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 РК №2

Выполнил:
студент группы ИУ5-31Б
Зелинский Даниил Михайлович

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич

Подпись и дата:

Подпись и дата:

Москва, 2021 г.

Задание.

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).

Код программы.

objects.py

```
class Building:
    def __init__(self, id, num, people, year):
        self.id=id
        self.num=num
        self.people=people
        self.year=year
        self.streetID=-1
    def set_street(self, streetID):
        self.streetID=streetID
    def __str__(self):
        return f" {self.id} {self.num}  {self.people}  {self.year}
{self.streetID}"

class Street:
    def __init__(self, id,name):
        self.id=id
        self.name=name
    def __str__(self):
        return f"{self.id}  {self.name}"

class BuildStreet:
    def __init__(self, streetID, buildID):
        self.streetID=streetID
        self.buildID=buildID
    def __str__(self):
        return f"{self.streetID}  {self.buildID}"
```

generators.py

```
from objects import *
import json

def one_to_many_generator(streets: list, builds: list):
    return [(s.name, b.num, b.people)
            for b in builds
            for s in streets
            if b.streetID == s.id]

def many_to_many_generator(streets: list, builds: list, buildstreets: list):
    many_to_many_temp = [(s.name, bs.streetID, bs.buildID)
                         for s in streets
                         for bs in buildstreets
                         if s.id == bs.streetID
                         ]
    return [(s_name, b.num, b.people)
            for s_name, streetID, buildID in many_to_many_temp
            for b in builds if b.id == buildID
            ]
```

```
def builds_generator():
    with open('data.json', 'r') as file:
        data = json.load(file)
    builds=[]
    index=0
    for build in data:
        builds.append(Building(index, build["num"], build["people"],
build["year"]))
        index = index + 1
    return builds

def steets_generator():
    streets=[]
    index=0
    with open('streets.json', 'r', encoding='utf-8') as file:
        data=json.load(file)
    for streetname in data:
        streets.append(Street(index, streetname))
        index = index + 1
    return streets

# для построения связи один-ко-многим
# {id улицы : [id зданий]}
def build_placing(input: dict, streets: list, builds: list):
    for streetnumb in range(0, len(streets)):
        if streetnumb in input:
            for buildnumb in input[streetnumb]:
                builds[buildnumb].set_street(streetnumb)

# для построения связи многие-ко-многим
# [id улицы, id здания]
def street_crossing(input: list):
    return [BuildStreet(con[0],con[1]) for con in input]
```

task.py

```
from operator import itemgetter

# выводится название улицы, номер здания на этой улице и количество
работающих в здании людей
def taskA1(one_to_many:list):
    return [i for i in sorted(one_to_many, key=itemgetter(0))]

# создание списка улиц, которые участвуют в связях со зданиями.
# вывод суммарного числа работников в зданиях на каждой улице
def taskA2(one_to_many:list):
    res=[]
    filled_streets = set([s_name for s_name, num, people in one_to_many])
    # перебор этого списка
    for s in filled_streets:
        sum = 0
        # поиск в списке связей зданий, расположенных на текущей улице, и
подсчёт людей, в них работающих
        for s_name, num, people in one_to_many:
            if s == s_name:
                sum += people
        res.append((s, sum))
    return sorted(res, key=itemgetter(1), reverse=True)

# вывод проспектов и номеров зданий на них
def taskA3(many_to_many: list):
    #выбирается слово, искомое в названиях улиц.
    str_to_find='пр-кт' #по заданию - поиск проспектов
```

```
    #формируется список проспектов, участвующих в отношениях
    prospects=[s_name for s_name, num, people in many_to_many if
s_name.find(str_to_find)!=-1]
    #создаётся словарь, ключ - название проспекта, значение - список номеров
домов, расположенных на этих улицах
    return {pr:sorted([num for s_name, num, people in many_to_many if
pr==s_name]) for pr in prospects }
```

city_test.py

```
import unittest
from tasks import *
from generators import *

class MyTestCase(unittest.TestCase):
    def setUp(self) -> None:
        self.builds = builds_generator()
        self.streets = steets_generator()
        build_placing({
            0: [0,1],
            2: [2,4,3],
            4: [5,6,7],
            5: [10, 9, 8]
                            }, self.streets, self.builds)
        self.crosses = street_crossing([
    [0, 1], [0, 0], [2, 2], [2, 4], [2, 3], [4, 5], [4, 6], [4, 7], [5, 10],
[5, 9], [5, 8],
    [3, 0], [1, 10], [1, 8]
])

    def test_taskA1(self):
        compareWith=[('Ленинский пр-кт', 483, 13), ('Ленинский пр-кт', 484,
18), ('Ленинский пр-кт', 485, 11),
                     ('Молдавский пр-кт', 1, 14), ('Молдавский пр-кт', 14,
28), ('Петра Великого', 13, 103),
                     ('Петра Великого', 15, 238), ('Петра Великого', 14, 89),
('Серебряная', 15, 45),
                     ('Серебряная', 16, 44), ('Серебряная', 17, 56)]
        self.assertEqual(taskA1(one_to_many_generator(self.streets,
self.builds)), compareWith)

    def test_taskA2(self):
        compareWith=[('Петра Великого', 430), ('Серебряная', 145),
('Ленинский пр-кт', 42), ('Молдавский пр-кт', 42)]
        self.assertEqual(taskA2(one_to_many_generator(self.streets,
self.builds)), compareWith)

    def test_taskA3(self):
        compareWith={'Молдавский пр-кт': [1, 14], 'Житомирский пр-кт': [1],
'Ленинский пр-кт': [483, 484, 485]}
        self.assertEqual(taskA3(many_to_many_generator(self.streets,
self.builds, self.crosses)), compareWith)

if __name__ == '__main__':
    unittest.main()
```

streets.json

```
[
  "Молдавский пр-кт",
  "Пушкина",
  "Серебряная",
  "Житомирский пр-кт",
  "Ленинский пр-кт",
```

```
  "Петра Великого"
]
```

data.json


```
[
    {
    "num": 1,
    "people": 14,
    "year": 2001
    },
  {
    "num": 14,
    "people": 28,
    "year": 2003
  },
  {
    "num": 15,
    "people": 45,
    "year": 1988
  },
  {
    "num": 16,
    "people": 44,
    "year": 1986
  },
  {
    "num": 17,
    "people": 56,
    "year": 1989
  },
  {
    "num": 483,
    "people": 13,
    "year": 1956
  },
  {
    "num": 484,
    "people": 18,
    "year": 1955
  },
  {
    "num": 485,
    "people": 11,
    "year": 1872
  },
  {
    "num": 13,
    "people": 103,
    "year": 2007
  },
  {
    "num": 15,
    "people": 238,
    "year": 2014
  },
  {
    "num": 14,
    "people": 89,
    "year": 2005
  }
]
```

Результаты работы программы.

```
PS C:\Users\user\PycharmProjects\RK2> python -m unittest city_test
...
----------------------------------------------------------------------
Ran 3 tests in 0.002s

OK
```