**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и управление»

Кафедра ИУ5 «Системы обработки информации и управления»

Рубежный контроль №2

## по дисциплине «Базовые компоненты интернет-технологий»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-33 |  | преподаватель каф. ИУ5 |
| Ефременко Д.С. |  | Гапанюк Ю.Е. |
|  |  |  |

2021

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:

main.py

# используется для сортировки  
from operator import itemgetter  
  
class Driver:  
 *"""Водитель"""* def \_\_init\_\_(self, id, fio, sal, exp\_id):  
 self.id = id  
 self.fio = fio  
 self.sal = sal  
 self.exp\_id = exp\_id #водительский стаж  
  
class Auto:  
 *"""Автопарк"""* def \_\_init\_\_(self, id, mode):  
 self.id = id  
 self.mode = mode #виды автобусов  
  
class AutoDriver:  
 *"""  
 'Водитель автопарка' для реализации  
 связи многие-ко-многим  
 """* def \_\_init\_\_(self, driv\_id, auto\_id):  
 self.driv\_id = driv\_id  
 self.auto\_id = auto\_id  
  
  
# Автопарк  
auto = [  
 Auto(1, 'туристический'),  
 Auto(2, 'городской'),  
 Auto(3, 'школьный'),  
 Auto(4, 'экскурсионный'),  
 Auto(5, 'перонный'),  
 Auto(6, 'пригородный'),  
]  
  
# Водители  
driv = [  
 Driver(1, 'Ефременко', 50000, 5),  
 Driver(2, 'Семенов', 30000, 3),  
 Driver(3, 'Стебунов', 45000, 4),  
 Driver(4, 'Носкин', 44000, 6),  
 Driver(5, 'Бегларов', 45000, 2),  
 Driver(6, 'Алешин', 23000, 4),  
 Driver(7, 'Ахтамбаев', 40000, 2),  
 Driver(8, 'Андреев', 45000, 3)  
]  
  
auto\_drivers = [  
 AutoDriver(1, 1),  
 AutoDriver(3, 2),  
 AutoDriver(3, 4),  
 AutoDriver(5, 1),  
 AutoDriver(6, 3),  
 AutoDriver(2, 1),  
 AutoDriver(4, 2),  
 AutoDriver(3, 3),  
 AutoDriver(4, 5),  
 AutoDriver(6, 2),  
]  
  
def Task1(one\_to\_many):  
 print('Задание В1')  
 res\_11 = []  
 for fio, sal, auto\_name in one\_to\_many:  
 if 'А' in fio[0]:  
 res\_11.append((fio, auto\_name))  
 return res\_11  
  
def Task2(one\_to\_many):  
 print('Задание В2')  
 buff = []  
 for a in auto:  
 # список видов транспорта  
 a\_modes = list(filter(lambda i: i[2] == a.mode, one\_to\_many))  
 if len(a\_modes) > 0:  
 a\_sal = [sal for \_, sal, \_ in a\_modes]  
 min\_sal = min(a\_sal)  
 buff.append((a.mode, min\_sal))  
 res\_12 = sorted(buff, key=itemgetter(1))  
 return res\_12  
  
def Task3(many\_to\_many):  
 print('Задание В3')  
 buff = []  
 for fio, sal, auto\_name in many\_to\_many:  
 buff.append((fio, auto\_name))  
 res\_13 = list(sorted(buff, key=itemgetter(0)))  
 return res\_13  
  
def main():  
 *"""Основная функция"""* # Соединение данных один-ко-многим  
 one\_to\_many = [(d.fio, d.sal, a.mode)  
 for a in auto  
 for d in driv  
 if d.exp\_id == a.id]  
  
 # Соединение данных многие-ко-многим  
 many\_to\_many\_temp = [(a.mode, ad.auto\_id, ad.driv\_id)  
 for a in auto  
 for ad in auto\_drivers  
 if a.id == ad.auto\_id]  
  
 many\_to\_many = [(d.fio, d.sal, auto\_name)  
 for auto\_name, auto\_id, driv\_id in many\_to\_many\_temp  
 for d in driv if d.id == driv\_id]  
  
 print('Test')#вывод списков со связями 1-м, м-м  
 res\_0 = (one\_to\_many)  
 print(res\_0)  
 res\_01 = (many\_to\_many)  
 print(res\_01)  
  
 print(Task1(one\_to\_many))  
 print(Task2(one\_to\_many))  
 print(Task3(many\_to\_many))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

test.py

import unittest  
import sys, os  
  
sys.path.append(os.getcwd())  
from main import \*  
  
  
class Test\_task\_1(unittest.TestCase):  
 def test\_task\_1(self):  
 one\_to\_many = [(d.fio, d.sal, a.mode)  
 for a in auto  
 for d in driv  
 if d.exp\_id == a.id]  
 self.assertEqual(Task1(one\_to\_many), [('Ахтамбаев', 'городской'),  
 ('Андреев', 'школьный'), ('Алешин', 'экскурсионный')])  
  
class Test\_task\_2(unittest.TestCase):  
 def test\_task\_2(self):  
 one\_to\_many = [(d.fio, d.sal, a.mode)  
 for a in auto  
 for d in driv  
 if d.exp\_id == a.id]  
 self.assertEqual(Task2(one\_to\_many),[('экскурсионный', 23000), ('школьный', 30000),  
 ('городской', 40000), ('пригородный', 44000), ('перонный', 50000)])  
  
  
class Test\_task\_3(unittest.TestCase):  
 def test\_task\_3(self):  
 many\_to\_many\_temp = [(a.mode, ad.auto\_id, ad.driv\_id)  
 for a in auto  
 for ad in auto\_drivers  
 if a.id == ad.auto\_id]  
 many\_to\_many = [(d.fio, d.sal, auto\_name)  
 for auto\_name, auto\_id, driv\_id in many\_to\_many\_temp  
 for d in driv if d.id == driv\_id]  
 self.assertEqual(Task3(many\_to\_many),  
 [('Алешин', 'городской'), ('Алешин', 'школьный'), ('Бегларов', 'туристический'),  
 ('Ефременко', 'туристический'), ('Носкин', 'городской'), ('Носкин', 'перонный'),  
 ('Семенов', 'туристический'), ('Стебунов', 'городской'), ('Стебунов', 'школьный'),  
 ('Стебунов', 'экскурсионный')])  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()

Результат выполнения программы:

