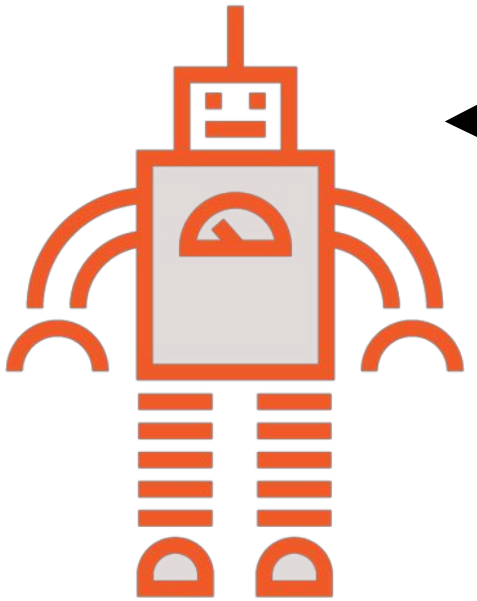
Test Double - Like a Stunt Double

The real object

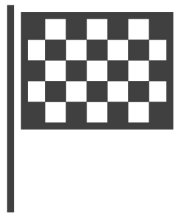


Complex, production logic

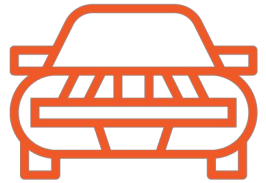
The test double

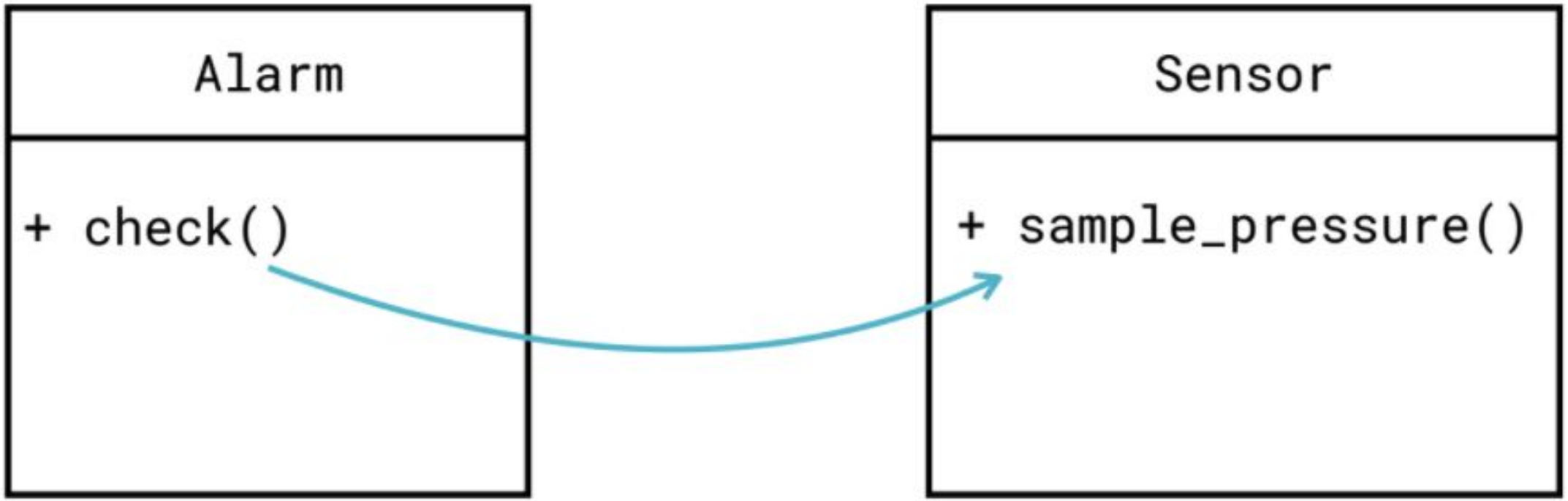


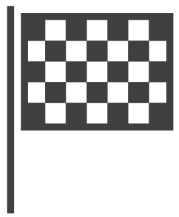
Everything it does is controlled by the test



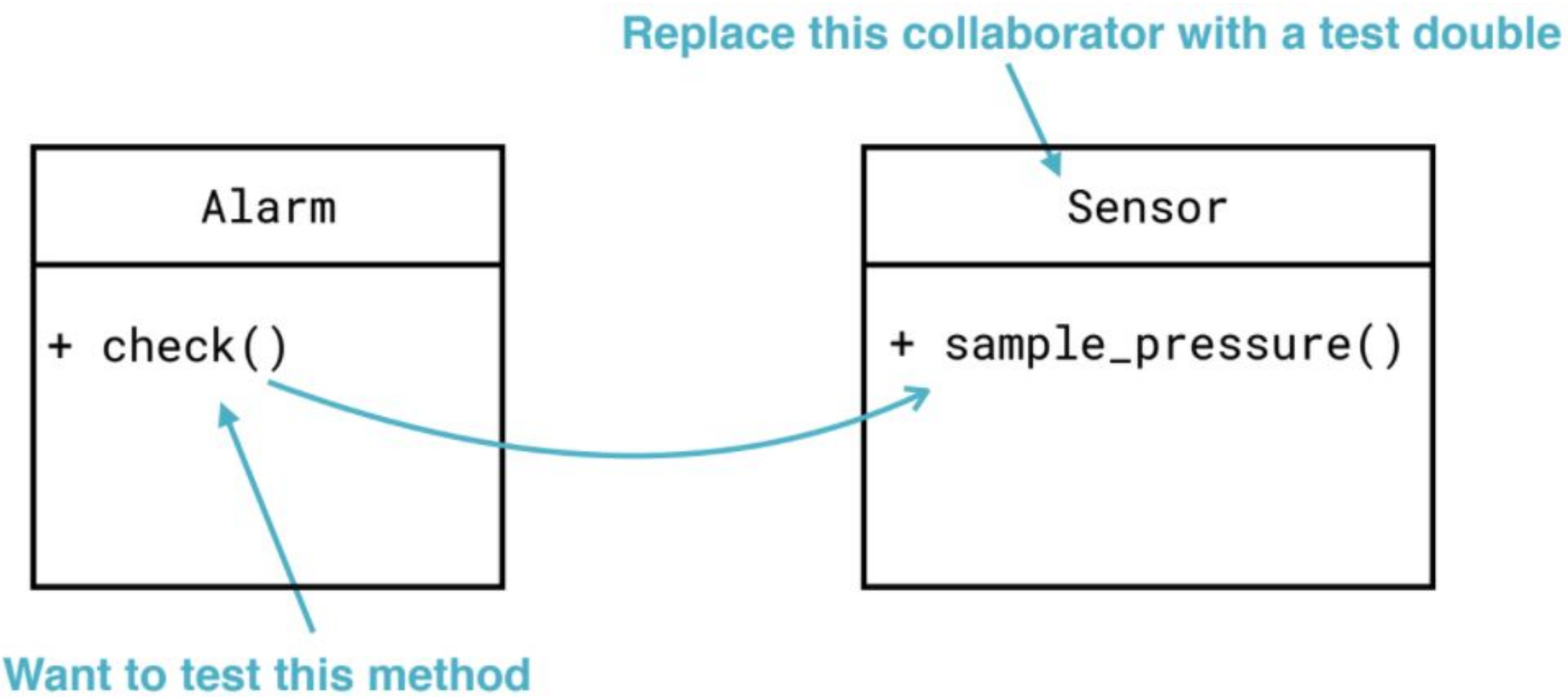
Racing Car Example

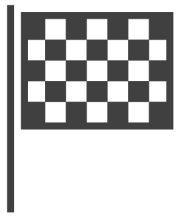




