Homework 4v2: Тестирование реализации торгового автомата

- **Выполнил: Ланин Георгий Михайлович, БПИ202.**

- Библиотеки pytest и pytest-cov указаны в файле requirements.txt и уже сохранены в venv. Чтобы ими воспользоваться нужно перейти в проектный venv или можете в своей среде выполнить команду «pip install –r requirements.txt»

- Для запуска тестов в открытом в папке проекта терминале нужно написать pytest. Для запуска с отображением покрытия: « pytest --cov .», а чтобы информация сохранилась в html: «pytest --cov-report html --cov .».

- В тестовом модуле есть две pytest.fixture: operation_machine и administrator_machine. Они возвращают объект класс VendingMachine с определённым режимом работы.

- Файл index.html из папки htmlcov нужно открыть в браузере. Для ознакомления с покрытием конкретного файла нужно нажать на его название.

Оглавление

Исправления в коде

Исправление №1

Требование 1g: int getCoins2() возвращает количество внесенных монет 2 вида, вне режима отладки возвращает 0;

Код до исправления:

```
def getCoins2(self):
    if self.__mode == VendingMachine.Mode.OPERATION:
        return self.coins1
    return self.__coins2
```

Ошибка: return self.coins1
Исправление: Заменить «return self.coins1» на «return 0»

Код после исправления:

```
def getCoins2(self):
    if self.__mode == VendingMachine.Mode.OPERATION:
        return 0
    return self.__coins2
```

Исправление №2

Требование 1j: Response fillProducts() заполняет отделение с продуктами до максимального количества соответствующего продукта; функционирует только в режиме отладки; при запуске в некорректном режиме возвращает ILLEGAL_OPERATION, при корректном запуске возвращает OK;

Код до исправления:

```
def fillProducts(self):
    self.__num1 = self.__max2
    self.__num2 = self.__max2
    return VendingMachine.Response.OK
```

Ошибка: нет проверки текущего режима, вне режима отладки метод не должен присваивать максимальные значения, а должен вернуть ILLEGAL_OPERATION.

Код после исправления:

```
def fillProducts(self):
    if self.__mode == VendingMachine.Mode.ADMINISTERING:
        self.__num1 = self.__max2
        self.__num2 = self.__max2
        return VendingMachine.Response.OK
    else:
        return VendingMachine.Response.ILLEGAL_OPERATION
```

Ошибка 2: Числу товаров первого вида присваивается максимальное число второго товара, должно присваиваться максимальное число первого.

Код после исправления:

```
def fillProducts(self):
    if self.__mode == VendingMachine.Mode.ADMINISTERING:
        self.__num1 = self.__max1
        self.__num2 = self.__max2
        return VendingMachine.Response.OK
    else:
        return VendingMachine.Response.ILLEGAL_OPERATION
```

Исправление №3

Требование 1k: Response fillCoins(int c1, int c2) заполняет\опустошает отделение монет 1 вида до заданного значения c1, также заполняет\опустошает отделение монет 2 вида до заданного значения c2; функционирует только в режиме отладки; при запуске в некорректном режиме возвращает ILLEGAL_OPERATION, при корректном запуске возвращает OK; попытке указать c1 <=0 или больше максимума монет 1 или при попытке задать c2 <=0 или больше максимума монет 2 вида возвращает INVALID_PARAM;

Код до исправления:

```
def fillCoins(self, c1: int, c2: int):
    if self.__mode == VendingMachine.Mode.OPERATION:
        return VendingMachine.Response.ILLEGAL_OPERATION
    if c1 <= 0 or c2 > self.__maxc1:
        return VendingMachine.Response.INVALID_PARAM
    if c1 <= 0 or c2 > self.__maxc2:
        return VendingMachine.Response.INVALID_PARAM
    self.__coins1 = c1
    self.__coins2 = c2
    return VendingMachine.Response.OK
```

Ошибка 1: «if c1 <= 0 or c2 > self.__maxc1:» - с максимумом монет 1 нужно сравнивать c1, а не c2
Исправление 1: if c1 <= 0 or c1 > self.__maxc1:
Ошибка 2: «if c1 <= 0 or c2 > self.__maxc2:» - при проверке параметра для монет 2 проверять на положительность нужно c2, а не c1
Исправление 2: «if c2 <= 0 or c2 > self.__maxc2:»

Код после исправления:

```
def fillCoins(self, c1: int, c2: int):
    if self.__mode == VendingMachine.Mode.OPERATION:
        return VendingMachine.Response.ILLEGAL_OPERATION
    if c1 <= 0 or c1 > self.__maxc1:
        return VendingMachine.Response.INVALID_PARAM
    if c2 <= 0 or c2 > self.__maxc2:
        return VendingMachine.Response.INVALID_PARAM
    self.__coins1 = c1
    self.__coins2 = c2
    return VendingMachine.Response.OK
```

Исправление №4

Требование 1l: Response enterAdminMode(long code) переводит автомат в режим администрирования; в качестве параметра принимает секретный код, при несовпадении кода с эталоном возвращает INVALID_PARAM; при наличии внесенных покупателем средств перехода в режим отладки не происходит и возвращается CANNOT_PERFORM; при удачном выполнении переводит автомат в режим ADMINISTERING и возвращает OK;

Код до исправления:

```
def enterAdminMode(self, code: int):
    if self.__balance != 0:
        return VendingMachine.Response.UNSUITABLE_CHANGE
    if code != self.__id:
        return VendingMachine.Response.INVALID_PARAM
    self.__mode = VendingMachine.Mode.ADMINISTERING
    return VendingMachine.Response.OK
```

Ошибка:

```
if self.__balance != 0:
    return VendingMachine.Response.UNSUITABLE_CHANGE
```

В случае, если покупатель внёс средства, метод должен вернуть статус CANNOT_PERFORM, а данная реализация возвращает UNSUITABLE_CHANGE

Код после исправления:

```
def enterAdminMode(self, code: int):
    if self.__balance != 0:
        return VendingMachine.Response.CANNOT_PERFORM
    if code != self.__id:
        return VendingMachine.Response.INVALID_PARAM
    self.__mode = VendingMachine.Mode.ADMINISTERING
    return VendingMachine.Response.OK
```

Исправление №5

Требование 1n: Response setPrices(int p1, int p2) устанавливает цену 1 продукта; функционирует только в режиме отладки; при запуске в некорректном режиме возвращает ILLEGAL_OPERATION; при попытке установки значений цен меньше или равно 0 возвращает INVALID_PARAM; при запуске в корректных условиях устанавливает цену 1 продукта равной p1, цену 2 продукта равной p2 и возвращает OK;

Код до исправления:

```
def setPrices(self, p1: int, p2: int):
    if self.__mode == VendingMachine.Mode.OPERATION:
        return VendingMachine.Response.ILLEGAL_OPERATION
    self.__price1 = p1
    self.__price2 = p2
    return VendingMachine.Response.OK
```

Ошибка: Нет проверки параметров на положительность, если они <= 0 должно быть возвращено INVALID_PARAM.

Код после исправления:

```
def setPrices(self, p1: int, p2: int):
    if self.__mode == VendingMachine.Mode.OPERATION:
        return VendingMachine.Response.ILLEGAL_OPERATION
    if p1 <= 0 or p2 <= 0:
        return VendingMachine.Response.INVALID_PARAM
    self.__price1 = p1
    self.__price2 = p2
    return VendingMachine.Response.OK
```

Исправление №5

Требования 1o и 1p:

- Response putCoin1() добавляет монету 1 вида на счет пользователя; функционирует только в режиме OPERATION; при запуске в некорректном режиме возвращает ILLEGAL_OPERATION; при попытке внести монету, когда отделение 1 заполнено до максимума возвращает CANNOT_PERFORM; при запуске в корректных условиях увеличивает количество монет 1 вида на 1, баланс пользователя на стоимость 1 монеты и возвращает OK;
- Response putCoin2() добавляет монету 2 вида на счет пользователя; функционирует только в режиме OPERATION; при запуске в некорректном режиме возвращает ILLEGAL_OPERATION; при попытке внести монету, когда отделение 2 заполнено до максимума возвращает CANNOT_PERFORM; при запуске в корректных условиях увеличивает количество монет 2 вида на 1, баланс пользователя на стоимость 2 монеты и возвращает OK;

Код до исправления:

```
def putCoin2(self):
    if self.__mode == VendingMachine.Mode.ADMINISTERING:
        return VendingMachine.Response.ILLEGAL_OPERATION
    if self.__coins1 == self.__maxc1:
        return VendingMachine.Response.CANNOT_PERFORM
    self.__balance += self.__coinval1
    self.__coins1 += 1
    return VendingMachine.Response.OK

def putCoin1(self):
    if self.__mode == VendingMachine.Mode.ADMINISTERING:
        return VendingMachine.Response.ILLEGAL_OPERATION
    if self.__coins2 == self.__maxc2:
        return VendingMachine.Response.CANNOT_PERFORM
    self.__balance += self.__coinval2
    self.__coins2 += 1
    return VendingMachine.Response.OK
```

Ошибка: В методе putCoin1 проверяется и добавляется монета второго типа, а в методе putCoin2 — первого. Реализации нужно поменять местами.

Код после исправления:

```
def putCoin1(self):
    if self.__mode == VendingMachine.Mode.ADMINISTERING:
        return VendingMachine.Response.ILLEGAL_OPERATION
    if self.__coins1 == self.__maxc1:
        return VendingMachine.Response.CANNOT_PERFORM
    self.__balance += self.__coinval1
    self.__coins1 += 1
    return VendingMachine.Response.OK

def putCoin2(self):
    if self.__mode == VendingMachine.Mode.ADMINISTERING:
        return VendingMachine.Response.ILLEGAL_OPERATION
    if self.__coins2 == self.__maxc2:
        return VendingMachine.Response.CANNOT_PERFORM
    self.__balance += self.__coinval2
    self.__coins2 += 1
    return VendingMachine.Response.OK
```

Исправление №6

Требование 1q: Response returnMoney() возвращает монетами текущий внесенный баланс; функционирует только в режиме OPERATION; при запуске в некорректном режиме возвращает ILLEGAL_OPERATION; при запросе на возврат 0 баланса возвращает OK; при условии, что баланс невозможно вернуть, используя текущее количество монет, возвращает TOO_BIG_CHANGE; иначе, если баланс больше суммарной стоимости монет 2 вида, то выдаются все монеты 2 вида и разница выдается монетами 1 вида и возвращается OK; иначе, если баланс четный, то возвращается баланс/2 монет 2 вида и возвращается OK; иначе, если нет монет 1 вида, то возвращается UNSUITABLE_CHANGE; во всех иных случаях выдается баланс/2 монет 2 вида и 1 монета 1 вида и возвращается OK; во всех удачных завершениях (при возвращении OK) баланс устанавливается в 0.

Ошибка: В требовании указано, что баланс возвращается монетам 2 типа и одной монетой первого, поэтому уменьшаться на 1 должно число монет первого типа, а второго на баланс/2.

Код до исправления:
В методе returnMoney():

```
# using coinval1 == 1
self.__coins1 -= self.__balance // self.__coinval2
self.__coins2 -= 1
self.__balance = 0
return VendingMachine.Response.OK
```

Код после исправления:
В методе returnMoney():

```
# using coinval1 == 1
self.__coins2 -= self.__balance // self.__coinval2
self.__coins1 -= 1
self.__balance = 0
return VendingMachine.Response.OK
```

