Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Радиотехнический»

КАФЕДРА ИУ-5 «Системы обработки информации и управления»

Отчет по лабораторной работе № 4 по курсу

Разработка интернет-приложений

Тема работы: "Шаблоны проектирования и модульное тестирование в Python"

Выполнил: Мирсонов В. А.
Группа: РТ5-51Б

Дата выполнения: «26» октября 2020 г.

Подпись:_____

Проверил: Гапанюк Ю. Е.

Дата проверки: «26» октября 2020 г.

Подпись:_____

Москва, 2020 г.

Содержание

Цель лабораторной работы - изучение реализации шаблонов проектирования и возможностей модульного тестирования в языке Python.

Общее описание задания

1. Необходимо для произвольной предметной области реализовать три шаблона проектирования: один порождающий, один структурный и один поведенческий.
2. Для каждой реализации шаблона необходимо написать модульный тест. В модульных тестах необходимо применить следующие технологии:
    - TDD - фреймворк.
    - BDD - фреймворк.
    - Создание Mock-объектов.

Выполнение лабораторной работы

Реализация порождающего шаблона

Фабричный шаблон

Реализация для предметной области работы с личными данными аккаунтов пользователей мобильного оператора и почты и их аутентификации.

```
from __future__ import annotations
from abc import ABC, abstractmethod
from enum import Enum


class User(ABC):
    """Абстрактный пользователь (создатель)"""
    pass


class Tele2User(User):
    """Пользователь(создатель записи) Теле2 с номером телефона"""

    def __init__(self, phone):
        print(f'Создан пользователь Теле2 с номером телефона: {phone}')


class BMSTUMailUser(User):
    """Пользователь(создатель записи) почты BMSTU с почтовым адресом email"""

    def __init__(self, email):
        print(f'Создан пользователь Теле2 с номером телефона: {email}')


class SelfData(ABC):
    """Абстрактные учетные данные пользователей"""
    pass


class Tele2SelfData(SelfData):
    """Учетные данные пользователя Теле2"""

    def __init__(self, phone, password):
        self._phone = phone
```

```
        self._password = password

    def print_phone(self):
        return self._phone

    def print_password(self):
        return self._password


class BMSTUSelfData(SelfData):
    """Учетные данные пользователя BMSTU почты"""

    def __init__(self, email, password):
        self._email = email
        self._password = password

    def print_email(self):
        return self._email

    def print_password(self):
        return self._password


class Auth(ABC):
    @abstractmethod
    def authenticate(self, selfdate: SelfData) -> User:
        pass


class Tele2Auth(Auth):
    """Аутентификатор Теле2
    Переопределяет абстрактный метод и возвращает продукт - учетные данные пользователя"""

    def authenticate(self, selfdate: Tele2SelfData) -> Tele2User:
        print(f'Пользователь аутентифицирован по номеру телефона: {selfdate.print_phone()}')
        return Tele2User(selfdate.print_phone())


class BMSTUAuth(Auth):
    """Аутентификатор BMSTU почты
    Переопределяет абстрактный метод и возвращает продукт - учетные данные пользователя"""

    def authenticate(self, selfdate: BMSTUSelfData) -> BMSTUMailUser:
        print(f'Пользователь аутентифицирован по номеру телефона: {selfdate.print_email()}')
        return BMSTUMailUser(selfdate.print_email())


"""Метод не завязан на конкретную реализацию продукта.
Логика метода аутентификации: метод принимает аутентификацию,
учетные данные и отдает некие сформированные данные пользователя"""


def authenticate(auth: Auth, selfdata: SelfData) -> User:
    return auth().authenticate(selfdata)


phone = '+79162002189'
password = '123456'
selfdateT = Tele2SelfData(phone, password)
userT: Tele2User = authenticate(Tele2Auth, selfdateT)

email = 'mirsonovva@bmstu.ru'
password_email = '654321'
```

```
selfdateB = BMSTUSelfData(email, password_email)
userB: BMSTUMailUser = authenticate(BMSTUAuth, selfdateB)
```

Экранная форма

```
Пользователь аутентифицирован по номеру телефона: +79162002189
Создан пользователь Теле2 с номером телефона: +79162002189
Пользователь аутентифицирован по номеру телефона: mirsonovva@bmstu.ru
Создан пользователь Теле2 с номером телефона: mirsonovva@bmstu.ru
```