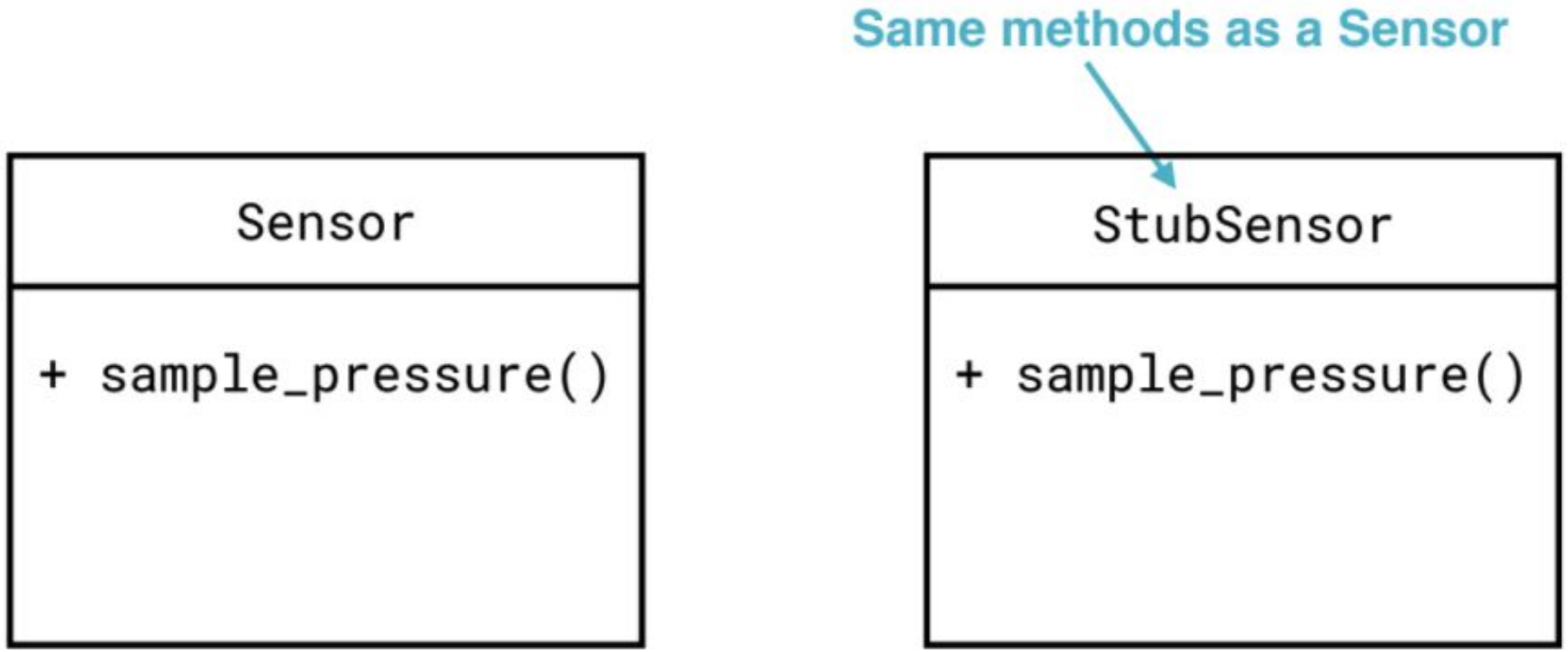


Racing Car Example



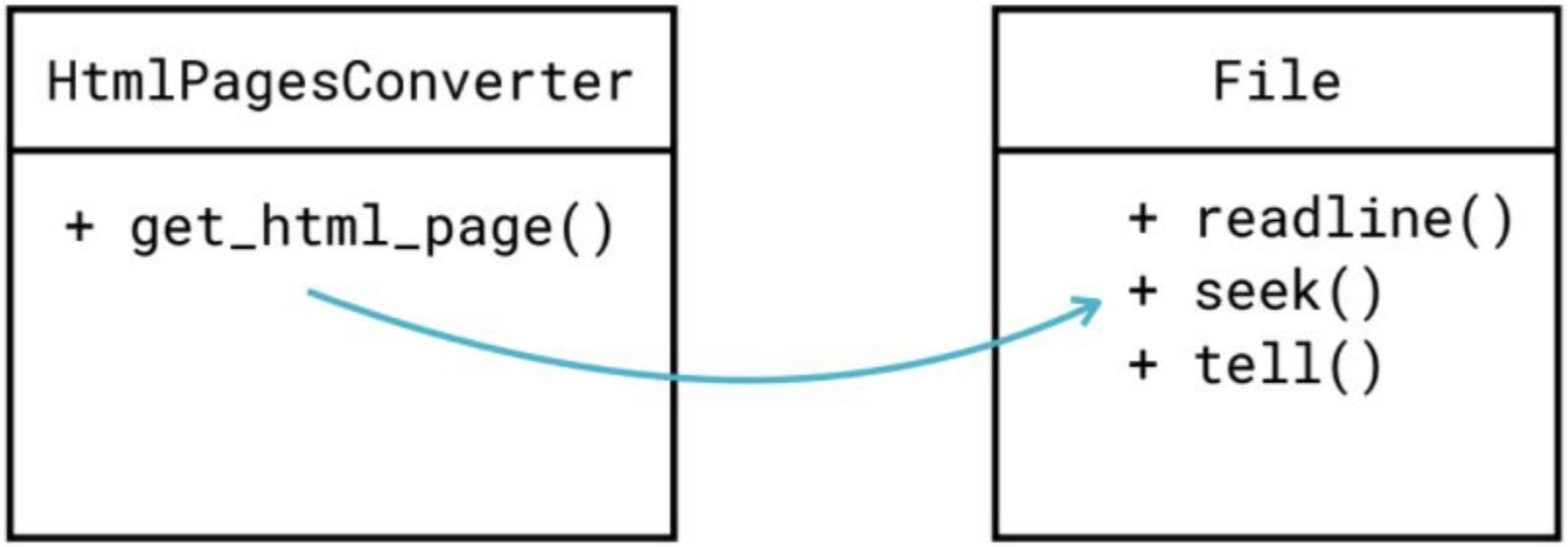


Racing Car Example

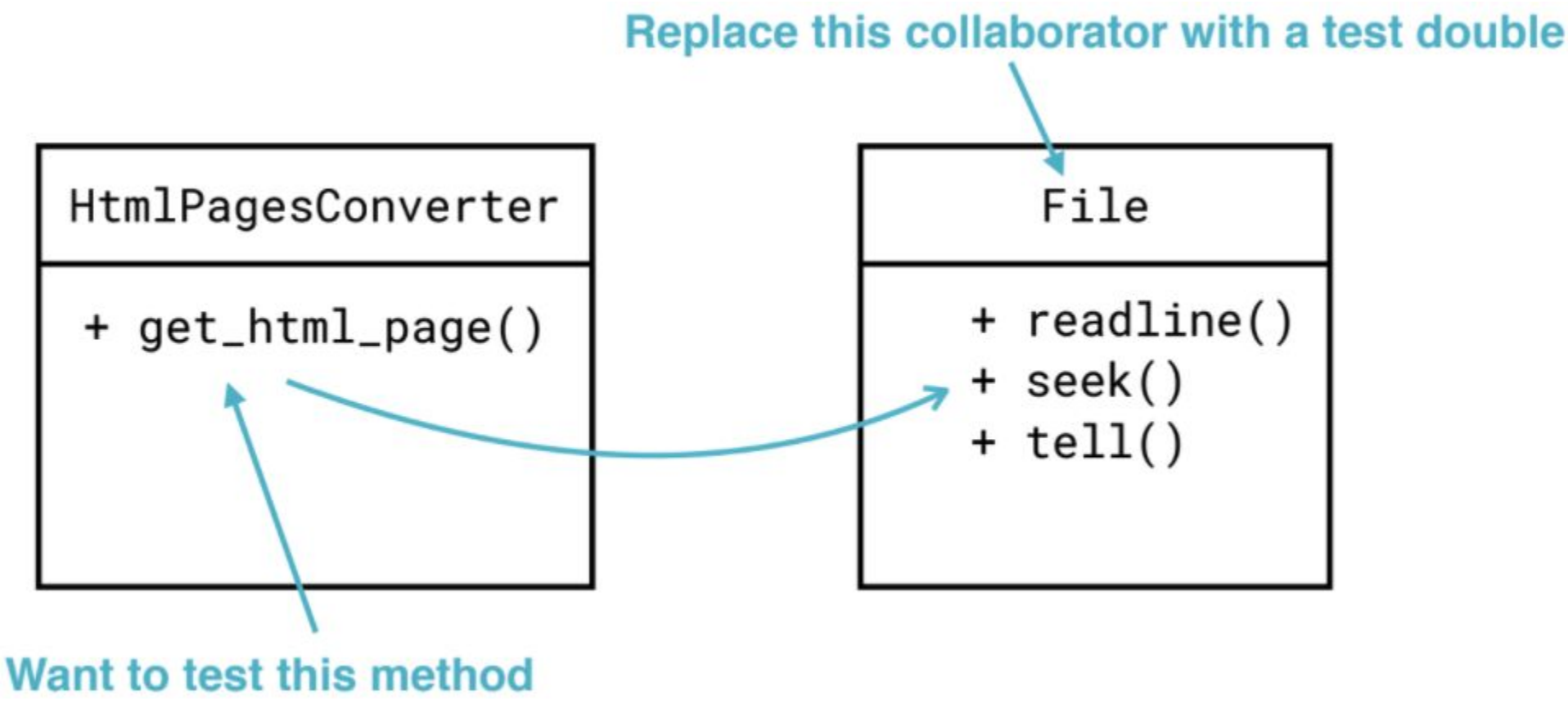


A Stub has the same methods as the class it replaces,
but the implementation is very simple

