**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по рубежному контролю №2 по курсу БКИТ

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-31Б |  | преподаватель каф. ИУ5 |
| Фотчин Алексей Дмитриевич |  | Гапанюк Юрий Евгеньевич |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021 г.

**Задание:**

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:**

from operator import itemgetter  
  
class Oper:  
 *"""Оператор"""* def \_\_init\_\_(self, id, Oper\_name, lines, Lang\_id):  
 self.Oper\_id = id  
 self.Oper\_name = Oper\_name  
 self.lines = lines #Кол-во строк  
 self.Lang\_id = Lang\_id  
  
class Lang:  
 *"""Язык программирования"""* def \_\_init\_\_(self, id, name):  
 self.Lang\_id = id  
 self.Lang\_name = name  
  
class Oper\_Lang:  
 *"""'Операторы языков' для реализации связи многие-ко-многим"""* def \_\_init\_\_(self, Lang\_id, Oper\_id):  
 self.Lang\_id = Lang\_id  
 self.Oper\_id = Oper\_id  
# Языки  
Langs = [  
 Lang(1, 'Python'),  
 Lang(2, 'C++ Lang'),  
 Lang(3, 'C# Lang'),  
 Lang(3, 'C# Lang'),  
  
 Lang(11, 'Basic Lang'),  
 Lang(22, 'Java'),  
 Lang(33, 'Pascal'),  
]  
# Операторы  
Opers = [  
 Oper(1, 'Оператор сравнения', 3, 1),  
 Oper(2, 'Оператор присваивания', 7, 2),  
 Oper(3, 'Оператор сложения', 4, 3),  
 Oper(4, 'Оператор разности', 10, 3),  
 Oper(5, 'Оператор умножения', 16, 3),  
]  
Opers\_Langs = [  
 Oper\_Lang(1, 1),  
 Oper\_Lang(2, 2),  
 Oper\_Lang(3, 3),  
 Oper\_Lang(3, 4),  
 Oper\_Lang(3, 5),  
  
 Oper\_Lang(11, 1),  
 Oper\_Lang(22, 2),  
 Oper\_Lang(33, 3),  
 Oper\_Lang(33, 4),  
 Oper\_Lang(33, 5),  
]  
  
  
def one\_to\_many\_def(Langs, Opers):  
 one\_to\_many = [(e.Oper\_name, e.lines, d.Lang\_name)  
 for d in Langs  
 for e in Opers  
 if e.Lang\_id == d.Lang\_id]  
 return one\_to\_many  
  
def res\_12\_def(Langs, one\_to\_many):  
 res\_12\_unsorted = []  
 for d in Langs:  
 # Сортированный список языков по их имени  
 d\_Langs = list(filter(lambda i: i[2] == d.Lang\_name, one\_to\_many))  
 if len(d\_Langs) > 0:  
 d\_Opers = [line for \_, line, \_ in d\_Langs]  
 # Суммарное количество сторок операторов языка  
 d\_Lines\_sum = sum(d\_Opers)  
 res\_12\_unsorted.append((d.Lang\_name, d\_Lines\_sum))  
 # Сортировка по строкам  
 res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)  
 return (res\_12)  
  
def res\_13\_def(Langs, many\_to\_many):  
 res\_13 = {}  
 for d in Langs:  
 if 'Lang' in d.Lang\_name:  
 # Список операторов языка программирования  
 d\_Langs = list(filter(lambda i: i[2] == d.Lang\_name, many\_to\_many))  
 # Только название операторов  
 d\_Opers\_names = [x for x, \_, \_ in d\_Langs]  
 # Добавляем результат в словарь  
 # ключ - язык, значение - список имен операторов  
 res\_13[d.Lang\_name] = d\_Opers\_names  
 return (res\_13)  
  
def many\_to\_many\_def(Langs, Opers):  
 many\_to\_many\_temp = [(d.Lang\_name, ed.Lang\_id, ed.Oper\_id)  
 for d in Langs  
 for ed in Opers\_Langs  
 if d.Lang\_id == ed.Lang\_id]  
  
 many\_to\_many = [(e.Oper\_name, e.lines, Lang\_name)  
 for Lang\_name, dep\_id, Lib\_id in many\_to\_many\_temp  
 for e in Opers if e.Oper\_id == Lib\_id]  
 return(many\_to\_many)  
  
def main():  
 *"""Основная функция"""* # Соединение данных один-ко-многим  
 one\_to\_many = one\_to\_many\_def(Langs, Opers)  
 # Соединение данных многие-ко-многим  
 many\_to\_many = many\_to\_many\_def(Langs, Opers)  
  
 print('Задание А1')  
 res\_11 = sorted(one\_to\_many, key=itemgetter(2))  
 print(res\_11)  
 print("#Список связанных операторов и языков, отсортированный по языкам")  
  
 print('\nЗадание А2')  
 res\_12\_unsorted = []  
 for d in Langs:  
 # Сортированный список языков по их имени  
 d\_Langs = list(filter(lambda i: i[2] == d.Lang\_name, one\_to\_many))  
 if len(d\_Langs) > 0:  
 d\_Opers = [line for \_, line, \_ in d\_Langs]  
 # Суммарное количество сторок операторов языка  
 d\_Lines\_sum = sum(d\_Opers)  
 res\_12\_unsorted.append((d.Lang\_name, d\_Lines\_sum))  
 # Сортировка по строкам  
 res\_12 = res\_12\_def(Langs, one\_to\_many)  
 print(res\_12)  
 print("#Список языков программирования и суммарного количества их строк,")  
 print("отсортированный по количеству строк")  
  
 print('\nЗадание А3')  
 res\_13 = res\_13\_def(Langs, many\_to\_many)  
 print(res\_13)  
 print("#Список всех языков, у которых в названии есть 'Lang' и список всех их операторов")  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

import unittest  
from main import \*  
  
class MyTestCase(unittest.TestCase):  
 def test\_something(self):  
 self.assertNotEqual(True, False) # add assertion here  
  
 def test\_language(self):  
 self.assertEqual(Langs[2].Lang\_name, Langs[3].Lang\_name )  
  
 def test\_A1(self):  
 self.assertEqual(sorted(one\_to\_many\_def(Langs, Opers), key=itemgetter(2)), [('Оператор сложения', 4, 'C# Lang'), ('Оператор разности', 10, 'C# Lang'), ('Оператор умножения', 16, 'C# Lang'), ('Оператор сложения', 4, 'C# Lang'), ('Оператор разности', 10, 'C# Lang'), ('Оператор умножения', 16, 'C# Lang'), ('Оператор присваивания', 7, 'C++ Lang'), ('Оператор сравнения', 3, 'Python')])  
  
 def test\_A2(self):  
 self.assertEqual([('C# Lang', 60), ('C# Lang', 60), ('C++ Lang', 7), ('Python', 3)], res\_12\_def(Langs, one\_to\_many\_def(Langs, Opers)))  
  
 def test\_A3(self):  
 self.assertEqual({'C++ Lang': ['Оператор присваивания'], 'C# Lang': ['Оператор сложения', 'Оператор разности', 'Оператор умножения', 'Оператор сложения', 'Оператор разности', 'Оператор умножения'], 'Basic Lang': ['Оператор сравнения']}, res\_13\_def(Langs, many\_to\_many\_def(Langs, Opers)))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

Результат выполнения программы:

