Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю № 2
Вариант В-27

Выполнил:
студент группы ИУ5-34Б
Бромберг Е.А.

Проверил:
преподаватель каф. ИУ5
Нардид А.Н.

Москва, 2024 г.

Текст программы

main.py

```
class Teacher:
    """Преподаватель"""

    def __init__(self, id, fio, salary, course_id):
        self.id = id
        self.fio = fio
        self.salary = salary
        self.course_id = course_id


class Course:
    """Учебный курс"""

    def __init__(self, id, name):
        self.id = id
        self.name = name


class TeacherCourse:
    """'Преподаватели учебного курса' для реализации связи многие-ко-многим"""

    def __init__(self, course_id, teacher_id):
        self.course_id = course_id
        self.teacher_id = teacher_id


# Соединение данных один-ко-многим
def make_one_to_many(courses, teachers):
    one_to_many = [(teacher.fio, teacher.salary, course.name)
                   for course in courses
                   for teacher in teachers
                   if teacher.course_id == course.id]
    return one_to_many


# Соединение данных многие-ко-многим
def make_many_to_many(courses, teachers, teachers_courses):
    many_to_many_temp = [(course.name, elem.course_id, elem.teacher_id)
                         for course in courses
                         for elem in teachers_courses
                         if course.id == elem.course_id]
    many_to_many = [(teacher.fio, teacher.salary, course_name)
                    for course_name, course_id, teacher_id in many_to_many_temp
                    for teacher in teachers if teacher.id == teacher_id]
    return many_to_many


def do_task_one(one_to_many):
    result_1_temp = list(filter(lambda x: x[0].startswith('A'), one_to_many))
    result_1 = [(fio, course) for fio, _, course in result_1_temp]
    return result_1


def do_task_two(one_to_many, courses):
    result_2_unsorted = []

    # Перебираем все учебные курсы
    for course in courses:
        # Список преподавателей учебного курса
        course_teachers = list(filter(lambda x: x[2] == course.name,
```

```
one_to_many))
        # Если учебный курс преподаёт хотя бы 1 преподаватель
        if len(course_teachers) > 0:
            # Зарплаты преподавателей учебного курса
            course_sals = [salary for _, salary, _ in course_teachers]
            # Минимальная зарплата преподавателей учебного курса
            course_sals_min = min(course_sals)
            result_2_unsorted.append((course.name, course_sals_min))

    # Сортировка по минимальной зарплате
    result_2 = sorted(result_2_unsorted, key=lambda x: x[1])
    return result_2


def do_task_three(many_to_many):
    result_3_temp = sorted(many_to_many, key=lambda x: x[0])
    result_3 = [(fio, course) for fio, _, course in result_3_temp]
    return result_3


def main():
    """Основная функция"""

    # Учебные курсы
    courses = [
        Course(1, 'математический анализ'),
        Course(2, 'физика'),
        Course(3, 'модели данных')
    ]
    # Преподаватели
    teachers = [
        Teacher(1, 'Акимов', 75_000, 1),
        Teacher(2, 'Бурова', 85_000, 2),
        Teacher(3, 'Александрова', 80_000, 2),
        Teacher(4, 'Корчагин', 70_000, 2),
        Teacher(5, 'Бычков', 90_000, 3),
    ]
    teachers_courses = [
        TeacherCourse(1, 1),
        TeacherCourse(2, 2),
        TeacherCourse(2, 3),
        TeacherCourse(2, 4),
        TeacherCourse(3, 5)
    ]
    one_to_many = make_one_to_many(courses, teachers)
    many_to_many = make_many_to_many(courses, teachers, teachers_courses)

    print('Задание B1')
    print(do_task_one(one_to_many))

    print('\nЗадание B2')
    print(do_task_two(one_to_many, courses))

    print('\nЗадание B3')
    print(do_task_three(many_to_many))


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

unit_test.py

```
import unittest
import main


class TestTeacherCourse(unittest.TestCase):
    # Учебные курсы
    courses = [
        main.Course(1, 'математический анализ'),
        main.Course(2, 'физика'),
        main.Course(3, 'модели данных')
    ]
    # Преподаватели
    teachers = [
        main.Teacher(1, 'Акимов', 75_000, 1),
        main.Teacher(2, 'Бурова', 85_000, 2),
        main.Teacher(3, 'Александрова', 80_000, 2),
        main.Teacher(4, 'Корчагин', 70_000, 2),
        main.Teacher(5, 'Бычков', 90_000, 3),
    ]
    teachers_courses = [
        main.TeacherCourse(1, 1),
        main.TeacherCourse(2, 2),
        main.TeacherCourse(2, 3),
        main.TeacherCourse(2, 4),
        main.TeacherCourse(3, 5)
    ]
    one_to_many = main.make_one_to_many(courses, teachers)
    many_to_many = main.make_many_to_many(courses, teachers, teachers_courses)

    def test_do_task_one(self):
        result_1 = main.do_task_one(self.one_to_many)
        result_2 = [('Акимов', 'математический анализ'), ('Александрова',
'физика')]
        self.assertEqual(result_1, result_2)

    def test_do_task_two(self):
        result_1 = main.do_task_two(self.one_to_many, self.courses)
        result_2 = [('физика', 70000), ('математический анализ', 75000),
('модели данных', 90000)]
        self.assertEqual(result_1, result_2)

    def test_do_task_three(self):
        result_1 = main.do_task_three(self.many_to_many)
        result_2 = [('Акимов', 'математический анализ'), ('Александрова',
'физика'), ('Бурова', 'физика'),
                    ('Бычков', 'модели данных'), ('Корчагин', 'физика')]
        self.assertEqual(result_1, result_2)


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

Результат выполнения программы