Реализация структурного шаблона

Адаптер

Реализация для предметной области реализации адаптивности текста под правило шифра цезаря.

```
def cesars(strx):
    str2 = ''
    for i in range(len(strx)):
        temp = ord(strx[i])
        str2 = str2 + chr(temp - 1)
    return str2


class Target:
    """Целевой класс объявляет интерфейс, с которым может работать клиентский код."""

    def request(self) -> str:
        return "Цель - классическое поведение целевого объекта (шифр цезаря -1)"


class Adaptee:
    """
    Адаптируемый класс содержит некоторое полезное поведение, но его интерфейс
    несовместим с существующим клиентским кодом. Адаптируемый класс нуждается в
    некоторой доработке, прежде чем клиентский код сможет его использовать.
    """

    def specific_request(self) -> str:
        return 'J!mpwf!Qzuipo'''


class Adapter(Target, Adaptee):
    """
    Адаптер делает интерфейс Адаптируемого класса совместимым с целевым
    интерфейсом благодаря множественному наследованию.
    """

    def request(self) -> str:
        str1 = self.specific_request()
        return f"Адаптер: (адаптировано) {cesars(str1)}"


def client_code(target: "Target") -> None:
    """
    Клиентский код поддерживает все классы, использующие интерфейс Target.
```

```
    """

    print(target.request(), end="")


def main():
    print("Клиент может корректно работать только с целевым классом:")
    target = Target()
    client_code(target)
    print("\n")

    adaptee = Adaptee()
    print("Клиент: адаптируемый класс имеет непонятный интерфейс, я совсем не могу разобрать послание")
    print(f"Адаптируемый класс: {adaptee.specific_request()}", end="\n\n")

    print("Клиент: но у меня получится работать с ним при помощи адаптера")
    adapter = Adapter()
    client_code(adapter)


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

Экранная форма

```
Клиент может корректно работать только с целевым классом:
Цель - классическое поведение целевого объекта (шифр цезаря -1)

Клиент: адаптируемый класс имеет непонятный интерфейс, я совсем не могу разобрать послание
Адаптируемый класс: J!mpwf!Qzuipo"

Клиент: но у меня получится работать с ним при помощи адаптера
Адаптер: (адаптировано) I love Python!
```

Реализация поведенческого шаблона

Итератор

Реализация для предметной области достопримечательностей города Санкт-Петербург и вывода их списка в различном порядке и с различных стартовых «позиций».

```
from __future__ import annotations
from collections.abc import Iterable, Iterator
from typing import Any, List


"""
Для создания итератора в Python есть два абстрактных класса из встроенного
модуля collections - Iterable, Iterator. Нужно реализовать метод __iter__() в
итерируемом объекте (списке), а метод __next__() в итераторе.
"""


class AlphabeticalOrderIterator(Iterator):
    """
    Конкретные Итераторы реализуют различные алгоритмы обхода. Эти классы
    постоянно хранят текущее положение обхода.
    """
```

```
    """
    Атрибут _position хранит текущее положение обхода. У итератора может быть
    множество других полей для хранения состояния итерации, особенно когда он
    должен работать с определённым типом коллекции.
    """
    _position: int = None

    """
    Этот атрибут указывает направление обхода.
    """
    _reverse: bool = False

    def __init__(self, collection: WordsCollection, reverse: bool = False, beginNumber: int = 0) -> None:
        self._collection = collection
        self._reverse = reverse
        self._beginnumber = beginNumber
        self._position = -1 if reverse else 0
        if self._beginnumber != 0:
            self._position = -(self._beginnumber+1) if reverse else self._beginnumber

    def __next__(self):
        """
        Метод __next __() должен вернуть следующий элемент в последовательности.
        При достижении конца коллекции и в последующих вызовах должно вызываться
        исключение StopIteration.
        """
        try:
            value = self._collection[self._position]
            self._position += -1 if self._reverse else 1
        except IndexError:
            raise StopIteration()

        return value


class WordsCollection(Iterable):
    """
    Конкретные Коллекции предоставляют один или несколько методов для получения
    новых экземпляров итератора, совместимых с классом коллекции.
    """

    def __init__(self, collection: List[Any] = []) -> None:
        self._collection = collection

    def __iter__(self) -> AlphabeticalOrderIterator:
        """
        Метод __iter __() возвращает объект итератора, по умолчанию мы возвращаем
        итератор с сортировкой по возрастанию.
        """
        return AlphabeticalOrderIterator(self._collection)

    def get_reverse_iterator(self) -> AlphabeticalOrderIterator:
        return AlphabeticalOrderIterator(self._collection, True)

    def get_reverse_iterator_from_number(self, numb) -> AlphabeticalOrderIterator:
        return AlphabeticalOrderIterator(self._collection, True, numb)

    def get_iterator_from_number(self, numb) -> AlphabeticalOrderIterator:
        return AlphabeticalOrderIterator(self._collection, False, numb)

    def add_item(self, item: Any):
        self._collection.append(item)
```

```
if __name__ == "__main__":
    collection = WordsCollection()
    collection.add_item("1 - Дворцовая площадь")
    collection.add_item("2 - Зимний дворец")
    collection.add_item("3 - Дворцовый мост")
    collection.add_item("4 - Васильевский остров")
    collection.add_item("5 - Петропавловская крепость")
    collection.add_item("6 - Крейсер 'Аврора'")
    collection.add_item("7 - Марсово поле")
    collection.add_item("8 - Спас на Крови")

    print("Прямая последовательность коллекции достопремечательностей Санкт-Петербурга:\n")
    print("\n".join(collection))
    print("----------------------------------\n\n")

    print("Обратная последовательность коллекции достопремечательностей Санкт-Петербурга:\n")
    print("\n".join(collection.get_reverse_iterator()), end="\n")
    print("----------------------------------\n\n")

    print("Прямая последовательность коллекции достопремечательностей Санкт-Петербурга начиная с
указанного номера ("
          "3 с начала):\n")
    print("\n".join(collection.get_iterator_from_number(2)), end="\n")
    print("----------------------------------\n\n")

    print("ОБратная последовательность коллекции достопремечательностей Санкт-Петербурга начиная с
указанного номера ("
          "3 с конца):\n")
    print("\n".join(collection.get_reverse_iterator_from_number(2)), end="\n")
    print("----------------------------------\n\n")
```

Экранная форма

