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

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчёт по рубежному контролю №2

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б |  | преподаватель каф. ИУ5 |
| Милевич Артём Андреевич |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021 г.

# Описание задания

(Вариант предметной области - 12, вариант запросов - Б)

1. Провести рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.
2. Для текста программы рубежного контроля №1 создать модульные тесты с применением TDD - фреймворка (3 теста).

# Текст программы

Файл **main.py**:

from operator import itemgetter

class Language:

    """Язык программирования"""

    def *\_\_init\_\_*(*self*, id, title, creator, tool\_id):

*self*.id = id

*self*.title = title

*self*.creator = creator

*self*.tool\_id = tool\_id

class Tool:

    """Средство разработки"""

    def *\_\_init\_\_*(*self*, id, name):

*self*.id = id

*self*.name = name

class LangTool:

    """

    'Язык программирования' для реализации

    связи многие-ко-многим

    """

    def *\_\_init\_\_*(*self*, tool\_id, lang\_id):

*self*.tool\_id = tool\_id

*self*.lang\_id = lang\_id

def task1(languages, tools):

    one\_to\_many = [(l.title, l.creator, t.name)

        for t in tools

        for l in languages

        if l.tool\_id == t.id]

    return sorted(one\_to\_many, key=itemgetter(2))

def task2(languages, tools):

    res = list()

    one\_to\_many = [(l.title, l.creator, t.name)

        for t in tools

        for l in languages

        if l.tool\_id == t.id]

    for t in tools:

        # Список языков в средствах разработки

        t\_leng = list(filter(lambda i: i[2] == t.name, one\_to\_many))

        # Если средства разработки не пустые

        if len(t\_leng) > 0:

            res.append((t.name, len(t\_leng)))

    return sorted(res, key=itemgetter(1), reverse=True)

def task3(languages, tools, lang\_tool):

    res = dict()

    many\_to\_many = [(l.title, l.creator, t.name)

        for t in tools

        for l in languages

        for relation in lang\_tool

        if t.id == relation.tool\_id and l.id == relation.lang\_id]

    for l in languages:

        if l.title.endswith("ов"):

            # Ищем средства разработки конкретного языка

            l\_tools = list(filter(lambda x: x[0] == l.title, many\_to\_many))

            # Получаем их названия

            l\_tools\_names = [x[2] for x in l\_tools]

            res[l.title] = l\_tools\_names

    return res

def main():

    tools = [

    Tool(1, 'Visual Studio'),

    Tool(2, 'XCode'),

    Tool(3, 'Visual Studio Code'),

    Tool(4, 'Netbeans'),

]

    # Языки программирования

    languages = [

        Language(1, 'Python', 'Гвидо ван Россум', 1),

        Language(2, 'C++\_ов', 'Бьёрн Страуструп', 2),

        Language(3, 'C', 'Деннис Ритчи', 3),

        Language(4, 'Pascal\_ов', 'Никлаус Вирт', 3),

        Language(5, 'Java', 'Джеймс Гослинг', 4),

    ]

    lang\_tool = [

        LangTool(1,1),

        LangTool(1,2),

        LangTool(1,3),

        LangTool(2,2),

        LangTool(2,4),

        LangTool(3,1),

        LangTool(3,3),

        LangTool(3,4),

    ]

    print('Задание Б1')

    res\_1 = task1(languages, tools)

    for res in res\_1:

        print(res)

    print('\nЗадание Б2')

    res\_2 = task2(languages, tools)

    [print(el) for el in res\_2]

    print("\nЗадание Б3")

    res\_3 = task3(languages, tools, lang\_tool)

    [print(k, v) for k, v in res\_3.items()]

if *\_\_name\_\_* == '\_\_main\_\_':

    main()

Файл **test.py**:

import unittest

from main import Tool, Language, LangTool, task2, task1, task3

class Test(unittest.TestCase):

    def setUp(*self*):

*self*.tools = [

        Tool(1, 'Visual Studio'),

        Tool(2, 'XCode'),

        Tool(3, 'Visual Studio Code'),

        Tool(4, 'Netbeans'),

        ]

*self*.languages = [

            Language(1, 'Python', 'Гвидо ван Россум', 1),

            Language(2, 'C++\_ов', 'Бьёрн Страуструп', 2),

            Language(3, 'C', 'Деннис Ритчи', 3),

            Language(4, 'Pascal\_ов', 'Никлаус Вирт', 3),

            Language(5, 'Java', 'Джеймс Гослинг', 4),

        ]

*self*.lang\_tool = [

            LangTool(1,1),

            LangTool(1,2),

            LangTool(1,3),

            LangTool(2,2),

            LangTool(2,4),

            LangTool(3,1),

            LangTool(3,3),

            LangTool(3,4),

        ]

    def test\_task1(*self*):

        keys = [

            ('Java', 'Джеймс Гослинг', 'Netbeans'),

            ('Python', 'Гвидо ван Россум', 'Visual Studio'),

            ('C', 'Деннис Ритчи', 'Visual Studio Code'),

            ('Pascal\_ов', 'Никлаус Вирт', 'Visual Studio Code'),

            ('C++\_ов', 'Бьёрн Страуструп', 'XCode'),

            ]

*self*.assertEqual(task1(*self*.languages, *self*.tools), keys)

    def test\_task2(*self*):

        keys = [

            ('Visual Studio Code', 2),

            ('Visual Studio', 1),

            ('XCode', 1),

            ('Netbeans', 1),

            ]

*self*.assertEqual(task2(*self*.languages, *self*.tools), keys)

    def test\_task3(*self*):

        keys = {

            "C++\_ов": ['Visual Studio', 'XCode'],

            "Pascal\_ов": ['XCode', 'Visual Studio Code'],

        }

*self*.assertEqual(task3(*self*.languages, *self*.tools, *self*.lang\_tool), keys)

if *\_\_name\_\_* == '\_\_main\_\_':

    unittest.main()

# Результат выполнения программы

