МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “Программирование и конструкции языков программирования”

Отчет по рубежному контролю №2

Вариант: А-14

Выполнил:
Рахманов Виталий
ИУ5-31Б

Преподаватель:
Гапанюк Ю.Е.

Москва 2025

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).

Вариант 14: CD-диск; Библиотека CD-дисков

Текст программы

1) rk2.py

```
from dataclasses import dataclass
from typing import List, Dict, Tuple

@dataclass
class CDLibraryDisk:
    id: int
    library_id: int
    disk_id: int

@dataclass
class CDLibrary:
    id: int
    name: str
    location: str

@dataclass
class CDDisk:
    id: int
    title: str
    artist: str
    genre: str
    price: float
    year: int

class CDLibrarySystem:
    def __init__(self):
        self.libraries: List[CDLibrary] = []
        self.disks: List[CDDisk] = []
        self.library_disks: List[CDLibraryDisk] = []

    def add_library(self, library: CDLibrary) -> None:
        self.libraries.append(library)

    def add_disk(self, disk: CDDisk) -> None:
        self.disks.append(disk)
```

```
def add_library_disk_relation(self, relation: CDLibraryDisk) -> None:
    self.library_disks.append(relation)

def create_one_to_many_relations(self) -> List[Tuple[CDLibrary, CDDisk]]:
    relations = []

    for disk in self.disks:
        for relation in self.library_disks:
            if relation.disk_id == disk.id:
                library = next((lib for lib in self.libraries
                                if lib.id == relation.library_id), None)
                if library:
                    relations.append((library, disk))

    return relations

def get_sorted_library_disks(self) -> List[Tuple[CDLibrary, CDDisk]]: # Запрос 1
    relations = self.create_one_to_many_relations()
    return sorted(relations, key=lambda x: x[0].name)

def get_library_costs(self) -> List[Dict]: # Запрос 2
    relations = self.create_one_to_many_relations()
    library_costs = {}

    for library, disk in relations:
        if library.id not in library_costs:
            library_costs[library.id] = {
                'library': library,
                'total_cost': 0.0
            }
        library_costs[library.id]['total_cost'] += disk.price

    return sorted(
        library_costs.values(),
        key=lambda x: x['total_cost']
    )

def get_department_libraries_with_disks(self) -> List[Dict]: # Запрос 3
    department_libraries = [
        lib for lib in self.libraries
        if "отдел" in lib.name.lower()
    ]
    result = []

    for library in department_libraries:
        # Находим связи для текущей библиотеки
        library_links = [
            link for link in self.library_disks
            if link.library_id == library.id
        ]

        # Находим CD-диски по связям
        library_disks = []
        for link in library_links:
            for disk in self.disks:
                if disk.id == link.disk_id:
                    library_disks.append(disk)
```

```
            result.append({
                'library': library,
                'disks': library_disks
            })

        return result

    def get_statistics(self) -> Dict[str, int]:
        return {
            'total_libraries': len(self.libraries),
            'total_disks': len(self.disks),
            'one_to_many_relations': len(self.create_one_to_many_relations()),
            'many_to_many_relations': len(self.library_disks)
        }

def create_test_data() -> CDLibrarySystem:
    system = CDLibrarySystem()

    system.add_library(CDLibrary(1, "Главный отдел музыки", "Центральный район"))
    system.add_library(CDLibrary(2, "Отдел классики", "Северный район"))
    system.add_library(CDLibrary(3, "Рок-коллекция", "Западный район"))
    system.add_library(CDLibrary(4, "Джазовый отдел", "Восточный район"))
    system.add_library(CDLibrary(5, "Поп-музыка отдел", "Южный район"))

    system.add_disk(CDDisk(1, "The Dark Side of the Moon", "Pink Floyd", "Rock", 25, 1973))
    system.add_disk(CDDisk(2, "Thriller", "Michael Jackson", "Pop", 19, 1982))
    system.add_disk(CDDisk(3, "Kind of Blue", "Miles Davis", "Jazz", 22, 1959))
    system.add_disk(CDDisk(4, "Symphony No. 9", "Beethoven", "Classical", 18, 1987))
    system.add_disk(CDDisk(5, "Back in Black", "AC/DC", "Rock", 21, 1980))
    system.add_disk(CDDisk(6, "The Wall", "Pink Floyd", "Rock", 24, 1979))

    # Связи многие-ко-многим
    system.add_library_disk_relation(CDLibraryDisk(1, 1, 1))
    system.add_library_disk_relation(CDLibraryDisk(2, 1, 2))
    system.add_library_disk_relation(CDLibraryDisk(3, 2, 4))
    system.add_library_disk_relation(CDLibraryDisk(4, 3, 1))
    system.add_library_disk_relation(CDLibraryDisk(5, 3, 5))
    system.add_library_disk_relation(CDLibraryDisk(6, 3, 6))
    system.add_library_disk_relation(CDLibraryDisk(7, 4, 3))
    system.add_library_disk_relation(CDLibraryDisk(8, 5, 2))
    system.add_library_disk_relation(CDLibraryDisk(9, 1, 6))

    return system


def main():
    system = create_test_data()

    print("\n3АПРОС 1")
    print("Список всех связанных библиотек и CD-дисков (один-ко-многим), отсортированный по
названиям библиотек:")

    sorted_library_disks = system.get_sorted_library_disks()
    for library, disk in sorted_library_disks:
        print(f"{library.name}: {disk.artist} - {disk.title}")
```

```
    print("\n3АПРОС 2")
    print("Список библиотек с суммарной стоимостью CD-дисков в каждой библиотеке:")

    library_costs = system.get_library_costs()
    for item in library_costs:
        print(f"{item['library'].name}: ${item['total_cost']:.2f}")

    print("\n3АПРОС 3")
    print("Список библиотек с 'отдел' в названии и имеющиеся в них CD-диски:")

    department_data = system.get_department_libraries_with_disks()
    for item in department_data:
        print(f"\n{item['library'].name}:")
        if item['disks']:
            for disk in item['disks']:
                print(f"  - {disk.artist} - {disk.title} ({disk.genre})")
        else:
            print("  - Нет CD-дисков")

    print("\nСТАТИСТИКА СИСТЕМЫ:")
    stats = system.get_statistics()
    for key, value in stats.items():
        print(f"{key.replace('_', ' ').title()}: {value}")

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

2) TDD.py

```
import unittest
from rk2 import CDLibrary, CDDisk, CDLibraryDisk, CDLibrarySystem


class TestCDLibrarySystem(unittest.TestCase):

    def setUp(self):
        self.system = CDLibrarySystem()

        self.library1 = CDLibrary(1, "Главный отдел музыки", "Центральный район")
        self.library2 = CDLibrary(2, "Отдел классики", "Северный район")
        self.library3 = CDLibrary(3, "Рок-коллекция", "Западный район")

        self.disk1 = CDDisk(1, "The Dark Side of the Moon", "Pink Floyd", "Rock", 25.0, 1973)
        self.disk2 = CDDisk(2, "Thriller", "Michael Jackson", "Pop", 19.0, 1982)
        self.disk3 = CDDisk(3, "Kind of Blue", "Miles Davis", "Jazz", 22.0, 1959)

        self.relation1 = CDLibraryDisk(1, 1, 1)
        self.relation2 = CDLibraryDisk(2, 1, 2)
        self.relation3 = CDLibraryDisk(3, 2, 3)

    def test_1_add_and_retrieve_libraries_disks(self): # тест1: Добавление и извлечение библиотек и
дисков
        self.system.add_library(self.library1)
        self.system.add_library(self.library2)

        self.system.add_disk(self.disk1)
        self.system.add_disk(self.disk2)
```

```
        self.assertEqual(len(self.system.libraries), 2)
        self.assertEqual(len(self.system.disks), 2)

        self.assertEqual(self.system.libraries[0].name, "Главный отдел музыки")
        self.assertEqual(self.system.disks[0].title, "The Dark Side of the Moon")
        self.assertEqual(self.system.disks[1].artist, "Michael Jackson")

    def test_2_library_disk_relations_and_sorting(self): # тест2: Связи библиотек с дисками и сортировка
        self.system.add_library(self.library1)
        self.system.add_library(self.library2)
        self.system.add_library(self.library3)

        self.system.add_disk(self.disk1)
        self.system.add_disk(self.disk2)
        self.system.add_disk(self.disk3)

        self.system.add_library_disk_relation(self.relation1)
        self.system.add_library_disk_relation(self.relation2)
        self.system.add_library_disk_relation(self.relation3)

        relations = self.system.create_one_to_many_relations()
        self.assertEqual(len(relations), 3)

        self.assertEqual(relations[0][0].id, 1)
        self.assertEqual(relations[0][1].id, 1)

        sorted_relations = self.system.get_sorted_library_disks()

        # Проверяем, что библиотеки отсортированы
        library_names = [lib.name for lib, _ in sorted_relations]
        self.assertEqual(library_names[0], "Главный отдел музыки")
        self.assertEqual(library_names[1], "Главный отдел музыки")
        self.assertEqual(library_names[2], "Отдел классики")
        self.assertEqual(len(sorted_relations), 3)

    def test_3_library_costs_and_department_filter(self): # тест3: Расчет стоимости и фильтрация по
отделу
        self.system.add_library(self.library1)
        self.system.add_library(self.library2)
        self.system.add_library(self.library3)

        self.system.add_disk(self.disk1)
        self.system.add_disk(self.disk2)
        self.system.add_disk(self.disk3)

        self.system.add_library_disk_relation(self.relation1)
        self.system.add_library_disk_relation(self.relation2)
        self.system.add_library_disk_relation(self.relation3)

        library_costs = self.system.get_library_costs()

        self.assertEqual(len(library_costs), 2)

        library_costs_sorted = sorted(library_costs, key=lambda x: x['library'].id)

        self.assertEqual(library_costs_sorted[0]['total_cost'], 44.0)  # 25.0 + 19.0
```

```
        self.assertEqual(library_costs_sorted[1]['total_cost'], 22.0)

        department_data = self.system.get_department_libraries_with_disks()

        self.assertEqual(len(department_data), 2)

        library_names = [item['library'].name for item in department_data]
        self.assertIn("Главный отдел музыки", library_names)
        self.assertIn("Отдел классики", library_names)
        self.assertNotIn("Рок-коллекция", library_names)

        department_data_sorted = sorted(department_data, key=lambda x: x['library'].id)

        self.assertEqual(len(department_data_sorted[0]['disks']), 2)
        self.assertEqual(len(department_data_sorted[1]['disks']), 1)


class CustomTestRunner:

    @staticmethod
    def run_tests():
        print("Запуск тестов...")

        test_instance = TestCDLibrarySystem()

        test_methods = [
            ('Тест1', test_instance.test_1_add_and_retrieve_libraries_disks),
            ('Тест2', test_instance.test_2_library_disk_relations_and_sorting),
            ('Тест3', test_instance.test_3_library_costs_and_department_filter)
        ]

        for test_name, test_method in test_methods:
            try:
                test_instance.setUp()
                test_method()
                print(f"{test_name} выполнен")
            except AssertionError as e:
                print(f"{test_name} не выполнен: {e}")
            except Exception as e:
                print(f"{test_name} ошибка: {e}")
            finally:
                test_instance.tearDown() if hasattr(test_instance, 'tearDown') else None


if __name__ == '__main__':
    CustomTestRunner.run_tests()
```

Результаты выполнения программы

1)

```
ЗАПРОС 1
Список всех связанных библиотек и CD-дисков (один-ко-многим), отсортированный по
названиям библиотек:
Главный отдел музыки: Pink Floyd - The Dark Side of the Moon
Главный отдел музыки: Michael Jackson - Thriller
Главный отдел музыки: Pink Floyd - The Wall
Джазовый отдел: Miles Davis - Kind of Blue
Отдел классики: Beethoven - Symphony No. 9
Поп-музыка отдел: Michael Jackson - Thriller
Рок-коллекция: Pink Floyd - The Dark Side of the Moon
Рок-коллекция: AC/DC - Back in Black
Рок-коллекция: Pink Floyd - The Wall

ЗАПРОС 2
Список библиотек с суммарной стоимостью CD-дисков в каждой библиотеке:
Отдел классики: $18.00
Поп-музыка отдел: $19.00
Джазовый отдел: $22.00
Главный отдел музыки: $68.00
Рок-коллекция: $70.00

ЗАПРОС 3
Список библиотек с 'отдел' в названии и имеющиеся в них CD-диски:

Главный отдел музыки:
  - Pink Floyd - The Dark Side of the Moon (Rock)
  - Michael Jackson - Thriller (Pop)
  - Pink Floyd - The Wall (Rock)

Отдел классики:
  - Beethoven - Symphony No. 9 (Classical)

Джазовый отдел:
  - Miles Davis - Kind of Blue (Jazz)

Поп-музыка отдел:
  - Michael Jackson - Thriller (Pop)

СТАТИСТИКА СИСТЕМЫ:
Total Libraries: 5
Total Disks: 6
One To Many Relations: 9
Many To Many Relations: 9
```

2)

```
Запуск тестов...
Тест1 выполнен
Тест2 выполнен
Тест3 выполнен
```