```
Прямая последовательность коллекции достопремечательностей Санкт-Петербурга:

1 - Дворцовая площадь
2 - Зимний дворец
3 - Дворцовый мост
4 - Васильевский остров
5 - Петропавловская крепость
6 - Крейсер 'Аврора'
7 - Марсово поле
8 - Спас на Крови
---------------------------------


Обратная последовательность коллекции достопремечательностей Санкт-Петербурга:

8 - Спас на Крови
7 - Марсово поле
6 - Крейсер 'Аврора'
5 - Петропавловская крепость
4 - Васильевский остров
3 - Дворцовый мост
2 - Зимний дворец
1 - Дворцовая площадь
---------------------------------
```

```
Прямая последовательность коллекции достопремечательностей Санкт-Петербурга начиная с указанного номера (3 с начала):

3 - Дворцовый мост
4 - Васильевский остров
5 - Петропавловская крепость
6 - Крейсер 'Аврора'
7 - Марсово поле
8 - Спас на Крови
---------------------------------


Обратная последовательность коллекции достопремечательностей Санкт-Петербурга начиная с указанного номера (3 с конца):

6 - Крейсер 'Аврора'
5 - Петропавловская крепость
4 - Васильевский остров
3 - Дворцовый мост
2 - Зимний дворец
1 - Дворцовая площадь
---------------------------------
```

Модульные тесты

TDD – фреймворк

Реализуем модульные тесты для предметной области кафе, реализующего продажу своей продукции внутри заведения. Будем проверять корректность данных заказа и то, насколько хорошо написанная функция создания записи о заказе справляется со своей задачей.

```
from abc import ABC
import unittest


menu = [
    'Пирожок с вишней',
    'Пирожок с яблоком',
    'Пирожок с малиной',
    'Кофе капучино',
    'Кофе американо',
    'Кофе латте',
]


def orderdetailn(object1):
    args = ['Наименование', 'Кол-во', 'Цена']
    for d in object1:
        elem = {}
        for arg in args:
            if arg in d.keys() and not d[arg] is None:
                elem[arg] = d[arg]
        if len(elem) != 0:
            yield elem


def is_number(str1):
    try:
        int(str1)
        return True
    except ValueError:
        return False


class TakeOrder(ABC):
    """Абстрактный класс с данными по заказу"""
    pass


class TakeOrderIn(TakeOrder):
    """Выдача заказа внутри кафе"""

    def __init__(self, orderNum=0, payCheck=True, order=None, tableNum=0):
        self._orderNum = orderNum
        self._payCheck = payCheck
        self._order = order
        self._tableNum = tableNum

    def ordernum(self):
        return self._orderNum

    def paycheck(self):
        return self._payCheck
```

```
def orderdetail(self):
    args = ['Наименование', 'Кол-во', 'Цена']
    for d in self._order:
        elem = {}
        for arg in args:
            if arg in d.keys() and not d[arg] is None:
                elem[arg] = d[arg]
        if len(elem) != 0:
            yield elem

def tablenum(self):
    return self._tableNum

def printdetails(self):
    print("Номер заказа: ", self.ordernum(), "\nОплата "
                                             "заказа: ", self.paycheck(), "\nЭлементы заказа:",
          str(list(self.orderdetail()())),
          "\nНомер стола: ", self.tablenum(), "\n//////////////////////")

def ordercheckn(self, orderNum, payCheck, order, tableNum):
    isError = False
    if not is_number(orderNum):
        isError = True
        return "---------Ошибка!---------\nНеверный формат номера заказа - отмена заказа"
    if not (str(payCheck) == str(True) or str(payCheck) == str(False)):
        isError = True
        return "---------Ошибка!---------\nНеверный формат отметки об оплате заказа - отмена заказа"
    if not is_number(tableNum):
        isError = True
        return "---------Ошибка!---------\nНеверный формат номера столика - отмена заказа"
    if not (isError):
        if orderNum <= 0:
            return "---------Ошибка!---------\nЗаказу присвоен неверный номер - отмена заказа"
        else:
            if not payCheck:
                return "---------Ошибка!---------\nЗаказ не оплачен - отмена заказа"
            else:
                if len(order) == 0:
                    return "---------Ошибка!---------\nЭлементы заказа отсутствуют - отмена заказа"
                else:
                    if (1 > tableNum) or (tableNum > 20):
                        return "---------Ошибка!---------\nУказанного в заказе столика не существует - отмена " \
                               "заказа "
                    else:
                        count = 0
                        costTotal = 0
                        menucheck = 0
                        countT = 0
                        args = ['Наименование', 'Кол-во', 'Цена']
                        if len(order) > 0:
                            for d in order:
                                elem = {}
                                for arg in args:
                                    if arg in d.keys() and not d[arg] is None:
                                        if arg == 'Наименование':
                                            menucheck = 0
                                            for i in menu:
                                                if str(d[arg]) == str(i):
                                                    menucheck = 1
                                                    break
                                            if menucheck == 0:
                                                return "---------Ошибка!---------\nЭлемент указанный в заказе не " \
```