Исправление №7-8

Требование 1s: Response giveProduct2(int number) выдает предмет 2 вида; функционирует только в режиме OPERATION, при запуске в некорректном режиме возвращает ILLEGAL_OPERATION; при попытке получить <=0 предметов или больше максимума предметов 2 вида возвращает INVALID_PARAM; при попытке получить больше текущего количества предметов 2 вида возвращает INSUFFICIENT_PRODUCT; при отсутствии на счету требуемой суммы возвращается INSUFFICIENT_MONEY; если после выполнения операции в автомате недостаточно сдачи, то возвращается TOO_BIG_CHANGE; если на сдачу не хватает монет 2 вида, то выплачиваются все монеты 2 вида и остаток выдается монетами 1 вида, выдаются предметы 2 вида в выбранном количестве, возвращается OK; если сдача нацело делится на стоимость монеты 2 вида, то сдача выдается полностью монетами 2 вида, выдаются предметы 2 вида в выбранном количестве, возвращается OK; в ситуации, когда сдача нечетная, а монет 1 вида нет возвращается UNSUITABLE_CHANGE; в остальных случаях сдача выдается монетами 2 вида когда это возможно, затем — монетами 1 вида, Выдается выбранное количество предметов 2 вида, возвращается OK; во всех удачных случаях(при возвращении OK) баланс устанавливается в 0.

Код до исправления:

```
def giveProduct2(self, number: int):
    if self.__mode == VendingMachine.Mode.ADMINISTERING:
        return VendingMachine.Response.ILLEGAL_OPERATION
    if number <= 0 or number >= self.__max2:
        return VendingMachine.Response.INVALID_PARAM
    if number > self.__num2:
        return VendingMachine.Response.INSUFFICIENT_PRODUCT

    res = self.__balance - number * self.__price2
    if res < 0:
        return VendingMachine.Response.INSUFFICIENT_MONEY
    if res > self.__coins1 * self.__coinval1 + self.__coins2 * self.__coinval2:
        return VendingMachine.Response.INSUFFICIENT_MONEY
```

Ошибка 1: В случае нехватки сдачи данная реализация возвращает статус INSUFFICIENT_MONEY, а должна вернуть TOO_BIG_CHANGE.

Код после исправления:

```
def giveProduct1(self, number: int):
    if self.__mode == VendingMachine.Mode.ADMINISTERING:
        return VendingMachine.Response.ILLEGAL_OPERATION
    if number <= 0 or number >= self.__max1:
        return VendingMachine.Response.INVALID_PARAM
    if number > self.__num1:
        return VendingMachine.Response.INSUFFICIENT_PRODUCT
```

Ошибка 2: Метод должен сдать сдачу преимущественно монетами второго типа и добавить монету первого, однако в реализации число монет первого типа уменьшается на число необходимых монет второго типа, а второго только на 1. Переменные нужно поменять местами.

Код до исправления:

```
if self.__coins1 == 0:
    return VendingMachine.Response.UNSUITABLE_CHANGE
self.__coins1 -= res // self.__coinval2
self.__coins2 -= 1
self.__balance = 0
self.__num2 -= number
return VendingMachine.Response.OK
```

Код после исправления:

```
if self.__coins1 == 0:
    return VendingMachine.Response.UNSUITABLE_CHANGE
self.__coins2 -= res // self.__coinval2
self.__coins1 -= 1
self.__balance = 0
self.__num2 -= number
return VendingMachine.Response.OK
```

Покрытие кода тестами

Coverage report: 97%

coverage.py v6.5.0, created at 2022-10-10 22:55 +0300

Module	*statements*	*missing*	*excluded*	*coverage*
VendingMachine.py	184	9	0	95%
test_vending_machine.py	173	0	0	100%
Total	**357**	**9**	**0**	**97%**

coverage.py v6.5.0, created at 2022-10-10 22:55 +0300

Модуль VendingMachine не покрыт тестами на 100%, так как есть ситуации, которые нет возможности вызвать не нарушая принцип тестирования чёрного ящика, например, в методах returnMoney(), giveProduct1(), giveProduct2() есть проверка случая, когда баланс больше, чем сумма денег, которые есть в автомате. На практике, даже если изначально в автомате не было монет, человек, положив на счёт монеты, может или потратить часть денег, или не потратить их вовсе, то есть его баланс будет или равен внесённой сумме, или станет меньше её. Доступными методами увеличить баланс без соответствующего увеличения денежной суммы не удалось.