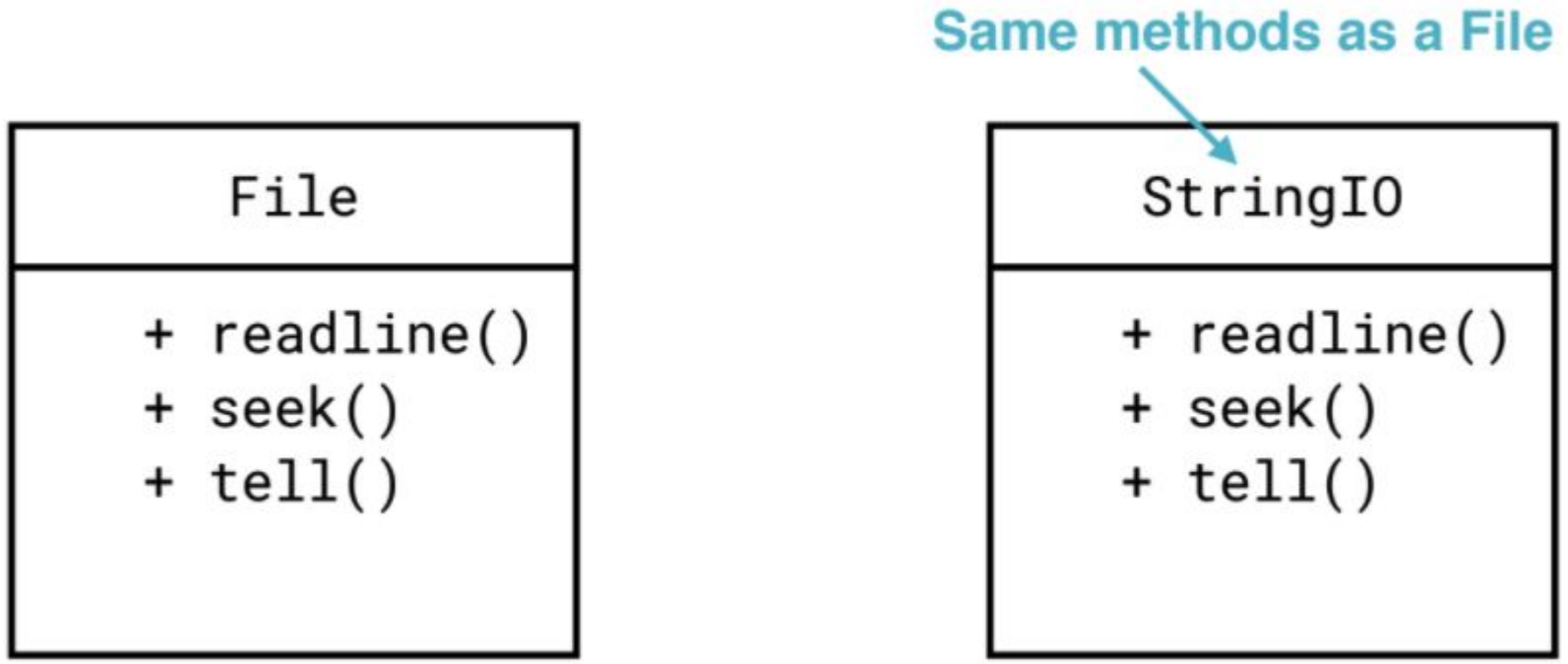
Html Converter Example



Html Converter Example

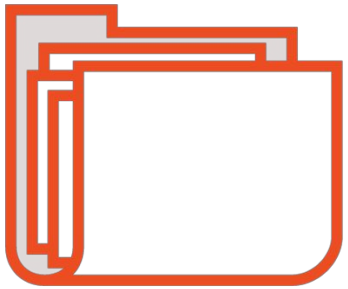


Html Converter Example



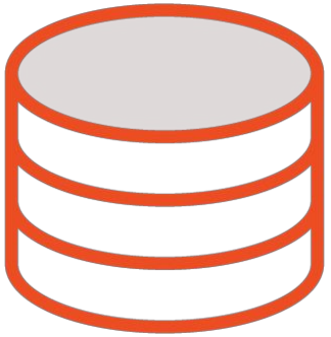
A Fake has an implementation with logic and behaviour, but is not suitable for production