```
                                    "присутвует в меню - отмена заказа"
                            if arg == 'Кол-во':
                                countT = 0
                                isError1 = False
                                if not is_number(d[arg]):
                                    isError1 = True
                                    return "---------Ошибка!---------\n" \
                                           "Неверный формат кол-ва элемента заказа - отмена заказа "
                                if (int(d[arg]) <= 0 or int(d[arg]) > 1000) and (isError1 == False):
                                    return "---------Ошибка!---------\n" \
                                           "Количество товаров указано неверно - отмена заказа"
                                else:
                                    countT = countT + int(d[arg])
                                    count = count + int(d[arg])
                            if arg == 'Цена':
                                # costTotal = 0
                                isError2 = False
                                if not is_number(d[arg]):
                                    isError2 = True
                                    return "---------Ошибка!---------\nНеверный формат цены элемента " \
                                           "заказа - отмена заказа "
                                if (int(d[arg]) <= 0 or int(d[arg]) > 1000) and (isError2 == False):
                                    return "---------Ошибка!---------\nЦена товаров указано неверно " \
                                           "- отмена заказа"
                                else:
                                    costTotal = costTotal + int(d[arg]) * countT
                        if count != 0 and costTotal != 0 and menucheck == 1:
                            return "Заказ номер " \
                                   "{0}, общей стоимостью {1} и общим количеством {2}".format(
                                orderNum,
                                costTotal,
                                count)

    def ordercheck(self):
        isError = False
        if not is_number(self._orderNum):
            isError = True
            return "---------Ошибка!---------\nНеверный формат номера заказа - отмена заказа"
        if not (str(self._payCheck) == str(True) or str(self._payCheck) == str(False)):
            isError = True
            return "---------Ошибка!---------\nНеверный формат отметки об оплате заказа - отмена заказа"
        if not is_number(self._tableNum):
            isError = True
            return "---------Ошибка!---------\nНеверный формат номера столика - отмена заказа"
        if not (isError):
            if self._orderNum <= 0:
                return "---------Ошибка!---------\nЗаказу присвоен неверный номер - отмена заказа"
            else:
                if not self._payCheck:
                    return "---------Ошибка!---------\nЗаказ не оплачен - отмена заказа"
                else:
                    if len(self._order) == 0:
                        return "---------Ошибка!---------\nЭлементы заказа отсутствуют - отмена заказа"
                    else:
                        if (1 > self._tableNum) or (self._tableNum > 20):
                            return "---------Ошибка!---------\nУказанного в заказе столика не существует - отмена " \
                                   "заказа "
                        else:
                            count = 0
                            costTotal = 0
                            menucheck = 0
                            countT = 0
                            args = ['Наименование', 'Кол-во', 'Цена']
```

```
                    if len(self._order) > 0:
                        for d in self._order:
                            elem = {}
                            for arg in args:
                                if arg in d.keys() and not d[arg] is None:
                                    if arg == 'Наименование':
                                        menucheck = 0
                                        for i in menu:
                                            # print(str(d[arg]))
                                            # print(str(i))
                                            if str(d[arg]) == str(i):
                                                menucheck = 1
                                                break
                                        if menucheck == 0:
                                            return "---------Ошибка!---------\nЭлемент указанный в заказе не " \
                                                   "присутвует в меню - отмена заказа"
                                    if arg == 'Кол-во':
                                        countT = 0

                                        isError1 = False
                                        if not is_number(d[arg]):
                                            isError1 = True
                                            return "---------Ошибка!---------\nНеверный формат кол-ва элемента " \
                                                   "заказа - отмена заказа "
                                        if (int(d[arg]) <= 0 or int(d[arg]) > 1000) and (isError1 == False):
                                            return "---------Ошибка!---------\nКоличество товаров указано неверно " \
                                                   "- отмена заказа"
                                        else:
                                            countT = countT + int(d[arg])
                                            count = count + int(d[arg])
                                    if arg == 'Цена':
                                        # costTotal = 0
                                        isError2 = False
                                        if not is_number(d[arg]):
                                            isError2 = True
                                            return "---------Ошибка!---------\nНеверный формат цены элемента " \
                                                   "заказа - отмена заказа "
                                        if (int(d[arg]) <= 0 or int(d[arg]) > 1000) and (isError2 == False):
                                            return "---------Ошибка!---------\nЦена товаров указано неверно " \
                                                   "- отмена заказа"
                                        else:
                                            costTotal = costTotal + int(d[arg]) * countT
                        if count != 0 and costTotal != 0 and menucheck == 1:
                            print(
                                '//////////////////////\nЗаказ стоимостью общей {0} с общим количеством '
                                'наименований товаров {1} '
                                'успешно сформирован'.format(costTotal, count))
                            return self.printdetails()


