**Условия рубежного контроля №2 по курсу ПиК ЯП**

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\_refactor.py

class Department:

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

self.id = id

self.name = name

self.students = students

self.faculty\_id = faculty\_id

class Faculty:

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

self.id = id

self.name = name

def \_\_lt\_\_(self, obj):

return self.name < obj.name

class DepartmentFaculty:

def \_\_init\_\_(self, dep\_id, fac\_id):

self.dep\_id = dep\_id

self.fac\_id = fac\_id

# Данные

deps = [

Department(1, "FN4", 1894, 4),

Department(2, "IU7", 1947, 1),

Department(3, "IU5", 1938, 1),

Department(4, "Э6", 1954, 5),

Department(5, "MT3", 1949, 2),

Department(6, "MT4", 2045, 2),

Department(7, "FN2", 1843, 4),

Department(8, "RK1", 1932, 2),

]

facs = [

Faculty(id=1, name="Computer science"),

Faculty(id=2, name="Engineering"),

Faculty(id=3, name="Business IT"),

Faculty(id=4, name="Physics"),

Faculty(id=5, name="Energetic"),

]

deps\_to\_facs = [

DepartmentFaculty(dep\_id=1, fac\_id=4),

DepartmentFaculty(dep\_id=2, fac\_id=1),

DepartmentFaculty(dep\_id=3, fac\_id=1),

DepartmentFaculty(dep\_id=4, fac\_id=5),

DepartmentFaculty(dep\_id=5, fac\_id=2),

DepartmentFaculty(dep\_id=6, fac\_id=2),

DepartmentFaculty(dep\_id=7, fac\_id=4),

DepartmentFaculty(dep\_id=8, fac\_id=2),

]

# Функции для здания №1

def get\_faculty\_departments(faculties, departments, prefix):

result = {}

for fac in faculties:

if fac.name.startswith(prefix):

result[fac.name] = [dep.name for dep in departments if dep.faculty\_id == fac.id]

return result

# Функция для здания №2

def get\_max\_students\_per\_faculty(faculties, departments):

result = {}

for fac in faculties:

depts = [dept for dept in departments if dept.faculty\_id == fac.id]

if depts:

max\_dept = max(depts, key=lambda d: d.students)

result[fac.name] = (max\_dept.name, max\_dept.students)

return sorted(result.items(), key=lambda d: d[1][1], reverse=True)

# Функция для здания №3

def get\_faculty\_with\_departments(faculties, departments, deps\_to\_facs):

many\_to\_many\_temp = [

(f.name, d.fac\_id, d.dep\_id) for f in faculties for d in deps\_to\_facs if f.id == d.fac\_id

]

many\_to\_many = [

(d.students, d.name, name)

for name, fac\_id, dep\_id in many\_to\_many\_temp

for d in departments

if d.id == dep\_id

]

result = {}

for fac in sorted(faculties):

result[fac.name] = list(filter(lambda i: i[2] == fac.name, many\_to\_many))

return result

# Основная функция

def main():

print("Здание №1")

print(get\_faculty\_departments(facs, deps, "E"))

print("Здание №2")

print(get\_max\_students\_per\_faculty(facs, deps))

print("Здание №3")

print(get\_faculty\_with\_departments(facs, deps, deps\_to\_facs))

if \_\_name\_\_ == "\_\_main\_\_":

main()

Test\_module.py

import unittest

from main\_refactor import Department, Faculty, DepartmentFaculty # Импорт классов

from main\_refactor import get\_faculty\_departments, get\_max\_students\_per\_faculty, get\_faculty\_with\_departments # Импорт функций

class TestDepartmentFacultyFunctions(unittest.TestCase):

def setUp(self):

self.deps = [

Department(1, "FN4", 1894, 4),

Department(2, "IU7", 1947, 1),

Department(3, "IU5", 1938, 1),

]

self.facs = [

Faculty(id=1, name="Engineering"),

Faculty(id=2, name="Energetic"),

]

self.deps\_to\_facs = [

DepartmentFaculty(dep\_id=1, fac\_id=4),

DepartmentFaculty(dep\_id=2, fac\_id=1),

]

def test\_get\_faculty\_departments(self):

result = get\_faculty\_departments(self.facs, self.deps, "E")

self.assertEqual(result, {"Engineering": ["IU7"], "Energetic": []})

def test\_get\_max\_students\_per\_faculty(self):

result = get\_max\_students\_per\_faculty(self.facs, self.deps)

expected = [("Engineering", ("IU7", 1947))]

self.assertEqual(result, expected)

def test\_get\_faculty\_with\_departments(self):

result = get\_faculty\_with\_departments(self.facs, self.deps, self.deps\_to\_facs)

expected = {

"Engineering": [(1947, "IU7", "Engineering")],

"Energetic": [],

}

self.assertEqual(result, expected)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()