Common Things to Replace with a Fake



File

Replace with StringIO



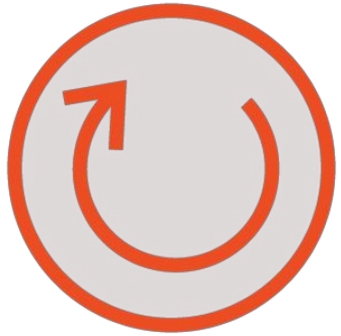
Database

Replace with in-memory database

WebServer

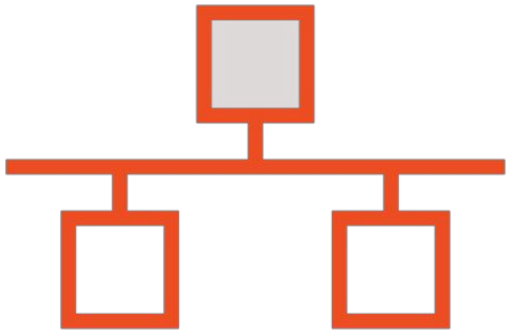
Replace with lightweight Web Server

Three Kinds of Assert



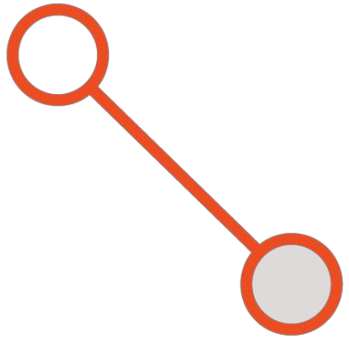
Return value

or an exception



State change

Use an API to query the new state



Method call

Did a specific method get called on a collaborator

Increasing complexity

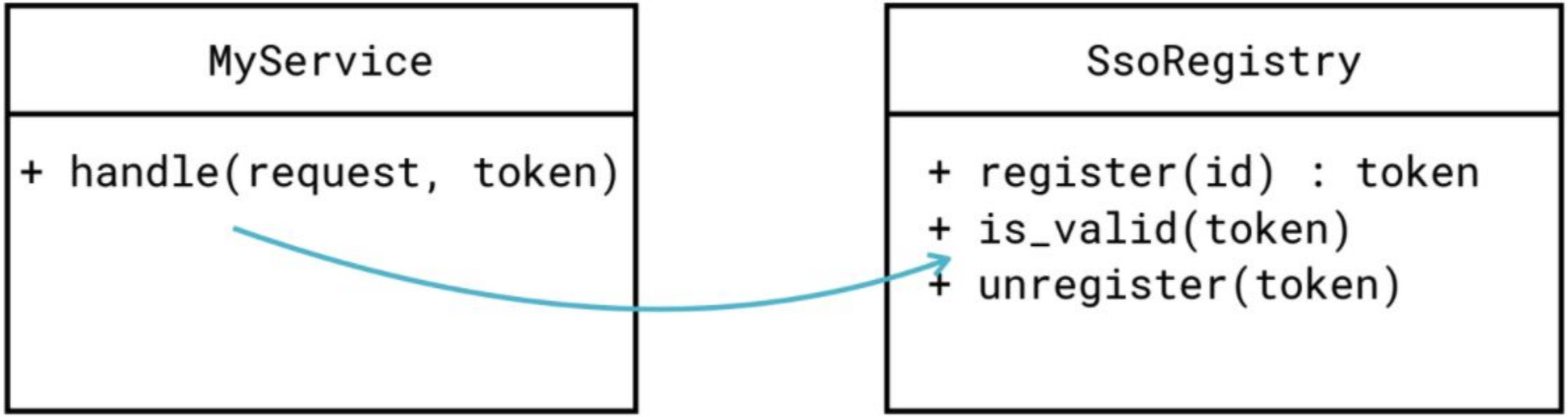
```
def test_convert_quotes():
    fake_file = io.StringIO("quote: ' ")
    converter = HtmlPagesConverter(open_file=fake_file)
    converted_text = converter.get_html_page(0)
    assert converted_text == "quote: &#x27;<br />"
```

Assert on the method return value

```
def test_alarm_is_off_by_default(self):
    alarm = Alarm()
    assert not alarm.is_alarm_on
```