class TestInS(unittest.TestCase):
    # Часть проверок с параметрами, не связанными с элментами заказа (т.е. не наименование, кол-во, цена)
    # будет произведена на основании значений для заказа по умолчанию (установлены при инициализации)
    def setUp(self):
        self.order = TakeOrderIn()
        self.default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': 100}]
        self.default_order_result = "Заказ номер 1, общей стоимостью 200 и общим количеством 2"

    def test_default_order(self):
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, 1, True, self.default_order_details, 1),
                         self.default_order_result, "defualt values aren`t okey")

    def test_correct_order_num_by_int(self):
```

```
        # Проверяем, является ли номер заказа целым числом
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, "ab1", True, self.default_order_details, 1),
                         self.default_order_result)

    def test_correct_order_num_by_value(self):
        # Проверяем, является ли номер присвоенный заказу положительным числом
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, -2, True, self.default_order_details, 1),
                         self.default_order_result)

    def test_correct_order_payCheck_by_bool(self):
        # Проверяем, имеет ли отметка об оплате заказа верный формат для корректной работы
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, 1, 'Оплачен', self.default_order_details, 1),
                         self.default_order_result)

    def test_correct_order_payCheck_by_value(self):
        # Проверяем, оплачен ли заказ
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, 1, False, self.default_order_details, 1),
                         self.default_order_result)

    def test_correct_table_num_by_int(self):
        # Проверяем, является ли номер столика целым числом
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, 1, True, self.default_order_details, "ab1"),
                         self.default_order_result)

    def test_correct_table_num_by_value(self):
        # Проверяем, является ли номер столика положительным числом от 1 до 20
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, 1, True, self.default_order_details, -2),
                         self.default_order_result)

    def test_correct_order_elements_not_null(self):
        # Проверяем, существуют ли элементы заказа - ошибка выводится первой
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, 1, True, {}, 1),
                         self.default_order_result)

    def test_correct_order_elements_in_menu(self):
        # Проверяем, присутсвуют ли элементы заказа в меню
        order_menu_check = [{'Наименование': 'Пирожок не с вишней НЕ ИЗ МЕНЮ', 'Кол-во': 2, 'Цена': 100}]
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, 1, True, order_menu_check, 1),
                         self.default_order_result)

    def test_correct_order_elements_kol_vo_by_int(self):
        # Проверяем, является ли количество элементов заказа целым числом
        order_menu_check = [{'Наименование': 'Пирожок с вишней', 'Кол-во': "ab2", 'Цена': 100}]
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, 1, True, order_menu_check, 1),
                         self.default_order_result)

    def test_correct_order_elements_kol_vo_by_value(self):
        # Проверяем, является ли количество элементов заказа целым положительным числом
        order_menu_check = [{'Наименование': 'Пирожок с вишней', 'Кол-во': -2, 'Цена': 100}]
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, 1, True, order_menu_check, 1),
                         self.default_order_result)

    def test_correct_order_elements_cost_by_int(self):
        # Проверяем, является ли цена элементов заказа целым числом
        order_menu_check = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': "ab100"}]
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, 1, True, order_menu_check, 1),
                         self.default_order_result)

    def test_correct_order_elements_cost_by_value(self):
        # Проверяем, является ли цена элементов заказа целым положительным числом
        order_menu_check = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': -100}]
        self.assertEqual(TakeOrderIn.ordercheckn(self.order, 1, True, order_menu_check, 1),
                         self.default_order_result)
```

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

Экранные формы

```
Заказ номер 1, общей стоимостью 200 и общим количеством 2 != ---------Ошибка!---------
Неверный формат цены элемента заказа - отмена заказа



+ Заказ номер 1, общей стоимостью 200 и общим количеством 2- ---------Ошибка!---------
- Неверный формат цены элемента заказа - отмена заказа



Заказ номер 1, общей стоимостью 200 и общим количеством 2 != ---------Ошибка!---------
Цена товаров указано неверно - отмена заказа



+ Заказ номер 1, общей стоимостью 200 и общим количеством 2- ---------Ошибка!---------
- Цена товаров указано неверно - отмена заказа



Заказ номер 1, общей стоимостью 200 и общим количеством 2 != ---------Ошибка!---------
Элемент указанный в заказе не присутвует в меню - отмена заказа



+ Заказ номер 1, общей стоимостью 200 и общим количеством 2- ---------Ошибка!---------
- Элемент указанный в заказе не присутвует в меню - отмена заказа



Заказ номер 1, общей стоимостью 200 и общим количеством 2 != ---------Ошибка!---------
Неверный формат кол-ва элемента заказа - отмена заказа



+ Заказ номер 1, общей стоимостью 200 и общим количеством 2- ---------Ошибка!---------
- Неверный формат кол-ва элемента заказа - отмена заказа



Заказ номер 1, общей стоимостью 200 и общим количеством 2 != ---------Ошибка!---------
Количество товаров указано неверно - отмена заказа



