ИУ5-32Б,

Старкин Александр

**Рубежный контроль 2**

**Условия:**

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**

*from* operator *import* itemgetter

class Creator:

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

*self*.id = *id*

*self*.name = *name*

class Detail:

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

*self*.id = *id*

*self*.name = *name*

*self*.GOST\_id = *GOST\_id*

*self*.creator\_id = *creator\_id*

class CreatorDetail:

    def \_\_init\_\_(*self*, *creator\_id*, *detail\_id*):

*self*.creator\_id = *creator\_id*

*self*.detail\_id = *detail\_id*

Creators = [

    Creator(1, "BMSTU\_RK1"),

    Creator(2, "BMSTU\_IU3"),

    Creator(3, "BMSTU\_L1"),

]

Details = [

    Detail(1, "Soska", 1999\_43,1),

    Detail(2, "Stul", 1945\_21,2),

    Detail(3, "Stol", 1000\_61,3),

    Detail(4, "Podik", 1943\_38, 3),

    Detail(5, "Girya", 1942\_47, 1),

    Detail(6, "Dver", 1999\_42, 1)

]

Creator\_to\_detail = [

    CreatorDetail(1, 1),

    CreatorDetail(2, 2),

    CreatorDetail(3, 3),

    CreatorDetail(3, 4),

    CreatorDetail(1, 5),

]

def first\_task(*details\_list*):

    res\_1 = sorted(*details\_list*, *key*=itemgetter(2))

*return* res\_1

def second\_task(*details\_list*):

    res\_2 = []

    temp\_dict = dict()

*for* i *in* *details\_list*:

*if* i[2] in temp\_dict:

            temp\_dict[i[2]] += 1

*else*:

            temp\_dict[i[2]] = 1

*for* i *in* temp\_dict.keys():

        res\_2.append((i, temp\_dict[i]))

    res\_2.sort(*key*=itemgetter(1), *reverse*=True)

*return* res\_2

def third\_task(*details\_list*, *end\_ch*):

    res\_3 = [(i[0], i[2]) *for* i *in* *details\_list* *if* str(i[1]).endswith(*end\_ch*)]

*return* res\_3

def main():

    one\_to\_many = [(Detail.name, Detail.GOST\_id, Creator.name)

*for* Creator *in* Creators

*for* Detail *in* Details

*if* Detail.creator\_id == Creator.id]

    many\_to\_many\_temp = [(Creator.name, connection.creator\_id, connection.detail\_id)

*for* Creator *in* Creators

*for* connection *in* Creator\_to\_detail

*if* connection.creator\_id == Creator.id]

    many\_to\_many = [(Detail.name, Detail.GOST\_id, Creator\_name)

*for* Creator\_name, Creator\_id, detail\_id *in* many\_to\_many\_temp

*for* Detail *in* Details *if* Detail.id == detail\_id]

    print('Задание Б1')

    print(first\_task(one\_to\_many))

    print("\nЗадание Б2")

    print(second\_task(one\_to\_many))

    print("\nЗадание Б3")

    print(third\_task(many\_to\_many, '1'))

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

    main()

**unit-tests.py**

*import* main

*from* operator *import* itemgetter

*import* unittest

class TestMainMethods(unittest.TestCase):

    def test\_first\_task\_method(*self*):

        test\_list = [('first', 1999\_45, "BMSTU\_RK1"), ('forth', 1924\_24, "BMSTU\_IU3"), ('second', 2024\_352,"BMSTU\_L3")]

        result = main.first\_task(test\_list)

        reference = sorted(test\_list, *key*=itemgetter(2))

*self*.assertEqual(result, reference)

    def test\_second\_task\_method(*self*):

        test\_list = [("Pen", 1999\_43, "John" ),

                    ("Car", 1945\_21, "Alex"),

                    ("Stol", 1000\_61, "Alex"),

                    ("Laptop", 1943\_38, "Alex"),

                    ("Carry", 1942\_47, "John"),

                    ("Lamp", 1999\_42, "Jean")]

        result = main.second\_task(test\_list)

        reference = [('Alex', 3), ('John', 2), ('Jean', 1)]

*self*.assertEqual(result, reference)

    def test\_third\_method(*self*):

        test\_list = [("Soska", 1999\_43,1),

                    ("Stul", 1945\_214,2),

                    ("Stol", 1000\_61,3),

                    ("Podik", 1943\_38, 3),

                    ("Girya", 1942\_474, 1),

                    ("Dver", 1999\_4245, 1)]

        result = main.third\_task(test\_list, '4')

        reference = [('Stul', 2), ('Girya', 1)]

*self*.assertEqual(result, reference)

**Результат запуска:**