Assert on the alarm state

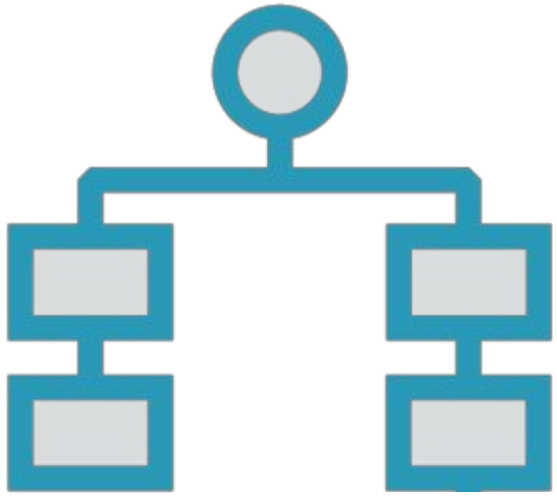
Single Sign On Example



Parameterised Tests

Branch Coverage

Conditional

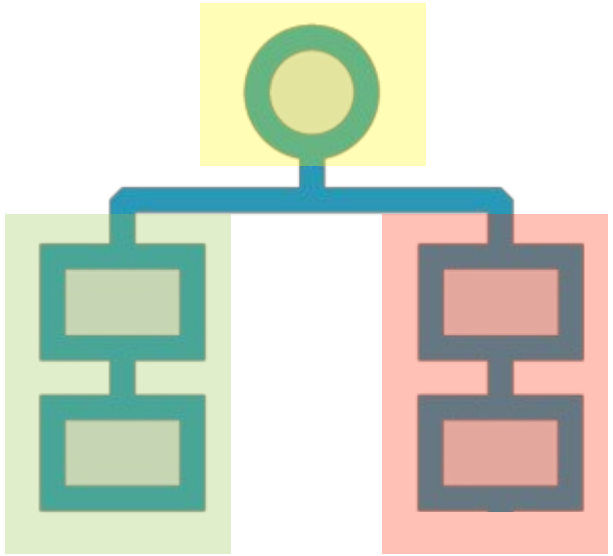


Statements executed when True

Statements executed when False

Branch Coverage

Conditional

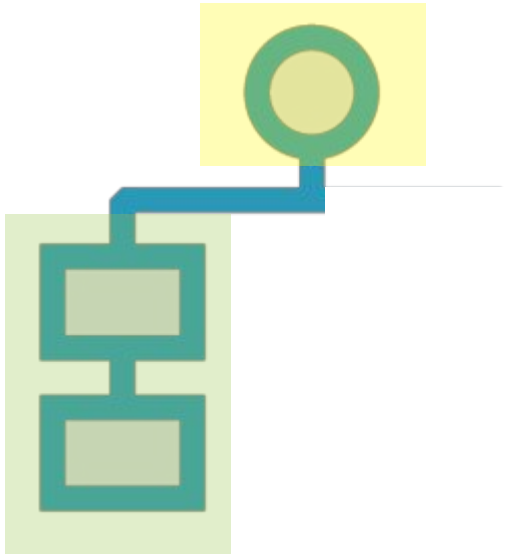


Statements executed when True

Statements executed when False

Branch Coverage

Conditional



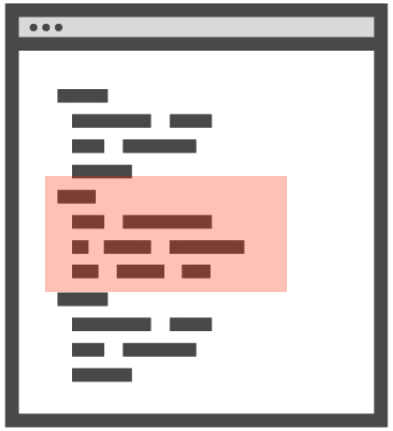
Statements executed when True

Statements executed when False

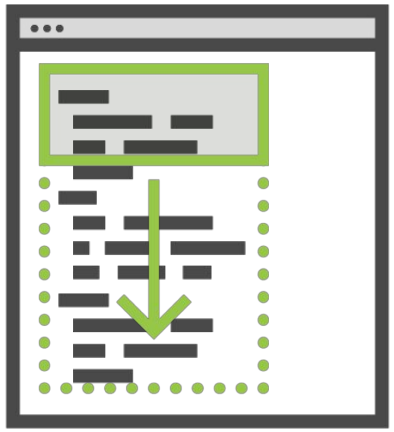
Gilded Rose Example

```
def update_quality(self):
    for item in self.items:
        if item.name != "Aged Brie" and item.name != "Backstage
            if item.quality > 0:
                if item.name != "Sulfuras, Hand of Ragnaros":
                    item.quality = item.quality - 1
        else:
            if item.quality < 50:
                item.quality = item.quality + 1
                if item.name == "Backstage passes to a TAFKAL80E
                    if item.sell_in < 11:
                        if item.quality < 50:
                            item.quality = item.quality + 1
                    if item.sell_in < 6:
                        if item.quality < 50:
                            item.quality = item.quality + 1
        if item.name != "Sulfuras, Hand of Ragnaros":
            item.sell_in = item.sell_in - 1
        if item.sell_in < 0:
            if item.name != "Aged Brie":
```

Situations to use Test Coverage

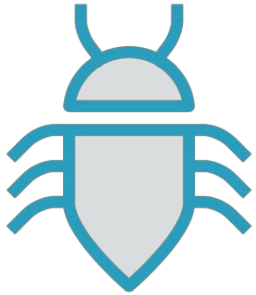


Spot missing tests for new code



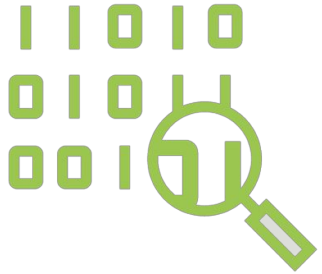
Adding tests to existing code

Evaluating Test Quality



Bugs in production

Confidence to refactor



Code Review

Mutation testing