|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

Факультет «Информатика и системы управления»

ДИСЦИПЛИНА:

«БКИТ»

**Рубежная контроль № 2**

Студент Распашнов А.А. ИУ5Ц-54Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(И.О. Фамилия) (Группа) (Подпись, дата)

Преподаватель Гапанюк Ю.Е.  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(И.О. Фамилия) (Подпись, дата)

**Условия рубежного контроля №2 по курсу БКИТ**

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).

**Main.py:**

import unittest

class Oper:

    def \_\_init\_\_(self, id, name, salary, lang\_id):

        self.id = id

        self.name = name

        self.salary = salary

        self.lang\_id = lang\_id

class Lang:

    def \_\_init\_\_(self, id, name):

        self.id = id

        self.name = name

class OperLang:

    def \_\_init\_\_(self, oper\_id, lang\_id):

        self.oper\_id = oper\_id

        self.lang\_id = lang\_id

languages = [

    Lang(1, 'Python'),

    Lang(2, 'Java'),

    Lang(3, 'AspectJ'),

    Lang(4, 'C++'),

    Lang(5, 'C'),

    Lang(6, 'Go'),

    Lang(7, 'Ada')

]

operators = [

    Oper(1, 'Maxim', 1000, 1),

    Oper(2, 'Andrey', 2000, 2),

    Oper(3, 'Daniel', 1500, 1),

    Oper(4, 'Viktor', 3000, 5),

    Oper(5, 'Misha', 2500, 4),

    Oper(6, 'Dima', 500, 7),

    Oper(7, 'Rebekka', 300, 3)

]

opers\_langs = [

    OperLang(1, 1),

    OperLang(1, 5),

    OperLang(1, 3),

    OperLang(2, 2),

    OperLang(2, 3),

    OperLang(3, 1),

    OperLang(4, 2),

    OperLang(4, 3),

    OperLang(6, 1),

    OperLang(6, 5)

]

def filter\_letter(one\_to\_many, letter):

    return [res for res in filter(lambda entry: entry[2][0] == letter, one\_to\_many)]

def sorted\_max\_salary(one\_to\_many):

    return sorted([(lang.name, max([salary for \_, salary, lang\_name in one\_to\_many if lang\_name == lang.name]))

                                 for lang in languages if len(list(filter(lambda entry: entry[2] == lang.name, one\_to\_many))) > 0],

                                key=lambda x: x[1],

                                reverse=True)

def links\_sorted\_langs(many\_to\_many):

    return sorted(many\_to\_many, key=lambda entry: entry[2])

def main():

    one\_to\_many = [(oper.name, oper.salary, lang.name)

                                 for lang in languages

                                 for oper in operators

                                 if oper.lang\_id == lang.id]

    many\_to\_many\_temp = [(lang.name, ol.oper\_id, ol.lang\_id)

                                             for lang in languages

                                             for ol in opers\_langs

                                             if lang.id == ol.lang\_id]

    many\_to\_many = [(oper.name, oper.salary, lang\_name)

                                    for lang\_name, \_, lang\_id in many\_to\_many\_temp

                                    for oper in operators if oper.lang\_id == lang\_id]

    print('Задание Г1')

    print(filter\_letter(one\_to\_many, 'A'))

    print('\nЗадание Г2')

    print(sorted\_max\_salary(one\_to\_many))

    print('\nЗадание Г3')

    print(links\_sorted\_langs(many\_to\_many))

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**test.py**

import unittest

from main import \*

one\_to\_many = [(oper.name, oper.salary, lang.name)

                                 for lang in languages

                                 for oper in operators

                                 if oper.lang\_id == lang.id]

many\_to\_many\_temp = [(lang.name, ol.oper\_id, ol.lang\_id)

                                         for lang in languages

                                         for ol in opers\_langs

                                         if lang.id == ol.lang\_id]

many\_to\_many = [(oper.name, oper.salary, lang\_name)

                                for lang\_name, \_, lang\_id in many\_to\_many\_temp

                                for oper in operators if oper.lang\_id == lang\_id]

class Tests(unittest.TestCase):

    def test\_filter\_letter(self):

        self.assertEqual(filter\_letter(one\_to\_many, 'A'), [('Rebekka', 300, 'AspectJ'), ('Dima', 500, 'Ada')])

    def test\_sorted\_max\_salary(self):

        self.assertEqual(sorted\_max\_salary(one\_to\_many), [('C', 3000), ('C++', 2500), ('Java', 2000), ('Python', 1500),

                                                                                                            ('Ada', 500), ('AspectJ', 300)])

    def test\_links\_sorted\_langs(self):

        self.assertEqual(links\_sorted\_langs(many\_to\_many), [('Rebekka', 300, 'AspectJ'), ('Rebekka', 300, 'AspectJ'),

                                                                                                             ('Rebekka', 300, 'AspectJ'), ('Viktor', 3000, 'C'),

                                                                                                             ('Viktor', 3000, 'C'), ('Andrey', 2000, 'Java'),

                                                                                                             ('Andrey', 2000, 'Java'), ('Maxim', 1000, 'Python'),

                                                                                                             ('Daniel', 1500, 'Python'), ('Maxim', 1000, 'Python'),

                                                                                                             ('Daniel', 1500, 'Python'), ('Maxim', 1000, 'Python'),

                                                                                                             ('Daniel', 1500, 'Python')])

if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

**Результат:**

