Земцов Сергей ИУ5-34Б

**Отчет РК-2 по дисциплине**

**Парадигмы и конструкторы языков программирования**

Задание

1. Проведите рефакторинг текста программы таким образом, чтобы он был пригоден для тестирования

2. Для текста программы создайте модульное тестирование с применением TDD-фреймворка from operator import itemgetter

Текст программы main.py

class Department:

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

self.id = id

self.name = name

class Employee:

def \_\_init\_\_(self, id, name, department\_id):

self.id = id

self.name = name

self.department\_id = department\_id

def get\_departments\_starting\_with\_A(departments):

return [department for department in departments if department.name.startswith('А')]

def get\_employees\_by\_department(department, employees):

return [employee.name for employee in employees if employee.department\_id == department.id]

def main(departments, employees):

departments\_starting\_with\_A = get\_departments\_starting\_with\_A(departments)

result = {}

for department in departments\_starting\_with\_A:

department\_employees = get\_employees\_by\_department(department, employees)

result[department.name] = department\_employees

print('Отделы, начинающиеся с "А", и их сотрудники:')

for dept, emp\_list in result.items():

print(f'{dept}: {", ".join(emp\_list) if emp\_list else "нет сотрудников"}')

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

# Примеры данных

departments = [

Department(1, 'Отдел продаж'),

Department(2, 'Отдел разработки'),

Department(3, 'Аналитический отдел'),

Department(4, 'Отдел маркетинга'),

]

employees = [

Employee(1, 'Иван', 3),

Employee(2, 'Светлана', 1),

Employee(3, 'Александр', 3),

Employee(4, 'Мария', 2),

Employee(5, 'Анастасия', 3),

]

main(departments, employees)

Текст программы tests.py

import unittest

class TestDepartmentEmployeeFunctions(unittest.TestCase):

def setUp(self):

self.departments = [

Department(1, 'Отдел продаж'),

Department(2, 'Отдел разработки'),

Department(3, 'Аналитический отдел'),

Department(4, 'Отдел маркетинга'),

]

self.employees = [

Employee(1, 'Иван', 3),

Employee(2, 'Светлана', 1),

Employee(3, 'Александр', 3),

Employee(4, 'Мария', 2),

Employee(5, 'Анастасия', 3),

]

def test\_get\_departments\_starting\_with\_A(self):

result = get\_departments\_starting\_with\_A(self.departments)

self.assertEqual(len(result), 1)

self.assertEqual(result[0].name, 'Аналитический отдел')

def test\_get\_employees\_by\_department(self):

department = self.departments[2] # Аналитический отдел

result = get\_employees\_by\_department(department, self.employees)

self.assertEqual(len(result), 3)

self.assertIn('Иван', result)

self.assertIn('Александр', result)

self.assertIn('Анастасия', result)

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

unittest.main()