+ Заказ номер 1, общей стоимостью 200 и общим количеством 2- ---------Ошибка!---------
- Количество товаров указано неверно - отмена заказа
```

Заказ номер 1, общей стоимостью 200 и общим количеством 2 != ---------Ошибка!---------
Элементы заказа отсутствуют - отмена заказа

+ Заказ номер 1, общей стоимостью 200 и общим количеством 2- ---------Ошибка!---------
- Элементы заказа отсутствуют - отмена заказа

Заказ номер 1, общей стоимостью 200 и общим количеством 2 != ---------Ошибка!---------
Неверный формат номера заказа - отмена заказа

+ Заказ номер 1, общей стоимостью 200 и общим количеством 2- ---------Ошибка!---------
- Неверный формат номера заказа - отмена заказа

Заказ номер 1, общей стоимостью 200 и общим количеством 2 != ---------Ошибка!---------
Заказу присвоен неверный номер - отмена заказа

+ Заказ номер 1, общей стоимостью 200 и общим количеством 2- ---------Ошибка!---------
- Заказу присвоен неверный номер - отмена заказа

Заказ номер 1, общей стоимостью 200 и общим количеством 2 != ---------Ошибка!---------
Неверный формат отметки об оплате заказа - отмена заказа

+ Заказ номер 1, общей стоимостью 200 и общим количеством 2- ---------Ошибка!---------
- Неверный формат отметки об оплате заказа - отмена заказа

Заказ номер 1, общей стоимостью 200 и общим количеством 2 != ---------Ошибка!---------
Заказ не оплачен - отмена заказа

+ Заказ номер 1, общей стоимостью 200 и общим количеством 2- ---------Ошибка!---------
- Заказ не оплачен - отмена заказа

Заказ номер 1, общей стоимостью 200 и общим количеством 2 != ---------Ошибка!---------
Неверный формат номера столика - отмена заказа

+ Заказ номер 1, общей стоимостью 200 и общим количеством 2- ---------Ошибка!---------
- Неверный формат номера столика - отмена заказа

```
Заказ номер 1, общей стоимостью 200 и общим количеством 2 != ---------Ошибка!---------
Указанного в заказе столика не существует - отмена заказа



+ Заказ номер 1, общей стоимостью 200 и общим количеством 2- ---------Ошибка!---------
- Указанного в заказе столика не существует - отмена заказа


Ran 13 tests in 0.090s

FAILED (failures=12)
```

Создание Mock-объектов

Реализуем модульные тесты для предметной области кафе, реализующего продажу своей продукции в качестве доставки. Будем проверять корректность данных заказа и то, насколько хорошо написанная функция создания записи о заказе справляется со своей задачей.

```python
import unittest
import time
from unittest.mock import patch
from lab4_python.FullOrderToZakaz import is_number
from lab4_python.FullOrderToZakaz import TakeOrder
from lab4_python.FullOrderToZakaz import menu

curiers = [
    'Иванов И.И.',
    'Сидоркин К.В.',
    'Мирсонов В.А.,'
    'Киряев А.О.',
]


class TakeOrderOut(TakeOrder):
    """Доставка заказа курьером кафе по указанному адресу"""

    def __init__(self, orderNum=0, payCheck=False, order=None, adress='', curier=''):
        self._orderNum = orderNum
        self._payCheck = payCheck
        self._order = order
        self._adress = adress
        self._curier = curier

    def ordernum(self):
        return self._orderNum

    def paycheck(self):
        return self._payCheck

    def orderdetail(self):
        args = ['Наименование', 'Кол-во', 'Цена']
        for d in self._order:
            elem = {}
            for arg in args:
                if arg in d.keys() and not d[arg] is None:
                    elem[arg] = d[arg]
```

```
        if len(elem) != 0:
            yield elem

    def adress(self):
        return self._adress

    def curier(self):
        return self._curier

    def printdetails(self):
        print("////////////////////////\nНомер заказа: ", self.ordernum(), "\nОплата заказа: ", self.paycheck(),
              "\nЭлементы заказа:", str(list(self.orderdetail())),
              "\nАдрес доставки: ", self.adress(), "\nКурьер: ", self.curier(), "\n////////////////////////")

    def ordercheck(self, orderNum, payCheck, order, adress, curier):
        isError = False
        time.sleep(5)
        if not is_number(orderNum):
            isError = True
            return "---------Ошибка!---------\nНеверный формат номера заказа - отмена заказа"
        if not (str(payCheck) == str(True) or str(payCheck) == str(False)):
            isError = True
            return "---------Ошибка!---------\nНеверный формат отметки об оплате заказа - отмена заказа"
        if curier == '':
            isError = True
            return "---------Ошибка!---------\nКурьер указан неверно - отмена заказа"
        if not (isError):
            if orderNum <= 0:
                return "---------Ошибка!---------\nЗаказу присвоен неверный номер - отмена заказа"
            else:
                if not payCheck:
                    return "---------Ошибка!---------\nЗаказ не оплачен - отмена заказа"
                else:
                    if len(order) == 0:
                        return "---------Ошибка!---------\nЭлементы заказа отсутствуют - отмена заказа"
                    else:
                        if adress == '':
                            return "---------Ошибка!---------\nУказанного в заказе адреса не существует - отмена " \
                                   "заказа "
                        else:
                            count = 0
                            costTotal = 0
                            menucheck = 0
                            countT = 0
                            correct_curier = 0
                            for i in curiers:
                                if curier == i:
                                    correct_curier = 1
                                    break
                            if correct_curier == 0:
                                return "---------Ошибка!---------\nКурьер, указанный в информации о заказе не " \
                                       "существует - отмена заказа"
                            args = ['Наименование', 'Кол-во', 'Цена']
                            if len(order) > 0:
                                for d in order:
                                    elem = {}
                                    for arg in args:
                                        if arg in d.keys() and not d[arg] is None:
                                            if arg == 'Наименование':
                                                menucheck = 0
                                                for i in menu:
                                                    if str(d[arg]) == str(i):
                                                        menucheck = 1
```

