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-34

Выполнил:
студент группы ИУ5-34Б
Евсеев Дмитрий Михайлович

Проверил:
преподаватель каф. ИУ5
Нардид Анатолий Николаевич

Москва, 2024 г.

Постановка задачи

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).

Рефакторинг кода заключался в перенос логики задания в функции для последующего тестирования.

Текст программы

main.py

```
from operator import itemgetter


class Procedure:
    def __init__(self, id, name, priority, db_id):
        self.id = id
        self.name = name
        self.priority = priority
        self.db_id = db_id


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


class ProcedureInDataBase:
    def __init__(self, db_id, proc_id):
        self.db_id = db_id
        self.proc_id = proc_id


def create_one_to_many(procedures, databases):
    return [
        (proc.name, proc.priority, db.name)
        for db in databases
        for proc in procedures
        if proc.db_id == db.id
```

```
    ]


def create_many_to_many(procedures, databases,
procedures_in_dbs):
    many_to_many_temp = [
        (db.name, proc_in_db.db_id, proc_in_db.proc_id)
        for db in databases
        for proc_in_db in procedures_in_dbs
        if db.id == proc_in_db.db_id
    ]
    return [
        (proc.name, proc.priority, db_name)
        for db_name, db_id, proc_id in many_to_many_temp
        for proc in procedures if proc.id == proc_id
    ]


def task1(one_to_many):
    return [(x[0], x[2]) for x in one_to_many if
x[0].startswith('g')]


def task2(one_to_many):
    dbs = {}
    for i in one_to_many:
        if i[-1] in dbs:
            dbs[i[-1]].append(i[:2])
        else:
            dbs[i[-1]] = [i[:2]]
    ans = [[db, min(dbs[db], key=lambda x: x[1])[1]] for db in
dbs]
    ans.sort(key=lambda x: x[1])
    return ans


def task3(many_to_many):
    res = []
    for proc_name, proc_priority, db_name in
sorted(many_to_many, key=itemgetter(0)):
```

```
        res.append(f'Процедура {proc_name} с приоритетом
{proc_priority} находится в базе данных {db_name}')
    return res


def main():
    databases = [
        DataBase(1, 'Main db'),
        DataBase(2, 'Users db'),
        DataBase(3, 'Analytics db'),
        DataBase(4, 'Debug db'),
        DataBase(5, 'Products db')
    ]

    procedures = [
        Procedure(1, 'login', 1, 1),
        Procedure(2, 'get user name', 2, 2),
        Procedure(3, 'get user email', 4, 2),
        Procedure(4, 'check user status', 5, 2),
        Procedure(5, 'build diagram', 10, 3),
        Procedure(6, 'add tests', 14, 4),
        Procedure(7, 'get errors list', 22, 4),
        Procedure(8, 'add new product', 14, 5)
    ]

    procedures_in_dbs = [
        ProcedureInDataBase(1, 1),
        ProcedureInDataBase(2, 1),
        ProcedureInDataBase(2, 2),
        ProcedureInDataBase(2, 3),
        ProcedureInDataBase(2, 4),
        ProcedureInDataBase(3, 5),
        ProcedureInDataBase(4, 6),
        ProcedureInDataBase(4, 7),
        ProcedureInDataBase(5, 8)
    ]
    one_to_many = create_one_to_many(procedures, databases)
    many_to_many = create_many_to_many(procedures, databases,
procedures_in_dbs)
```

```
    print('Задание 1:\nСписок процедур, у которых название начинается на букву "g" и '
          'названия баз данных, в которых они хранятся')
    print(task1(one_to_many))
    print('\nЗадание 2:\nСписок баз данных с минимальным приоритетом процедуры')
    print(task2(one_to_many))
    print('\nЗадание 3:\nСписок всех процедур во всех базах данных (отсортировано по названию процедур)')
    print(task3(many_to_many))


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

test_main.py

```
import unittest
from main import *


class TestTasks(unittest.TestCase):
    def setUp(self):
        self.databases = [
            DataBase(1, 'Main db'),
            DataBase(2, 'Users db'),
            DataBase(3, 'Analytics db'),
            DataBase(4, 'Debug db'),
            DataBase(5, 'Products db')
        ]

        self.procedures = [
            Procedure(1, 'login', 1, 1),
            Procedure(2, 'get user name', 2, 2),
            Procedure(3, 'get user email', 4, 2),
            Procedure(4, 'check user status', 5, 2),
            Procedure(5, 'build diagram', 10, 3),
            Procedure(6, 'add tests', 14, 4),
            Procedure(7, 'get errors list', 22, 4),
            Procedure(8, 'add new product', 14, 5)
        ]
```

```
        self.procedures_in_dbs = [
            ProcedureInDataBase(1, 1),
            ProcedureInDataBase(2, 1),
            ProcedureInDataBase(2, 2),
            ProcedureInDataBase(2, 3),
            ProcedureInDataBase(2, 4),
            ProcedureInDataBase(3, 5),
            ProcedureInDataBase(4, 6),
            ProcedureInDataBase(4, 7),
            ProcedureInDataBase(5, 8)
        ]

        self.one_to_many = create_one_to_many(self.procedures,
self.databases)
        self.many_to_many = create_many_to_many(self.procedures,
self.databases, self.procedures_in_dbs)

    def test_task1(self):
        result = task1(self.one_to_many)
        true_result = [('get user name', 'Users db'), ('get user
email', 'Users db'), ('get errors list', 'Debug db')]
        self.assertEqual(result, true_result)

    def test_task2(self):
        result = task2(self.one_to_many)
        true_result = [['Main db', 1], ['Users db', 2],
['Analytics db', 10], ['Debug db', 14], ['Products db', 14]]
        self.assertEqual(result, true_result)

    def test_task3(self):
        result = task3(self.many_to_many)
        true_result = [
            'Процедура add new product с приоритетом 14
находится в базе данных Products db',
            'Процедура add tests с приоритетом 14 находится в
базе данных Debug db',
            'Процедура build diagram с приоритетом 10 находится
в базе данных Analytics db',
            'Процедура check user status с приоритетом 5
находится в базе данных Users db',
```

```
            'Процедура get errors list с приоритетом 22
находится в базе данных Debug db',
            'Процедура get user email с приоритетом 4 находится
в базе данных Users db',
            'Процедура get user name с приоритетом 2 находится в
базе данных Users db',
            'Процедура login с приоритетом 1 находится в базе
данных Main db',
            'Процедура login с приоритетом 1 находится в базе
данных Users db'
        ]
        self.assertEqual(result, true_result)


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

Результат выполнения программы:

```
Testing started at 15:33 ...


Ran 3 tests in 0.002s


OK
```