ИУ5-32Б,

Фролов Илья

***Рубежный контроль 2***

***Вариант 22***

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.

from operator import itemgetter  
  
  
class ProgramLang:  
 def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
  
class Library:  
 def \_\_init\_\_(self, id, name, doc\_lib\_href, prog\_lang\_id):  
 self.id = id  
 self.name = name  
 self.doc\_lib\_href = doc\_lib\_href  
 self.prog\_lang\_id = prog\_lang\_id  
  
  
class ProgLangLib:  
 def \_\_init\_\_(self, prog\_lang\_id, lib\_id):  
 self.prog\_lang\_id = prog\_lang\_id  
 self.lib\_id = lib\_id  
  
  
def get\_one\_to\_many(prog\_langs, libs):  
 return [(lib.name, lib.doc\_lib\_href, pl.name)  
 for pl in prog\_langs  
 for lib in libs  
 if lib.prog\_lang\_id == pl.id]  
  
  
def get\_many\_to\_many(pl\_libs, libs, prog\_langs):  
 many\_to\_many\_temp = [(pl.name, ps.prog\_lang\_id, ps.lib\_id)  
 for pl in prog\_langs  
 for ps in pl\_libs  
 if ps.prog\_lang\_id == pl.id]  
  
 return [(lib.name, lib.doc\_lib\_href, pl\_name)  
 for pl\_name, pl\_id, lib\_id in many\_to\_many\_temp  
 for lib in libs if lib.id == lib\_id]  
  
  
def first\_task(lib\_list):  
 return sorted(lib\_list, key=itemgetter(0))  
  
  
def second\_task(lib\_list):  
 res\_2 = []  
 temp\_dict = dict()  
 for i in lib\_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(lib\_list, end\_ch):  
 return [(i[0], i[2]) for i in lib\_list if str(i[0]).endswith(end\_ch)]  
  
  
def main():  
 prog\_langs = [  
 ProgramLang(1, "C++"),  
 ProgramLang(2, "Java"),  
 ProgramLang(3, "Kotlin"),  
 ]  
  
 libs = [  
 Library(1, "JUnit4", "https://kotlinlang.org/docs/jvm-test-using-junit.html", 3),  
 Library(2, "JUnit4", "https://junit.org/junit4/", 2),  
 Library(3, "JUnit5", "https://junit.org/junit5/", 3),  
 Library(4, "Cucumber", "https://cucumber.io/docs/installation/java/", 3),  
 Library(5, "iostream", "https://en.cppreference.com/w/cpp/header/iostream", 1),  
 Library(6, "stdlib", "https://en.cppreference.com/w/cpp/header/cstdlib", 1)  
 ]  
  
 pl\_libs = [  
 ProgLangLib(1, 5),  
 ProgLangLib(1, 6),  
 ProgLangLib(3, 4),  
 ProgLangLib(3, 3),  
 ProgLangLib(2, 2),  
 ProgLangLib(3, 1),  
 ]  
  
 one\_to\_many = get\_one\_to\_many(prog\_langs, libs)  
 many\_to\_many = get\_many\_to\_many(pl\_libs, libs, prog\_langs)  
  
 print('Задание Б1')  
 for lib in first\_task(one\_to\_many):  
 print(lib)  
  
 print("\nЗадание Б2")  
 print(second\_task(one\_to\_many))  
  
 print("\nЗадание Б3")  
 print(third\_task(many\_to\_many, '4'))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

1. Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

UNIT\_TESTS.py

import main  
from operator import itemgetter  
import unittest  
  
  
class TestMainMethods(unittest.TestCase):  
  
 def setUp(self):  
 self.prog\_langs = [  
 main.ProgramLang(1, "C++"),  
 main.ProgramLang(2, "Java"),  
 main.ProgramLang(3, "Kotlin"),  
 ]  
  
 self.libs = [  
 main.Library(1, "JUnit4", "https://kotlinlang.org/docs/jvm-test-using-junit.html", 3),  
 main.Library(2, "JUnit4", "https://junit.org/junit4/", 2),  
 main.Library(3, "JUnit5", "https://junit.org/junit5/", 3),  
 main.Library(4, "Cucumber", "https://cucumber.io/docs/installation/java/", 3),  
 main.Library(5, "iostream", "https://en.cppreference.com/w/cpp/header/iostream", 1),  
 main.Library(6, "stdlib", "https://en.cppreference.com/w/cpp/header/cstdlib", 1)  
 ]  
  
 self.pl\_libs = [  
 main.ProgLangLib(1, 5),  
 main.ProgLangLib(1, 6),  
 main.ProgLangLib(3, 4),  
 main.ProgLangLib(3, 3),  
 main.ProgLangLib(2, 2),  
 main.ProgLangLib(3, 1),  
 ]  
  
 self.one\_to\_many = [  
 ('JUnit5', 'https://junit.org/junit5/', 'Kotlin'),  
 ('Cucumber', 'https://cucumber.io/docs/installation/java/', 'Kotlin'),  
 ('JUnit4', 'https://junit.org/junit4/', 'Java'),  
 ('JUnit4', 'https://kotlinlang.org/docs/jvm-test-using-junit.html', 'Kotlin'),  
 ('iostream', 'https://en.cppreference.com/w/cpp/header/iostream', 'C++'),  
 ('stdlib', 'https://en.cppreference.com/w/cpp/header/cstdlib', 'C++')  
 ]  
  
 def test\_first\_task\_method(self):  
 result = main.first\_task(self.one\_to\_many)  
 reference = sorted(self.one\_to\_many, key=itemgetter(0))  
 self.assertEqual(result, reference)  
  
 def test\_second\_task\_method(self):  
 result = main.second\_task(self.one\_to\_many)  
 reference = [('Kotlin', 3), ('C++', 2), ('Java', 1)]  
 self.assertEqual(result, reference)  
  
 def test\_third\_task\_method(self):  
 many\_to\_many = [  
 ('JUnit4', 'https://kotlinlang.org/docs/jvm-test-using-junit.html', 'Kotlin'),  
 ('JUnit4', 'https://junit.org/junit4/', 'Java'),  
 ('JUnit5', 'https://junit.org/junit5/', 'Kotlin'),  
 ('Cucumber', 'https://cucumber.io/docs/installation/java/', 'Kotlin'),  
 ('iostream', 'https://en.cppreference.com/w/cpp/header/iostream', 'C++'),  
 ('stdlib', 'https://en.cppreference.com/w/cpp/header/cstdlib', 'C++'),  
 ]  
  
 result = main.third\_task(many\_to\_many, '4')  
 reference = [('JUnit4', 'Java'), ('JUnit4', 'Kotlin')]  
 self.assertEqual(sorted(result), sorted(reference))

*# Сравниваем отсортированные списки*if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

Результаты тестирования:  
