ИУ5-33Б

Костяев Вячеслав

Отчет по РК2 по дисциплине
“Парадигмы и конструкции языков программирования”

```
# orchestra_module.py


from dataclasses import dataclass
from typing import List, Optional, Tuple



@dataclass
class Musician:
    id: int
    last_name: str
    experience: int
    orchestra_id: int

    def __repr__(self):
        return (f"Музыкант(id={self.id}, фамилия='{self.last_name}', "
                f"опыт={self.experience}, orchestra_id={self.orchestra_id})")



@dataclass
```

```
class Orchestra:
    id: int
    name: str

    def __repr__(self):
        return f"Оркестр(id={self.id}, название='{self.name}')"


@dataclass
class MusicianOrchestra:
    musician_id: int
    orchestra_id: int

    def __repr__(self):
        return (f"Связь(музыкант_id={self.musician_id}, "
                f"оркестр_id={self.orchestra_id})")


class OrchestraManager:
    def __init__(self,
                 orchestras: List[Orchestra],
                 musicians: List[Musician],
                 musician_orchestra_links: List[MusicianOrchestra]):
```

```
        self.orchestras = orchestras
        self.musicians = musicians
        self.musician_orchestra_links = musician_orchestra_links

    def get_musicians_with_last_name_starting(self, initial: str) -> List[Tuple[Musician, Optional[Orchestra]]]:
        filtered_musicians = [m for m in self.musicians if m.last_name.startswith(initial)]
        result = []
        for musician in filtered_musicians:
            orchestra = self.get_orchestra_by_id(musician.orchestra_id)
            result.append((musician, orchestra))
        return result

    def get_orchestra_min_experience_sorted(self) -> List[Tuple[str, int]]:
        orchestra_experience = {}
        for musician in self.musicians:
            orchestra_id = musician.orchestra_id
            orchestra_experience.setdefault(orchestra_id, []).append(musician.experience)

        orchestra_min_exp = [
            (self.get_orchestra_by_id(orchestra_id).name, min(experiences))
            for orchestra_id, experiences in orchestra_experience.items()
```

```
        ]

        orchestra_min_exp_sorted = sorted(orchestra_min_exp, key=lambda x: x[1])
        return orchestra_min_exp_sorted

    def get_sorted_musician_orchestra_relationships(self) -> List[Tuple[str, str]]:
        relationships = [
            (self.get_musician_by_id(link.musician_id).last_name,
             self.get_orchestra_by_id(link.orchestra_id).name)
            for link in self.musician_orchestra_links
        ]

        relationships_sorted = sorted(relationships, key=lambda x: x[0])
        return relationships_sorted

    def get_orchestra_by_id(self, orchestra_id: int) -> Optional[Orchestra]:
        return next((o for o in self.orchestras if o.id == orchestra_id), None)

    def get_musician_by_id(self, musician_id: int) -> Optional[Musician]:
        return next((m for m in self.musicians if m.id == musician_id), None)
```

```
def main():
    orchestras = [
        Orchestra(1, 'Симфонический Оркестр'),
        Orchestra(2, 'Камерный Оркестр'),
        Orchestra(3, 'Джазовый Оркестр')
    ]

    musicians = [
        Musician(1, 'Александров', 5, 1),
        Musician(2, 'Иванов', 3, 2),
        Musician(3, 'Антонов', 7, 1),
        Musician(4, 'Борисов', 2, 3),
        Musician(5, 'Андреев', 4, 3)
    ]

    musician_orchestra_links = [
        MusicianOrchestra(1, 1),
        MusicianOrchestra(2, 2),
        MusicianOrchestra(3, 1),
        MusicianOrchestra(3, 2),
        MusicianOrchestra(4, 3),
        MusicianOrchestra(5, 3),
```

```
        MusicianOrchestra(5, 1)
    ]

    manager = OrchestraManager(orchestras, musicians,
musician_orchestra_links)

    print("Запрос 1: Музыканты, у которых фамилия начинается с 'A', и
их оркестры:\n")
    musicians_with_a = manager.get_musicians_with_last_name_starting('A')
    for musician, orchestra in musicians_with_a:
        if orchestra:
            print(f"Музыкант: {musician.last_name}, Оркестр:
{orchestra.name}")

    print("\nЗапрос 2: Оркестры с минимальным опытом музыкантов,
отсортированные по минимальному опыту:\n")
    orchestra_min_exp_sorted =
manager.get_orchestra_min_experience_sorted()
    for name, min_exp in orchestra_min_exp_sorted:
        print(f"Оркестр: {name}, Минимальный опыт музыкантов:
{min_exp} лет")

    print("\nЗапрос 3: Все связанные музыканты и оркестры,
отсортированные по фамилии музыкантов:\n")
```

```
    relationships_sorted =
manager.get_sorted_musician_orchestra_relationships()
    for last_name, orchestra_name in relationships_sorted:
        print(f"Музыкант: {last_name}, Оркестр: {orchestra_name}")



if __name__ == '__main__':
    main()



# test_orchestra.py

import unittest
from orchestra_module import (
    Orchestra,
    Musician,
    MusicianOrchestra,
    OrchestraManager
)



class TestOrchestraManager(unittest.TestCase):
    def setUp(self):
```

```
self.orchestras = [
    Orchestra(1, 'Симфонический Оркестр'),
    Orchestra(2, 'Камерный Оркестр'),
    Orchestra(3, 'Джазовый Оркестр')
]

self.musicians = [
    Musician(1, 'Александров', 5, 1),
    Musician(2, 'Иванов', 3, 2),
    Musician(3, 'Антонов', 7, 1),
    Musician(4, 'Борисов', 2, 3),
    Musician(5, 'Андреев', 4, 3)
]

self.musician_orchestra_links = [
    MusicianOrchestra(1, 1),
    MusicianOrchestra(2, 2),
    MusicianOrchestra(3, 1),
    MusicianOrchestra(3, 2),
    MusicianOrchestra(4, 3),
    MusicianOrchestra(5, 3),
    MusicianOrchestra(5, 1)
]
```

```
        self.manager = OrchestraManager(
            self.orchestras,
            self.musicians,
            self.musician_orchestra_links
        )

    def test_get_musicians_with_last_name_starting_A(self):
        result = self.manager.get_musicians_with_last_name_starting('A')
        expected = [
            (self.musicians[0], self.orchestras[0]),
            (self.musicians[2], self.orchestras[0]),
            (self.musicians[4], self.orchestras[2])
        ]
        self.assertEqual(result, expected)

    def test_get_orchestra_min_experience_sorted(self):
        result = self.manager.get_orchestra_min_experience_sorted()
        expected = [
            ('Джазовый Оркестр', 2),
            ('Камерный Оркестр', 3),
            ('Симфонический Оркестр', 5)
        ]
```

```
        self.assertEqual(result, expected)

    def test_get_sorted_musician_orchestra_relationships(self):
        result = self.manager.get_sorted_musician_orchestra_relationships()
        expected = [
            ('Александров', 'Симфонический Оркестр'),
            ('Антонов', 'Симфонический Оркестр'),
            ('Антонов', 'Камерный Оркестр'),
            ('Андреев', 'Джазовый Оркестр'),
            ('Андреев', 'Симфонический Оркестр'),
            ('Борисов', 'Джазовый Оркестр'),
            ('Иванов', 'Камерный Оркестр')
        ]
        self.assertEqual(result, expected)


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

Результаты выполнения

```
Запрос 1: Музыканты, у которых фамилия начинается с 'А', и их оркестры:

Музыкант: Александров, Оркестр: Симфонический Оркестр
Музыкант: Антонов, Оркестр: Симфонический Оркестр
Музыкант: Андреев, Оркестр: Джазовый Оркестр

Запрос 2: Оркестры с минимальным опытом музыкантов, отсортированные по минимальному опыту:

Оркестр: Джазовый Оркестр, Минимальный опыт музыкантов: 2 лет
Оркестр: Камерный Оркестр, Минимальный опыт музыкантов: 3 лет
Оркестр: Симфонический Оркестр, Минимальный опыт музыкантов: 5 лет

Запрос 3: Все связанные музыканты и оркестры, отсортированные по фамилии музыкантов:

Музыкант: Александров, Оркестр: Симфонический Оркестр
Музыкант: Андреев, Оркестр: Джазовый Оркестр
Музыкант: Андреев, Оркестр: Симфонический Оркестр
Музыкант: Антонов, Оркестр: Симфонический Оркестр
Музыкант: Антонов, Оркестр: Камерный Оркестр
Музыкант: Борисов, Оркестр: Джазовый Оркестр
Музыкант: Иванов, Оркестр: Камерный Оркестр
```

```
PS C:\Users\s\Desktop\labsf\rk2>  c:; cd 'c:\Users\s\Desktop\labsf\rk2'
ra.py'
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```