Московский государственный технический

университет им. Н.Э. Баумана

Факультет ИУ
Кафедра ИУ5

Курс «Основы информатики»

Отчет по рубежному контролю №2

Вариант 10

Выполнил:
студент группы ИУ5-33Б:
Номоконов В.А.
Подпись и дата:

Проверил:
преподаватель каф. _
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2024 г.

Постановка задачи

Рубежный контроль представляет собой разработку тестов на языке Python.

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

Текст программы

\files_for_rk2\refactoringCode.py

```
class Browser:
    def __init__(self, id, name, version, year):
        self.id = id
        self.name = name
        self.version = version
        self.year = year
        self.computer_id = None


class Computer:
    def __init__(self, id, model, ram):
        self.id = id
        self.model = model
        self.ram = ram
        self.browsers = []


def create_browsers():
    return [
        Browser(1, "Google Chrome", "90.0.4430.212", 2024),
        Browser(2, "Mozilla Firefox", "88.0.1", 2021),
```

```
        Browser(3, "Microsoft Edge", "90.0.818.42", 2024),
        Browser(4, "Safari", "14.0.3", 2300),
        Browser(5, "Opera", "76.0.4017.107", 2021),
        Browser(6, "Atom", "26.0.0.21", 1812),
        Browser(7, "Vivaldi", "5.5.0.0", 2023),
    ]

def create_computers():
    return [
        Computer(1, "Dell Inspiron", 16),
        Computer(2, "HP Envy", 8),
        Computer(3, "Apple MacBook", 16),
        Computer(4, "Apple MacBook Pro", 32),
        Computer(5, "Apple MacBook Air", 16)
    ]

def link_browsers_with_computers(browsers, computers):
    browsers[0].computer_id = computers[0].id
    browsers[1].computer_id = computers[0].id
    browsers[2].computer_id = computers[1].id
    browsers[3].computer_id = computers[2].id
    browsers[4].computer_id = computers[2].id
    browsers[5].computer_id = computers[3].id
    browsers[6].computer_id = computers[3].id

    computers[0].browsers = [browsers[0], browsers[1]]
    computers[1].browsers = [browsers[2]]
    computers[2].browsers = [browsers[3], browsers[4]]
    computers[3].browsers = [browsers[5], browsers[6]]
```

```
def query_1(browsers, computers):
    return [
        (browser.name, next((computer.model for computer in computers if computer.id == browser.computer_id), None))
        for browser in browsers
        if browser.name.startswith("A") and browser.computer_id is not None
    ]


def query_2(computers):
    return sorted(
        [(computer.model, min((browser.year for browser in computer.browsers), default=None)) for computer in computers],
        key=lambda x: x[1] if x[1] is not None else float('inf'),
    )


def query_3(browsers, computers):
    return sorted(
        [(browser.name, next((computer.model for computer in computers if computer.id == browser.computer_id), None), min(browser.year for browser in browsers if browser.computer_id == next((computer.id for computer in computers if computer.model == next((computer.model for computer in computers if computer.id == browser.computer_id), None)), None)))
         for browser in browsers
         if browser.computer_id is not None],
        key=lambda x: x[0],
    )
```

\files_for_rk2\RK2(tests).py

```
import unittest
from refactoringCode import query_1, query_2, query_3, create_browsers, create_computers, link_browsers_with_computers
```

```
class TestRefactoringCode(unittest.TestCase):
    def test_query_1(self):
        browsers = create_browsers()
        computers = create_computers()
        link_browsers_with_computers(browsers, computers)
        result = query_1(browsers, computers)
        self.assertEqual(len(result), 1)
        self.assertEqual(result[0][0], "Atom")

    def test_query_2(self):
        browsers = create_browsers()
        computers = create_computers()
        link_browsers_with_computers(browsers, computers)
        result = query_2(computers)
        self.assertEqual(len(result), 5)
        self.assertEqual(result[0][0], "Dell Inspiron")
        self.assertEqual(result[1][0], "HP Envy")
        self.assertEqual(result[2][0], "Apple MacBook")
        self.assertEqual(result[3][0], "Apple MacBook Pro")
        self.assertEqual(result[4][0], "Apple MacBook Air")

    def test_query_3(self):
        browsers = create_browsers()
        computers = create_computers()
        link_browsers_with_computers(browsers, computers)
        result = query_3(browsers, computers)
        self.assertEqual(len(result), 7)
        self.assertEqual(result[0][0], "Atom")
```

```
        self.assertEqual(result[1][0], "Google Chrome")
        self.assertEqual(result[2][0], "Microsoft Edge")
        self.assertEqual(result[3][0], "Mozilla Firefox")
        self.assertEqual(result[4][0], "Opera")
        self.assertEqual(result[5][0], "Safari")
        self.assertEqual(result[6][0], "Vivaldi")

if __name__ == "__main__":
    unittest.main()
```

Анализ результатов: