Московский государственный технический университет имени Н.Э. Баумана

Факультет «Информатика и вычислительная техника»
Кафедра ИУ5 «Система обработки информации и управления»

Отчет по рубежному контролю №2
«Разработка тестов на языке Python»

Выполнил
студент ИУ5-35Б
Рябова С.А.

Проверил
Преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва 2024

Описание задания

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).

Текст программ

Main.py

```
from operator import itemgetter


class DataColumn:
    def __init__(self, id, text_data, size, table_id):
        self.id = id
        self.text_data = text_data
        self.size = size
        self.table_id = table_id


class DataTable:
    def __init__(self, id, table_name):
        self.id = id
        self.table_name = table_name


class TableColumns:
    def __init__(self, column_id, table_id):
        self.column_id = column_id
        self.table_id = table_id


tables = [DataTable(1, "Clients"),
          DataTable(2, "Transactions"),
          DataTable(3, "Shipments"),
          DataTable(4, "Stocks"),
          DataTable(5, "Losses"),
          DataTable(6, "Profits")]

columns = [DataColumn(11, "Name", 1024, 1),
           DataColumn(12, "Date", 128, 1),
           DataColumn(21, "Type", 1, 2),
           DataColumn(22, "Cost", 1024, 2),
           DataColumn(31, "Destination", 4096, 3),
```

```
            DataColumn(32, "Mass", 8, 3),
            DataColumn(41, "Company", 1024, 4),
            DataColumn(51, "Investment", 1024, 5),
            DataColumn(52, "Size", 1024, 5),
            DataColumn(61, "Investment", 1024, 6),
            DataColumn(62, "Size", 1024, 6)
            ]

tabCol = [
    TableColumns(11, 1),
    TableColumns(12, 1),
    TableColumns(12, 3),
    TableColumns(12, 5),
    TableColumns(21, 2),
    TableColumns(22, 2),
    TableColumns(31, 3),
    TableColumns(32, 3),
    TableColumns(41, 1),
    TableColumns(41, 4),
    TableColumns(51, 5),
    TableColumns(52, 5),
    TableColumns(61, 6),
    TableColumns(62, 6)
]


def gen_otm(cols, tabs):
    one_to_many = [(c.text_data, c.size, t.table_name)
                   for c in cols
                   for t in tabs
                   if c.table_id == t.id]
    return one_to_many


def gen_mtm(cols, tabs, tabcol):
    many_to_many_temp = [(t.table_name, tc.table_id,
tc.column_id)
                         for t in tabs
                         for tc in tabcol
                         if t.id == tc.table_id]
    many_to_many = [(c.text_data, c.size, table_name)
                    for table_name, table_id, column_id in
many_to_many_temp
                    for c in cols if c.id == column_id]
    return many_to_many


def g1(one_to_many):
    g1_temp = list(filter(lambda i: i[2][0] == 'S',
one_to_many))
    g1_t = list(set(list(x[2] for x in g1_temp)))
    g1_c = list(set(list(x[0] for x in g1_temp)))
    return g1_t, g1_c
```

```
def g2(one_to_many):
    g2_temp1 = list((x[2], x[1]) for x in one_to_many)
    g2_temp2 = list(reversed(sorted(
        list(set(list(filter(lambda i: i[1] == max(list(x[1] for
x in one_to_many if x[2] == i[0])), g2_temp1)))),
        key=itemgetter(1))))
    return g2_temp2


def g3(many_to_many):
    return sorted(list((x[0], x[2]) for x in many_to_many),
key=itemgetter(1))

def main():
    one_to_many = gen_otm(columns, tables)
    many_to_many = gen_mtm(columns, tables, tabCol)

    # Г1
    answer1, answer2 = g1(one_to_many)
    print(answer1)
    print(answer2)

    # Г2
    print(g2(one_to_many))

    # Г3
    print(g3(many_to_many))

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

Test_g1.py

```
from project.main import g1, gen_otm, DataTable, DataColumn


def test_g1():
    tables = [DataTable(1, "Se"),
              DataTable(2, "Sx"),
              DataTable(3, "S"),
              DataTable(4, "A")]

    columns = [DataColumn(11, "A", 1024, 1),
               DataColumn(12, "B", 1024, 1),
               DataColumn(21, "B", 1024, 2),
               DataColumn(31, "C", 1024, 3),
               DataColumn(41, "D", 1024, 4)
```

```
                ]
    a1, a2 = g1(gen_otm(columns, tables))
    assert sorted(a1) == sorted(["Se", "Sx", "S"]) and
sorted(a2) == sorted(["A", "B", "C"])
```

Test_g2.py

```
from project.main import g2, gen_otm, DataTable, DataColumn


def test_g2():
    tables = [DataTable(1, "Se"),
              DataTable(2, "Sx"),
              DataTable(3, "S"),
              DataTable(4, "A")]

    columns = [DataColumn(11, "A", 2, 1),
               DataColumn(12, "B", 1, 1),
               DataColumn(21, "B", 3, 2),
               DataColumn(22, "C", 2, 2),
               DataColumn(31, "C", 10, 3),
               DataColumn(32, "D", 11, 3),
               DataColumn(41, "E", 20, 4),
               DataColumn(42, "F", 0, 4)
               ]

    a = g2(gen_otm(columns, tables))
    assert a == [('A', 20), ('S', 11), ('Sx', 3), ('Se', 2)]
```

Test_g3.py

```
from project.main import g3, gen_mtm, DataTable, DataColumn,
TableColumns


def test_g3():
    tables = [DataTable(1, "D"),
              DataTable(2, "C"),
              DataTable(3, "A"),
              DataTable(4, "B")]

    columns = [DataColumn(11, "1", 2, 1),
               DataColumn(12, "2", 1, 1),
               DataColumn(21, "3", 3, 2),
               DataColumn(22, "4", 2, 2),
               DataColumn(31, "5", 10, 3),
               DataColumn(32, "6", 11, 3),
               DataColumn(41, "7", 20, 4),
               DataColumn(42, "8", 0, 4)
               ]
    tabCol = [
```

```
        TableColumns(11, 1),
        TableColumns(12, 1),
        TableColumns(12, 2),
        TableColumns(21, 2),
        TableColumns(22, 2),
        TableColumns(31, 3),
        TableColumns(32, 3),
        TableColumns(41, 4),
        TableColumns(42, 4)
    ]

    a = g3(gen_mtm(columns, tables, tabCol))
    assert a == [('5', 'A'), ('6', 'A'), ('7', 'B'), ('8', 'B'),
('2', 'C'), ('3', 'C'), ('4', 'C'), ('1', 'D'),
                 ('2', 'D')]
```

Экранные формы выполнения