Задание

Разработать набор тестов по методу белого ящика для всех методов реализации банковского счета в классе `root.vending.VendingMachine`.
Набор тестов должен обеспечивать покрытие всех строк в коде методов.

Что сдавать?

Архив содержащий следующее:

1. Полный проект с тестами и необходимыми зависимостями (допускаются указанные в сборочном скрипте, но они должны присутствовать)
2. Файл с описанием ошибок в требованиях и коде (если они были обнаружены). Формат описания проблем приведен в Приложении 2.
3. HTML отчет о покрытии тестами кода реализации. При этом желательно разместить отчет полностью в подпапку рядом с содержимым проекта (например, coverage). Стоит помнить, что в отчет обычно входит не один файл, а целый набор связанных ресурсов, нужно добавить их все.

Что оценивается?

1. Наличие тестов для каждого из методов

2. Адекватность тестового набора

-разбиение на отдельные тесты

-построение тестов по принципу AAA

-соответствие тестов решаемым задачам

-при наличии проблем — наличие тестов на них, возвращающих вердикт за определенное время

3. Наличие сообщения о проблеме при наличии проблемы в коде (-1 балл за отсутствие наличия найденной проблемы если она была);

4. Наличие полного (с возможностью переходить по ссылкам, не только index.html) отчета о покрытии со 100 процентным покрытием по строкам

Приложение 1. Тестирование реализации торгового автомата

Тестируемый объект представляет собой торговый автомат который функционирует в одном из 2 режимов – операционном в котором можно осуществлять внесение наличности и покупки и режим администрирования в котором можно менять цены и пополнять запасы продукции.
Стандартный код для авторизации 117345294655382
Максимальное количество продукции — 40 единиц, стартовое — 0. Стоимость продукта — 5 условных единиц.
Принимаются монеты в количестве до 50 штук номиналом 1 условная единица и 2 условные единицы, стартовое их количество — 0 и 0 соответственно.
Общий баланс для покупок изначально равен 0 и пополняется при внесении монет в оперативном режиме.

Требования к реализации

1. Реализация должна предоставлять следующие методы:

a. `int getNumberOfProduct()`
   возвращает количество доступного продукта;

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

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

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

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

f. `int getCoins2()`
   возвращает количество монет 2 вида в автомате,
   вне режима отладки возвращает 0;

g. `int getPrice()`
   возвращает цену продукта;

h. `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;

i. `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 с2 <=0 или больше максимума монет 2 вида возвращает INVALID_PARAM;

j. `Response enterAdminMode(long code)`

вне зависимости от текущего режима, 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;

k. `void exitAdminMode()`

если автомат находился в режиме ADMINISTERING, то переводит автомат в режим OPERATION, иначе — оставляет режим прежним;

l. `Response setPrices(int p)`

устанавливает цену 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 продукта равной p и возвращает OK;

m. `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;

n. `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;

o. `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.

p. `Response giveProduct(int number)`

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

Замечание 1

Стоит стремиться к достижению максимально возможного покрытия.

Приложение 2. Формат представления сообщений о проблеме

Для каждой проблемы нужно привести следующие данные

1) код до исправления;

2) данные, на которых наблюдается некорректное поведение;

3) полученное значение, ожидаемое значение;

4) код после исправления.

Код до исправления и код после исправления может представлять из себя небольшой фрагмент (место с исправлением и до 1-2 строк вокруг конткеста если необходимо).