ИУ5-32Б

Ушаков Л. И.

Отчёт по РК2 по курсу ПиК ЯП, вариант 14Б

Задание:

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).

Листинг кода:

```
from dataclasses import dataclass
from itertools import groupby

import unittest

@dataclass
class CDDisk:
"""
Класс объекта "CD-диск"
"""
id: int
library_id: int
name: str
capacity: int



@dataclass
class CDLibrary:
"""
Класс объекта "Библиотека CD-дисков"
"""
id: int
name: str



@dataclass
```

```
class CDDiskInCDLibrary:
"""
Класс связи многие-ко-многим объектов
"CD-диск" и "Библиотека CD-дисков"
"""
disk_id: int
library_id: int

def disk_capacity(entry):
return entry[0].capacity

def disk_name(entry):
return entry[0].name

def library_name(entry):
return entry[1].name

def query_1(table):
return map(
lambda entry: (
f"{disk_name(entry):<32}\t"
f"{disk_capacity(entry)} MB\t"
f"{library_name(entry)}"
),
sorted(table, key=disk_capacity),
)



def query_2(table):
grouped = groupby(
sorted(table, key=library_name),
key=library_name,
)

return map(
lambda entry: f"{entry[0]:<32}\t{entry[1]}",
sorted(
(
(name, len(list(items)))
for name, items in grouped
),
key=lambda entry: entry[1],
),
)

def query_3(table):
```

```
return map(
lambda entry: (
f"{disk_name(entry):<32}\t{library_name(entry)}"
),
sorted(
filter(
lambda entry: disk_name(entry).endswith("Pack"),
table,
),
key=disk_name,
),
)

class UnitTests(unittest.TestCase):
libraries = [
CDLibrary(1, "Main Library"),
CDLibrary(2, "Branch Library"),
]

disks = [
CDDisk(1, 1, "Windows Pack", 700),
CDDisk(2, 1, "Classical Pack", 650),
CDDisk(3, 2, "Movie Pack", 4700),
]

library_disks = [
# Основные связи
CDDiskInCDLibrary(1, 1),
CDDiskInCDLibrary(2, 1),
CDDiskInCDLibrary(3, 2),
# Дополнительные перекрёстные связи
CDDiskInCDLibrary(2, 2),
]

def test_query_1(self):
table = list(
map(
lambda disk: (
disk,
next(
filter(
lambda lib: disk.library_id == lib.id,
self.libraries,
)
),
),
self.disks,
```

```
)
)

expected = [
'Classical Pack \t650 MB\tMain Library',
'Windows Pack \t700 MB\tMain Library',
'Movie Pack \t4700 MB\tBranch Library'
]
result = list(query_1(table))

self.assertEqual(result, expected)

def test_query_2(self):
table = list(
map(
lambda disk: (
disk,
next(
filter(
lambda lib: disk.library_id == lib.id,
self.libraries,
)
),
),
self.disks,
)
)

expected = [
'Branch Library \t1',
'Main Library \t2'
]

result = list(query_2(table))

self.assertEqual(result, expected)

def test_query_3(self):
table_mtm = list(
map(
lambda link: (
next(
filter(
lambda disk: link.disk_id == disk.id,
self.disks,
)
```

```
),
next(
filter(
lambda lib: link.library_id == lib.id,
self.libraries,
)
),
),
self.library_disks,
)
)
expected = [
'Classical Pack \tMain Library',
'Classical Pack \tBranch Library',
'Movie Pack \tBranch Library',
'Windows Pack \tMain Library'
]

result = list(query_3(table_mtm))

self.assertEqual(result, expected)



def main():
libraries = [
CDLibrary(1, "Main Library"),
CDLibrary(2, "Branch Library"),
CDLibrary(3, "University Archive"),
]

disks = [
CDDisk(1, 1, "Windows Pack", 700),
CDDisk(2, 1, "Classical Pack", 650),
CDDisk(3, 2, "Movie Pack", 4700),
CDDisk(4, 3, "Backup Collection", 4900),
CDDisk(5, 1, "Linux Pack", 780),
CDDisk(6, 2, "Pop Pack", 640),
CDDisk(7, 2, "Educational Pack", 3500),
CDDisk(8, 3, "Photo Archive", 4600),
CDDisk(9, 1, "Software Pack", 1200),
CDDisk(10, 3, "Database Dump", 5000),
]

library_disks = [
# Основные связи
CDDiskInCDLibrary(1, 1),
```

```
CDDiskInCDLibrary(2, 1),
CDDiskInCDLibrary(3, 2),
CDDiskInCDLibrary(4, 3),
CDDiskInCDLibrary(5, 1),
CDDiskInCDLibrary(6, 2),
CDDiskInCDLibrary(7, 2),
CDDiskInCDLibrary(8, 3),
CDDiskInCDLibrary(9, 1),
CDDiskInCDLibrary(10, 3),
# Дополнительные перекрёстные связи
CDDiskInCDLibrary(2, 2),
CDDiskInCDLibrary(1, 3),
CDDiskInCDLibrary(7, 1),
CDDiskInCDLibrary(9, 2),
]

# Запрос Б-1
print("------- Запрос Б-1 -------")
print(
*query_1(table),
sep="\n",
)
print()

# Запрос Б-2
print("------- Запрос Б-2 -------")
print(
*query_2(table),
sep="\n",
)
print()

# Таблица связей многие-ко-многим

table = list(
map(
lambda link: (
next(
filter(
lambda disk: link.disk_id == disk.id,
disks,
)
),
next(
filter(
lambda lib: link.library_id == lib.id,
libraries,
```

```
)
),
),
library_disks,
)
)

# Запрос Б-3
print("------- Запрос Б-3 -------")
print(
*query_3(table),
sep="\n",
)



if __name__ == "__main__":
unittest.main(argv=[''], exit=False)
```

Результат выполнения:

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```