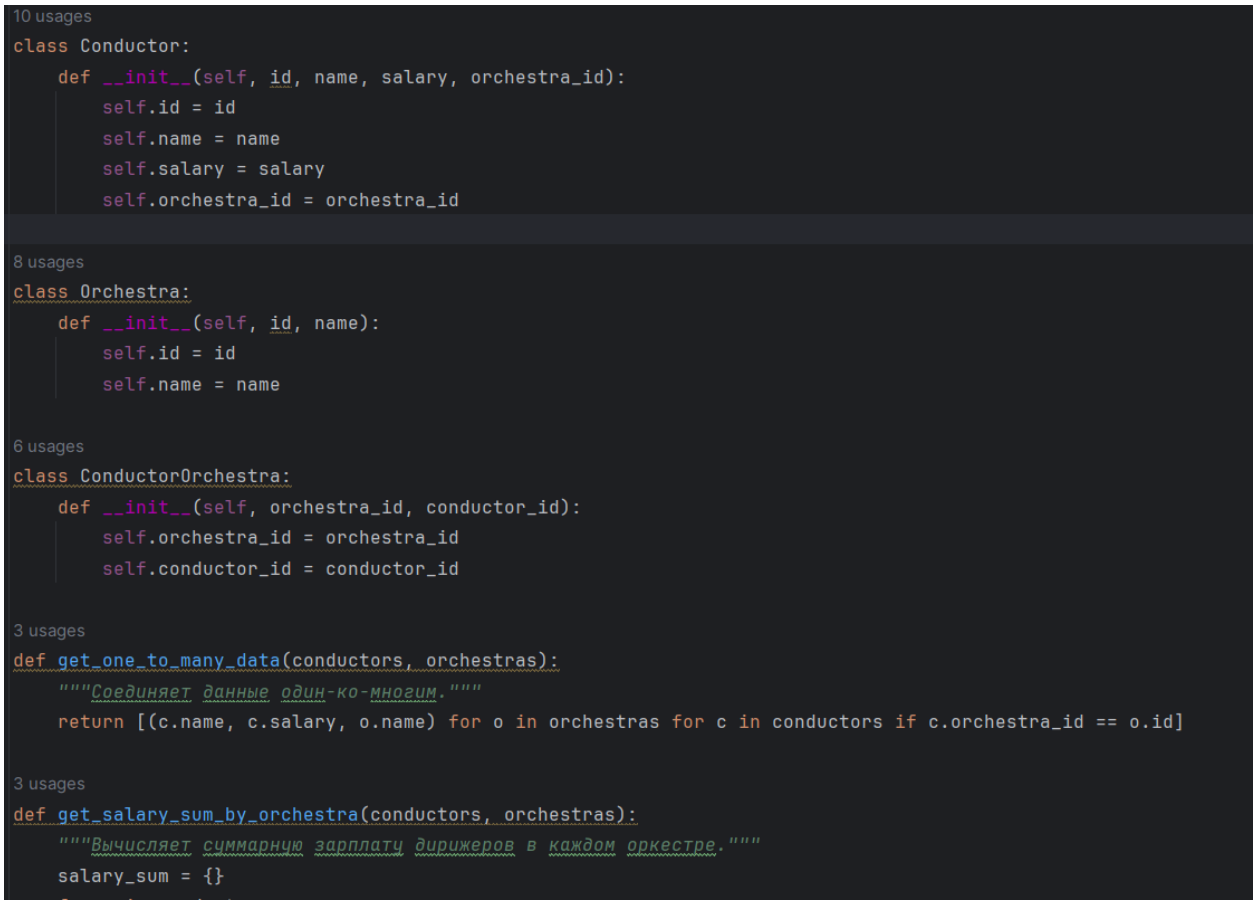
РК №2

Парадигмы и конструкции языков программирования

ИБМ3-34Б

Новачлы Надежда

1)Провести рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.



```
10 usages
class Conductor:
    def __init__(self, id, name, salary, orchestra_id):
        self.id = id
        self.name = name
        self.salary = salary
        self.orchestra_id = orchestra_id

8 usages
class Orchestra:
    def __init__(self, id, name):
        self.id = id
        self.name = name

6 usages
class ConductorOrchestra:
    def __init__(self, orchestra_id, conductor_id):
        self.orchestra_id = orchestra_id
        self.conductor_id = conductor_id

3 usages
def get_one_to_many_data(conductors, orchestras):
    """Соединяет данные один-ко-многим."""
    return [(c.name, c.salary, o.name) for o in orchestras for c in conductors if c.orchestra_id == o.id]

3 usages
def get_salary_sum_by_orchestra(conductors, orchestras):
    """Вычисляет суммарную зарплату дирижеров в каждом оркестре."""
    salary_sum = {}
```

```
    for c in conductors:
        salary_sum[c.orchestra_id] = salary_sum.get(c.orchestra_id, 0) + c.salary
    return [(o.name, salary_sum.get(o.id, 0)) for o in orchestras]


3 usages
def get_many_to_many_data(conductors, orchestras, conductors_orchestras):
    """Соединяет данные многие-ко-многим."""
    many_to_many_temp = [(o.name, co.orchestra_id, co.conductor_id)
                         for o in orchestras
                         for co in conductors_orchestras
                         if o.id == co.orchestra_id]
    many_to_many = {}
    for orchestra_name, orchestra_id, conductor_id in many_to_many_temp:
        conductor_name = next((c.name for c in conductors if c.id == conductor_id), None)  #Обработка отсутствующего дирижера
        if orchestra_name not in many_to_many:
            many_to_many[orchestra_name] = []
        many_to_many[orchestra_name].append(conductor_name)
    return many_to_many

1 usage
def main():
    orchestras = [
        Orchestra( id: 1,  name: 'Государственный симфонический оркестр'),
        Orchestra( id: 2,  name: 'Молодежный оркестр'),
        Orchestra( id: 3,  name: 'Филармонический оркестр'),
    ]
```

```
    conductors = [
        Conductor( id: 1,  name: 'Желяпова',  salary: 50000,  orchestra_id: 1),
        Conductor( id: 2,  name: 'Калинков',  salary: 60000,  orchestra_id: 2),
        Conductor( id: 3,  name: 'Иванов',  salary: 55000,  orchestra_id: 1),
    ]
    conductors_orchestras = [
        ConductorOrchestra( orchestra_id: 1,  conductor_id: 1),
        ConductorOrchestra( orchestra_id: 2,  conductor_id: 2),
        ConductorOrchestra( orchestra_id: 1,  conductor_id: 3),
    ]

    print("Задание A1")
    print(sorted(get_one_to_many_data(conductors, orchestras)))
    print("Задание A2")
    print(get_salary_sum_by_orchestra(conductors, orchestras))
    print("Задание A3")
    print(get_many_to_many_data(conductors, orchestras, conductors_orchestras))

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

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

```python
import unittest
from PK2 import get_one_to_many_data, get_salary_sum_by_orchestra, get_many_to_many_data, Conductor, Orchestra, ConductorOrchestra

class TestMusicData(unittest.TestCase):

    def test_one_to_many(self):
        orchestras = [Orchestra( id: 1,  name: 'Orch1')]
        conductors = [Conductor( id: 1,  name: 'Cond1',  salary: 1000,  orchestra_id: 1)]
        expected = [('Cond1', 1000, 'Orch1')]
        self.assertEqual(get_one_to_many_data(conductors, orchestras), expected)

    def test_salary_sum(self):
        orchestras = [Orchestra( id: 1,  name: 'Orch1'), Orchestra( id: 2,  name: 'Orch2')]
        conductors = [Conductor( id: 1,  name: 'Cond1',  salary: 1000,  orchestra_id: 1), Conductor( id: 2,  name: 'Cond2',  salary: 2000,  orchestra_id: 1), Conductor( id: 3,
        expected = [('Orch1', 3000), ('Orch2', 3000)]
        self.assertEqual(get_salary_sum_by_orchestra(conductors, orchestras), expected)

    def test_many_to_many(self):
        orchestras = [Orchestra( id: 1,  name: 'Orch1')]
        conductors = [Conductor( id: 1,  name: 'Cond1',  salary: 1000,  orchestra_id: 1), Conductor( id: 2,  name: 'Cond2',  salary: 2000,  orchestra_id: 1)]
        conductors_orchestras = [ConductorOrchestra( orchestra_id: 1,  conductor_id: 1), ConductorOrchestra( orchestra_id: 1,  conductor_id: 2)]
        expected = {'Orch1': ['Cond1', 'Cond2']}
        self.assertEqual(get_many_to_many_data(conductors, orchestras, conductors_orchestras), expected)


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

Вывод:

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
(venv) PS D:\PYTHON (snape)\pythonProject>
```