Домашнее задание 4.

Дудников Кирилл.

**Отчет об ошибках.**

Ссылка на этот же документ, но в Google Docs: <https://docs.google.com/document/d/1_sr7fBIYzX2n64l3gjD3Ko-Zyi8Fqj1HGLWf8dWkttE/edit?usp=sharing>

Формат:

1. код до исправления;
2. данные, на которых наблюдается некорректное поведение;
3. полученное значение, ожидаемое значение;
4. код после исправлений.

Все методы располагаются в классе root.vending.VendingMachine.

№ 1.

Метод fillProducts:

| **public** Response **fillProducts**()  {  num1 = max2;  num2 = max2;  **return** Response.OK;  } |
| --- |

Входные данные 1:

| vm.enterAdminMode(adminCode);  vm.fillProducts();  assertEquals(max1, vm.getNumberOfProduct1()); |
| --- |

Ожидаемое значение: 30

Полученное значение: 40

Входные данные 2:

| vm.exitAdminMode();  assertEquals(VendingMachine.Response.ILLEGAL\_OPERATION, vm.fillProducts()); |
| --- |

Ожидаемое значение: ILLEGAL\_OPERATION

Полученное значение: OK

Нарушает требование: 1.j

Код после исправлений:

| **public** Response **fillProducts**()  {  **if**(mode == Mode.OPERATION) **return** Response.ILLEGAL\_OPERATION;  num1 = max1;  num2 = max2;  **return** Response.OK;  } |
| --- |

№ 2.

Метод fillCoins:

| **public** Response **fillCoins**(**int** c1, **int** c2)  {  **if**(mode == Mode.OPERATION) **return** Response.ILLEGAL\_OPERATION;  **if**(c1 <= 0 || c2 > maxc1) **return** Response.INVALID\_PARAM;  **if**(c2 <= 0 || c2 > maxc2) **return** Response.INVALID\_PARAM;  coins1 = c1;  coins2 = c2;  **return** Response.OK;  } |
| --- |

Входные данные:

| vm.enterAdminMode(adminCode);  assertEquals(VendingMachine.Response.INVALID\_PARAM, vm.fillCoins(maxc1 + 1, maxc2)); |
| --- |

Ожидаемое значение: INVALID\_PARAM

Полученное значение: OK

Нарушает требование: 1.k

Код после исправлений:

| **public** Response **fillCoins**(**int** c1, **int** c2)  {  **if**(mode == Mode.OPERATION) **return** Response.ILLEGAL\_OPERATION;  **if**(c1 <= 0 || c1 > maxc1) **return** Response.INVALID\_PARAM;  **if**(c2 <= 0 || c2 > maxc2) **return** Response.INVALID\_PARAM;  coins1 = c1;  coins2 = c2;  **return** Response.OK;  } |
| --- |

№ 3.

Метод enterAdminMode:

| **public** Response **enterAdminMode**(**long** code)  {  **if**(balance != 0) **return** Response.UNSUITABLE\_CHANGE;  **if**(code != id) **return** Response.INVALID\_PARAM;   mode = Mode.ADMINISTERING;  **return** Response.OK;  } |
| --- |

Входные данные:

| vm.putCoin1();  assertTrue(vm.getCurrentBalance() != 0);  assertEquals(VendingMachine.Response.CANNOT\_PERFORM, vm.enterAdminMode(adminCode)); |
| --- |

Ожидаемое значение: CANNOT\_PERFORM

Полученное значение: UNSUITABLE\_CHANGE

Нарушает требование: 1.l

Код после исправлений:

| **public** Response **enterAdminMode**(**long** code)  {  **if**(balance != 0) **return** Response.CANNOT\_PERFORM;  **if**(code != id) **return** Response.INVALID\_PARAM;   mode = Mode.ADMINISTERING;  **return** Response.OK;  } |
| --- |

№ 4.

Метод setPrices:

| **public** Response **setPrices**(**int** p1, **int** p2)  {  **if**(mode == Mode.OPERATION) **return** Response.ILLEGAL\_OPERATION;  **if**(p1 <= 0 || p1 <= 0) **return** Response.INVALID\_PARAM;  price1 = p1;  price2 = p2;  **return** Response.OK;  } |
| --- |

Входные данные:

| vm.enterAdminMode(adminCode);  assertEquals(VendingMachine.Response.INVALID\_PARAM, vm.setPrices(1, 0)); |
| --- |

Ожидаемое значение: INVALID\_PARAM

Полученное значение: OK

Нарушает требование: 1.n

Код после исправлений:

| **public** Response **setPrices**(**int** p1, **int** p2)  {  **if**(mode == Mode.OPERATION) **return** Response.ILLEGAL\_OPERATION;  **if**(p1 <= 0 || p2 <= 0) **return** Response.INVALID\_PARAM;  price1 = p1;  price2 = p2;  **return** Response.OK;  } |
| --- |

№ 5.

Метод putCoin1:

| **public** Response **putCoin1**()  {  **if**(mode == Mode.ADMINISTERING) **return** Response.ILLEGAL\_OPERATION;  **if**(coins2 == maxc2) **return** Response.CANNOT\_PERFORM;   balance += coinval2;  coins2++;   **return** Response.OK;  } |
| --- |

Входные данные:

| vm.exitAdminMode();  assertEquals(VendingMachine.Response.OK, vm.putCoin1());  assertEquals(1, vm.getCurrentBalance()); |
| --- |

Ожидаемое значение: 1

Полученное значение: 2

Нарушает требование: 1.o

Код после исправлений:

| **public** Response **putCoin1**()  {  **if**(mode == Mode.ADMINISTERING) **return** Response.ILLEGAL\_OPERATION;  **if**(coins1 == maxc1) **return** Response.CANNOT\_PERFORM;   balance += coinval1;  coins1++;   **return** Response.OK;  } |
| --- |

№ 6.

Метод putCoin2:

| **public** Response **putCoin2**()  {  **if**(mode == Mode.ADMINISTERING) **return** Response.ILLEGAL\_OPERATION;  **if**(coins1 == maxc1) **return** Response.CANNOT\_PERFORM;   balance += coinval1;  coins1++;   **return** Response.OK;  } |
| --- |

Входные данные:

| vm.exitAdminMode();  assertEquals(VendingMachine.Response.OK, vm.putCoin2());  assertEquals(2, vm.getCurrentBalance()); |
| --- |

Ожидаемое значение: 2

Полученное значение: 1

Нарушает требование: 1.p

Код после исправлений:

| **public** Response **putCoin2**()  {  **if**(mode == Mode.ADMINISTERING) **return** Response.ILLEGAL\_OPERATION;  **if**(coins2 == maxc2) **return** Response.CANNOT\_PERFORM;   balance += coinval2;  coins2++;   **return** Response.OK;  } |
| --- |

№ 7.

Метод returnMoney:

| **public** Response **returnMoney**()  {  **if**(mode == Mode.ADMINISTERING) **return** Response.ILLEGAL\_OPERATION;   **if**(balance == 0)  {  **return** Response.OK;  }  **else** **if**(balance > coins1\*coinval1 + coins2\*coinval2)  {  **return** Response.TOO\_BIG\_CHANGE;  }  **else** **if**(balance > coins2\*coinval2)  {  *// using coinval1 == 1*  coins1 -= (balance-coins2\*coinval2);  coins2 = 0;  balance = 0;   **return** Response.OK;  }  **else** **if**(balance%coinval2 == 0)  {  coins2 -= (balance/coinval2);  balance = 0;   **return** Response.OK;  }  **else** **if**(coins1 == 0)  {  *// using coinval1 == 1*  **return** Response.UNSUITABLE\_CHANGE;  }  **else**  {  *// using coinval1 == 1*  coins1 -= (balance/coinval2);  coins2--;  balance = 0;   **return** Response.OK;  }  } |
| --- |

Входные данные:

| vm.enterAdminMode(adminCode);  assertEquals(VendingMachine.Response.OK, vm.fillCoins(7, 7));  vm.exitAdminMode();  IntStream.rangeClosed(1, 9).forEach(j -> vm.putCoin1());  vm.returnMoney();  vm.enterAdminMode(adminCode);  assertEquals(3, vm.getCoins2()); |
| --- |

Ожидаемое значение: 3

Полученное значение: 6

Нарушает требование: 1.q

Код после исправлений:

| **public** Response **returnMoney**()  {  **if**(mode == Mode.ADMINISTERING) **return** Response.ILLEGAL\_OPERATION;   **if**(balance == 0)  {  **return** Response.OK;  }  **else** **if**(balance > coins1\*coinval1 + coins2\*coinval2)  {  **return** Response.TOO\_BIG\_CHANGE;  }  **else** **if**(balance > coins2\*coinval2)  {  *// using coinval1 == 1*  coins1 -= (balance-coins2\*coinval2);  coins2 = 0;  balance = 0;   **return** Response.OK;  }  **else** **if**(balance%coinval2 == 0)  {  coins2 -= (balance/coinval2);  balance = 0;   **return** Response.OK;  }  **else** **if**(coins1 == 0)  {  *// using coinval1 == 1*  **return** Response.UNSUITABLE\_CHANGE;  }  **else**  {  *// using coinval1 == 1*  coins2 -= (balance/coinval2);  coins1--;  balance = 0;   **return** Response.OK;  }  } |
| --- |

№ 8.