```
                                    break
                            if menucheck == 0:
                                return "---------Ошибка!---------\nЭлемент указанный в заказе не " \
                                       "присутвует в меню - отмена заказа"
                        if arg == 'Кол-во':
                            countT = 0
                            isError1 = False
                            if not is_number(d[arg]):
                                isError1 = True
                                return "---------Ошибка!---------\n" \
                                       "Неверный формат кол-ва элемента заказа - отмена заказа "
                            if (int(d[arg]) <= 0 or int(d[arg]) > 1000) and (isError1 == False):
                                return "---------Ошибка!---------\n" \
                                       "Количество товаров указано неверно - отмена заказа"
                            else:
                                countT = countT + int(d[arg])
                                count = count + int(d[arg])
                        if arg == 'Цена':
                            # costTotal = 0
                            isError2 = False
                            if not is_number(d[arg]):
                                isError2 = True
                                return "---------Ошибка!---------\nНеверный формат цены элемента " \
                                       "заказа - отмена заказа "
                            if (int(d[arg]) <= 0 or int(d[arg]) > 1000) and (isError2 == False):
                                return "---------Ошибка!---------\nЦена товаров указано неверно " \
                                       "- отмена заказа"
                            else:
                                costTotal = costTotal + int(d[arg]) * countT
                if count != 0 and costTotal != 0 and menucheck == 1 and correct_curier == 1:
                    return 'Заказ номер ' \
                           '{0}, общей стоимостью {1} и общим количеством {2}' \
                           ' на адрес {3}, курьером {4}'.format(
                        orderNum,
                        costTotal,
                        count,
                        adress,
                        curier)


class TestOutS(unittest.TestCase):
    """def setUp(self):
        self.order = TakeOrderOut()
        self.default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': 100}]


    def test_default(self):
        self.assertEqual(TakeOrderOut.ordercheck(self.order, 1, True, self.default_order_details, "г. Москва, ул. 
Пушкина, 9, 12", 'Сидоркин К.В.'),
                         "Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес г. Москва, ул. 
Пушкина, 9, 12")
    """
    """Декоратор patch из unittest.mock заменяет фактическую функцию
    ordercheck ложной функцией, которая ведет себя именно так,
    как мы хотим - фактически переписываем нашу функцию в mock тест так, чтобы не было затрат на время.
    В этом случае наша фиктивная функция всегда возвращает return_value.
    В течение всего теста функция ordercheck заменяется на mock."""

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_correct_order_out_with_mock(self, ordercheck):
        order = TakeOrderOut()
```

```
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': 100}]
        self.assertEqual(
            ordercheck(order, 1, True, default_order_details, "г. Москва, ул. Пушкина, 9, 12", 'Сидоркин К.В.'),
            'Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером Сидоркин К.В.')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_console_order_out_with_mock_with_ornumb_int(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': 100}]
        self.assertEqual(
            ordercheck(order, 'abc', True, default_order_details, "г. Москва, ул. Пушкина, 9, 12", 'Сидоркин К.В.'),
            'Заказ номер abc, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером Сидоркин К.В.')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_console_order_out_with_mock_with_ornumb_by_value(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': 100}]
        self.assertEqual(
            ordercheck(order, -2, True, default_order_details, "г. Москва, ул. Пушкина, 9, 12", 'Сидоркин К.В.'),
            'Заказ номер -2, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером Сидоркин К.В.')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_console_order_out_with_mock_with_payCheck_by_bool(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': 100}]
        self.assertEqual(
            ordercheck(order, 1, 'Оплачен', default_order_details, "г. Москва, ул. Пушкина, 9, 12", 'Сидоркин К.В.'),
            'Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером Сидоркин К.В.')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_console_order_out_with_mock_with_payCheck_by_value(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': 100}]
        self.assertEqual(
            ordercheck(order, 1, False, default_order_details, "г. Москва, ул. Пушкина, 9, 12", 'Сидоркин К.В.'),
            'Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером Сидоркин К.В.')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_console_order_out_with_mock_with_adress_by_str(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': 100}]
        self.assertEqual(
            ordercheck(order, 1, False, default_order_details, 123456, 'Сидоркин К.В.'),
```

