

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»
Отчет по лабораторной работе №5-6
«Модульное тестирование в Python»

Выполнил:

студент группы ИУ5-33Б

Багин М. В.

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю. Е.

2024 г.

Задание:

1. Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.
2. Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
3. Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:
 - TDD - фреймворк (не менее 3 тестов).
 - BDD - фреймворк (не менее 3 тестов).
 - Создание Моск-объектов (необязательное дополнительное задание).

Текст программы

singleton_meta.py

```
class SingletonMeta(type):
    _instances = {}

    def __call__(cls, *args, **kwargs):
        if cls not in cls._instances:
            instance = super().__call__(*args, **kwargs)
            cls._instances[cls] = instance
        return cls._instances[cls]
```

adapter.py

```
class TrafficLightAdapter:
    def __init__(self, traffic_light):
        self.traffic_light = traffic_light

    def switch(self):
        self.traffic_light.change_state()

    def current_light(self):
        return self.traffic_light.get_state()
```

state.py

```
class State:
    def next(self, context):
        raise NotImplementedError

    def __str__(self):
        raise NotImplementedError

class Red(State):
    def next(self, context):
        context.set_state(Yellow())
        print("Switching from RED to YELLOW")

    def __str__(self):
        return "RED"
```

```

class Yellow(State):
    def next(self, context):
        context.set_state(Green())
        print("Switching from YELLOW to GREEN")

    def __str__(self):
        return "YELLOW"

```

```

class Green(State):
    def next(self, context):
        context.set_state(Red())
        print("Switching from GREEN to RED")

    def __str__(self):
        return "GREEN"

```

traffic_light.py

```

from state import Red
from singleton_meta import SingletonMeta

```

```

class TrafficLight(metaclass=SingletonMeta):
    def __init__(self):
        self.state = Red()

    def set_state(self, state):
        self.state = state

    def change_state(self):
        self.state.next(self)

    def get_state(self):
        return str(self.state)

```

test_traffic_light.py

```

import unittest
from unittest.mock import patch
from traffic_light import TrafficLight
from adapter import TrafficLightAdapter

```

```

class TestTrafficLight(unittest.TestCase):
    def setUp(self):
        self.traffic_light = TrafficLight()
        self.adapter = TrafficLightAdapter(self.traffic_light)

    def test_singleton(self):
        traffic_light2 = TrafficLight()

```

```
self.assertIs(self.traffic_light, traffic_light2, "TrafficLight is
not a singleton")

def test_state_transition(self):
    with patch('builtins.print') as mocked_print:
        self.assertEqual(self.traffic_light.get_state(), "RED")
        self.traffic_light.change_state()
        mocked_print.assert_called_with("Switching from RED to YELLOW")
        self.assertEqual(self.traffic_light.get_state(), "YELLOW")
        self.traffic_light.change_state()
        mocked_print.assert_called_with("Switching from YELLOW to GREEN")
        self.assertEqual(self.traffic_light.get_state(), "GREEN")
        self.traffic_light.change_state()
        mocked_print.assert_called_with("Switching from GREEN to RED")
        self.assertEqual(self.traffic_light.get_state(), "RED")

def test_adapter(self):
    with patch('builtins.print') as mocked_print:
        self.assertEqual(self.adapter.current_light(), "RED")
        self.adapter.switch()
        mocked_print.assert_called_with("Switching from RED to YELLOW")
        self.assertEqual(self.adapter.current_light(), "YELLOW")
```

Анализ результатов

test_singleton

```
Run Python tests for test_traffic_light.TestTrafficLight.test_... x
Test Results 0 ms Tests passed: 1 of 1 test - 0 ms
/Users/nachernev/Desktop/lab_python_test/pythonProject/.venv/bin/python /Users/nachernev/Applications/PyCharm Community Edition.app/C
Testing started at 15:04 ...
Launching unittests with arguments python -m unittest test_traffic_light.TestTrafficLight.test_singleton in /Users/nachernev/Desktop/

Ran 1 test in 0.000s

OK

Process finished with exit code 0
```

test_state_transition

```
Run Python tests for test_traffic_light.TestTrafficLight.test_... x
Test Results 1 ms Tests passed: 1 of 1 test - 1 ms
/Users/nachernev/Desktop/lab_python_test/pythonProject/.venv/bin/python /Users/nachernev/Applications/PyCharm Community Edition.app/C
Testing started at 15:06 ...
Launching unittests with arguments python -m unittest test_traffic_light.TestTrafficLight.test_state_transition in /Users/nachernev/D

Ran 1 test in 0.001s

OK

Process finished with exit code 0
```

test_adapter

```
Run Python tests for test_traffic_light.TestTrafficLight.test_... x
Test Results 0 ms Tests passed: 1 of 1 test - 0 ms
/Users/nachernev/Desktop/lab_python_test/pythonProject/.venv/bin/python /Users/nachernev/Applications/PyCharm Community Edition.app/C
Testing started at 15:06 ...
Launching unittests with arguments python -m unittest test_traffic_light.TestTrafficLight.test_adapter in /Users/nachernev/Desktop/La

Ran 1 test in 0.001s

OK

Process finished with exit code 0
```