Метод getCoins2:

| **public** **int** **getCoins2**()  {  **if**(mode == Mode.OPERATION)  **return** coins1;  **else**  **return** coins2;  } |
| --- |

Входные данные:

| vm.enterAdminMode(adminCode);  vm.fillCoins(23, 14);  vm.exitAdminMode();  assertEquals(0, vm.getCoins2()); |
| --- |

Ожидаемое значение: 0

Полученное значение: 23

Нарушает требование: 1.g

Код после исправлений:

| **public** **int** **getCoins2**()  {  **if**(mode == Mode.OPERATION)  **return** 0;  **else**  **return** coins2;  } |
| --- |

№ 9.

Метод giveProduct2:

| **public** Response **giveProduct2**(**int** number)  {  **if**(mode == Mode.ADMINISTERING) **return** Response.ILLEGAL\_OPERATION;   **if**(number <= 0 || number > max2) **return** Response.INVALID\_PARAM;  **if**(number > num2) **return** Response.INSUFFICIENT\_PRODUCT;   **int** res = balance - number\*price2;   **if**(res < 0) **return** Response.INSUFFICIENT\_MONEY;  **else** **if**(res > coins1\*coinval1 + coins2\*coinval2)  {  **return** Response.INSUFFICIENT\_MONEY;  }  **else** **if**(res > coins2\*coinval2)  {  *// using coinval1 == 1*  coins1 -= (res-coins2\*coinval2);  coins2 = 0;  balance = 0;  num2 -= number;   **return** Response.OK;  }  **else** **if**(res%coinval2 == 0)  {  coins2 -= (res/coinval2);  balance = 0;  num2 -= number;   **return** Response.OK;  }  **else** **if**(coins1 == 0)  {  *// using coinval1 == 1*  **return** Response.UNSUITABLE\_CHANGE;  }  **else**  {  *// using coinval1 == 1*  coins1 -= (res/coinval2);  coins2--;  balance = 0;  num2 -= number;   **return** Response.OK;  }  } |
| --- |

Входные данные:

| vm.enterAdminMode(adminCode);  vm.fillProducts();  vm.fillCoins(1, 5);  vm.setPrices(1, 4);  vm.exitAdminMode();  IntStream.rangeClosed(1, 4 + 5).forEach(i -> vm.putCoin1());  vm.giveProduct1(1);  vm.enterAdminMode(adminCode);  assertEquals(1 + 9 - 1, vm.getCoins1()); |
| --- |

Ожидаемое значение: 9

Полученное значение: 10

Нарушает требование: 1.s

Код после исправлений:

| **public** Response **giveProduct2**(**int** number)  {  **if**(mode == Mode.ADMINISTERING) **return** Response.ILLEGAL\_OPERATION;   **if**(number <= 0 || number > max2) **return** Response.INVALID\_PARAM;  **if**(number > num2) **return** Response.INSUFFICIENT\_PRODUCT;   **int** res = balance - number\*price2;   **if**(res < 0) **return** Response.INSUFFICIENT\_MONEY;  **else** **if**(res > coins1\*coinval1 + coins2\*coinval2)  {  **return** Response.TOO\_BIG\_CHANGE;  }  **else** **if**(res > coins2\*coinval2)  {  *// using coinval1 == 1*  coins1 -= (res-coins2\*coinval2);  coins2 = 0;  balance = 0;  num2 -= number;   **return** Response.OK;  }  **else** **if**(res%coinval2 == 0)  {  coins2 -= (res/coinval2);  balance = 0;  num2 -= number;   **return** Response.OK;  }  **else** **if**(coins1 == 0)  {  *// using coinval1 == 1*  **return** Response.UNSUITABLE\_CHANGE;  }  **else**  {  *// using coinval1 == 1*  coins2 -= (res/coinval2);  coins1--;  balance = 0;  num2 -= number;   **return** Response.OK;  }  } |
| --- |

**Дополнение.**

В связи с тем, что некоторые требования из ТЗ недостижимы (по крайней мере, методом “черного ящика”), о чем я ранее писал Вам по почте, не удается формально описать одну ошибку.

Эта ошибка: в файле VendingMachine.java в методе giveProduct2:

строки 274-277:

else if(res > coins1\*coinval1 + coins2\*coinval2)

{

return Response.INSUFFICIENT\_MONEY;

}

Строку “return Response.INSUFFICIENT\_MONEY;”   
нужно заменить на “return Response.TOO\_BIG\_CHANGE;”