```
        'Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "123456", курьером
Сидоркин К.В.')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_console_order_out_with_mock_with_adress_by_value(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': 100}]
        self.assertEqual(
            ordercheck(order, 1, False, default_order_details, '', 'Сидоркин К.В.'),
            'Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "", курьером Сидоркин К.В.')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_console_order_out_with_mock_with_order_elements_not_null(self, ordercheck):
        order = TakeOrderOut()
        self.assertEqual(
            ordercheck(order, 1, False, {}, 'г. Москва, ул. Пушкина, 9, 12', 'Сидоркин К.В.'),
            'Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером Сидоркин К.В.')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_console_order_out_with_mock_with_order_elements_in_menu(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней НЕ ИЗ МЕНЮ', 'Кол-во': 2, 'Цена': 100}]
        self.assertEqual(
            ordercheck(order, 1, False, default_order_details, 'г. Москва, ул. Пушкина, 9, 12', 'Сидоркин К.В.'),
            'Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером Сидоркин К.В.')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_correct_order_out_with_mock_order_elements_kol_vo_by_int(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': "ab2", 'Цена': 100}]
        self.assertEqual(
            ordercheck(order, 1, True, default_order_details, "г. Москва, ул. Пушкина, 9, 12", 'Сидоркин К.В.'),
            'Заказ номер 1, общей стоимостью 200 и общим количеством ab2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером Сидоркин К.В.')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_correct_order_out_with_mock_order_elements_kol_vo_by_value(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': -2, 'Цена': 100}]
        self.assertEqual(
            ordercheck(order, 1, True, default_order_details, "г. Москва, ул. Пушкина, 9, 12", 'Сидоркин К.В.'),
            'Заказ номер 1, общей стоимостью 200 и общим количеством -2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером Сидоркин К.В.')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
```

```
    def test_correct_order_out_with_mock_elements_cost_by_int(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': "ab100"}]
        self.assertEqual(
            ordercheck(order, 1, True, default_order_details, "г. Москва, ул. Пушкина, 9, 12", 'Сидоркин К.В.'),
            'Заказ номер 1, общей стоимостью ab100 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером Сидоркин К.В.')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_correct_order_out_with_mock_elements_cost_by_value(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': -100}]
        self.assertEqual(
            ordercheck(order, 1, True, default_order_details, "г. Москва, ул. Пушкина, 9, 12", 'Сидоркин К.В.'),
            'Заказ номер 1, общей стоимостью -100 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером Сидоркин К.В.')


    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_console_order_out_with_mock_with_curier_in_curiers(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': 100}]
        self.assertEqual(
            ordercheck(order, 1, False, default_order_details, "г. Москва, ул. Пушкина, 9, 12", 'Неопределенный
курьер'),
            'Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером Неопределенный курьер')

    @patch('MockTestToOrderOut.TakeOrderOut.ordercheck',
           return_value='Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес ' \
                        '"г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.')
    def test_console_order_out_with_mock_with_curier_not_null(self, ordercheck):
        order = TakeOrderOut()
        default_order_details = [{'Наименование': 'Пирожок с вишней', 'Кол-во': 2, 'Цена': 100}]
        self.assertEqual(
            ordercheck(order, 1, False, default_order_details, "г. Москва, ул. Пушкина, 9, 12", ''),
            'Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9,
12", '
            'курьером ')


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

Экранная форма

```
Expected :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина,
9, 12", курьером Сидоркин К.В.
Actual   :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "123456", курьером
Сидоркин К.В.


- Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12",
курьером Сидоркин К.В.
?                                                 --------------------------
```

```
+ Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "123456", курьером Сидоркин К.В.
?                                                  ++++



Expected :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
Actual   :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "", курьером Сидоркин К.В.


- Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
?                                                  ----------------------------
+ Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "", курьером Сидоркин К.В.



Expected :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
Actual   :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Неопределенный курьер


- Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
?                                                                 ^^ ^ ^^^^^^^^
+ Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Неопределенный курьер
?                                                                 ^^^^^^ ^^^^^^^^ ^^^ ^^^



Expected :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
Actual   :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером


- Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
?                                                                       ----------
+ Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером



Expected :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
Actual   :Заказ номер -2, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.


- Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
?           ^
+ Заказ номер -2, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
?           ^^



Expected :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
```

```
Actual   :Заказ номер abc, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.


- Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
?           ^
+ Заказ номер abc, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
?           ^^^


Expected :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
Actual   :Заказ номер 1, общей стоимостью ab100 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.


- Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
?                          ^
+ Заказ номер 1, общей стоимостью ab100 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
?                          ^^^


Expected :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
Actual   :Заказ номер 1, общей стоимостью -100 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.


- Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
?                          ^
+ Заказ номер 1, общей стоимостью -100 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
?                          ^^


Expected :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
Actual   :Заказ номер 1, общей стоимостью 200 и общим количеством ab2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.


- Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
+ Заказ номер 1, общей стоимостью 200 и общим количеством ab2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
?                                           ++


Expected :Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
Actual   :Заказ номер 1, общей стоимостью 200 и общим количеством -2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.


- Заказ номер 1, общей стоимостью 200 и общим количеством 2 на адрес "г. Москва, ул. Пушкина, 9, 12", курьером Сидоркин К.В.
+ Заказ номер 1, общей стоимостью 200 и общим количеством -2 на адрес "г. Москва, ул. Пушкина, 9, 12",
```

```
курьером Сидоркин К.В.
?                                       +




Ran 15 tests in 0.125s

FAILED (failures=10)
```