Тестирование - ДЗ4

Баханкова Дарья | БПИ215

Тестирование реализации торгового автомата

```
Примитивные функции (геттеры)
Корректные функции
Содержащие ошибки функции
Места, в которых содержалась ошибка
```

Требования к реализации

1. Реализация должна предоставлять следующие методы:

a. `int getNumberOfProduct1()`
   возвращает количество доступного 1 продукта;

b. `int getNumberOfProduct2()`
   возвращает количество доступного 2 продукта;

c. `int getCurrentBalance()`
   возвращает количество внесенных пользователем средств;

d. `Mode getCurrentMode()`
   возвращает текущий активный режим; поддерживаются два режима: OPERATION (рабочий режим) и ADMINISTERING (режим отладки);

e. `int getCurrentSum()`
   возвращает баланс монет доступный для выдачи, вне режима отладки возвращает 0;

f. `int getCoins1()`
   возвращает количество монет 1 вида в автомате,
   вне режима отладки возвращает 0;

g. `int getCoins2()`
   возвращает количество внесенных монет 2 вида,
   вне режима отладки возвращает 0;

h. `int getPrice1()`
   возвращает цену 1 продукта;

i. `int getPrice2()`
   возвращает цену 2 продукта;

j. `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;

k. `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;

l. `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;

m. `void exitAdminMode()`

переводит автомат в режим OPERATION;

n. `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;

o. `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;

p. `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;

q. `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.

r. `Response giveProduct1(int number)`

выдает предмет 1 вида; функционирует только в режиме OPERATION; при запуске в некорректном режиме возвращает ILLEGAL_OPERATION; при попытке получить <=0 предметов или больше максимума предметов 1 вида возвращает INVALID_PARAM; при попытке получить больше текущего количества предметов 1 вида возвращает INSUFFICIENT_PRODUCT; при отсутствии на счете 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 1 вида в выбранном количестве, возвращается OK; если сдача нацело делится на стоимость монеты 2 вида то сдача выдается полностью монетами 2 вида, выдаются предметы 1 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 1 вида, возвращается OK; во всех удачных случаях(при возвращении OK) баланс устанавливается в 0;

s. `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.

Задание

Разработать набор тестов для всех методов реализации банковского счета в классе `root.vending.VendingMachine`.

Набор тестов должен обеспечивать покрытие всех ветвлений в коде методов.

Замечание 1

Тестирование должно проводиться по принципу «серого ящика»:

- можно использовать имеющиеся константы
- нельзя использовать поля напрямую
- можно использовать все публичные методы

Замечание 2

Стоит стремиться к достижению максимально возможного покрытия.

Изменения в коде:

Изменения можно посмотреть в файле **CodeChanges_Bakhankova.pdf**