Assessment 2 Testing Documentation - Defects

DEFECT\_1

Defect Description

VendItem(s) were not being sorted in the stock array, causing them to be ordered by ID like 1,2,3,5,4. This made searching for stock quite inefficient and the stock would not be ordered properly when displayed.

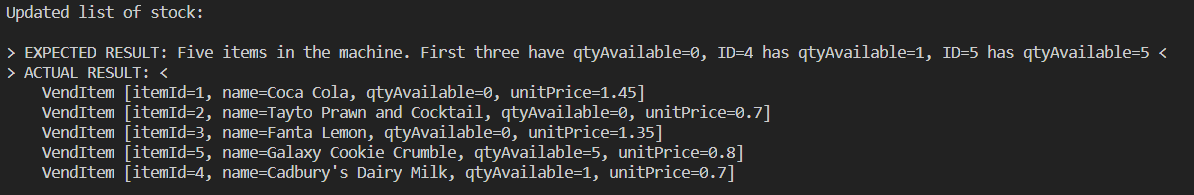
Defect Reason

VendingMachine had no algorithm in place to sort stock.

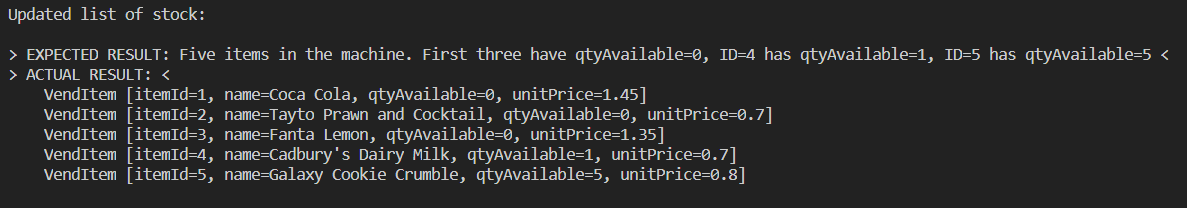
Defect Solution

Implemented a bubble sort algorithm sortStock() which sorted the stock based on Item ID.

Defect Screenshot



Fixed Screenshot



DEFECT\_2

Defect Description

Attempting to purchase an item which does not exist threw a NullPointerException. This exception was thrown by the VendingMachine findItem() method.

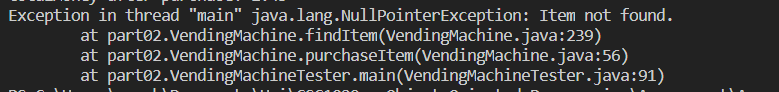
Defect Reason

NullPointerException was thrown by the findItem() method, but was not caught by the VendingMachine purchaseItem() method.

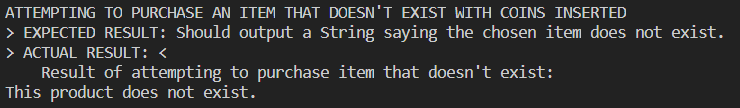
Defect Solution

Updated the purchaseItem() method to catch and manage the NullPointerException thrown by the findItem() method.

Defect Screenshot



Fixed Screenshot



DEFECT\_3

Defect Description

Defect was an infinite loop caused by a floating point precision error. I didn't realise this happening originally and had a CPU core running at 100% usage for a bit before I did. The error happened at REFERENCE\_1 in the VendingMachineTester code, where I input 0.80 altogether in userMoney.

Defect Reason

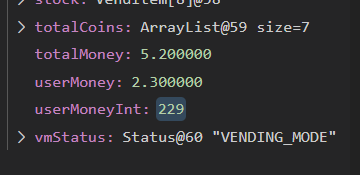
Although I had mostly changed to using integer for all money calculations, the 0.80 in userMoney was actually being stored as 0.799999999999..., meaning when I \*100 and converted it to an integer, it would be 79, and the while loop in the chooseReturnCoins() method would never terminate.

Defect Solution

I ensured that userMoney was rounded to the next 10 using Math.round(float) AFTER it was multiplied by 100. So for example, 0.79999999 \* 100 would be 79 when casted to an integer, then Math.round() brought it to 80.

Defect Screenshot

NOTE: The screenshot is of the same defect as described, but just a different amount of userMoney. Also note, the debugging utility in the IDE I use automatically rounds up/down floating point values so it claims userMoney=2.3000000 when it really isn't, but the problem can be seen in userMoneyInt after I performed userMoneyInt = (int)(userMoney\*100).



Fixed Screenshot



DEFECT\_4

Defect Description

The deliver() method was instead just returning null, rather than the String stated in UNIT\_TEST\_13.

Defect Reason

I hadn't implemented the String stated in the test.

Defect Solution

Instead of returning null when deliver() is called on a VendItem with qtyAvailable=0, instead return a String that says it's out of stock.

DEFECT\_5

Defect Description

In the menu, if the user enters an invalid number for their menu choice, they are returned to where they can make a choice as they should be, but the system doesn’t inform them of their error.

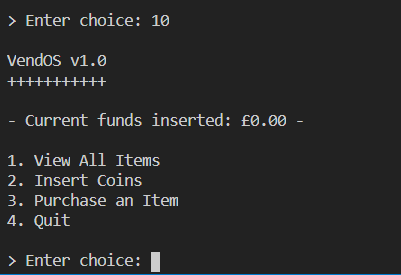
Defect Reason

The message was not implemented.

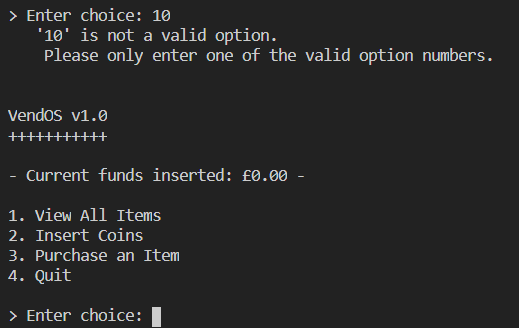
Defect Solution

Implement the message in the switch case of the VendingMachineApp.processChoice() method.

Defect Screenshot



Fixed Screenshot



DEFECT\_6

Defect Description

After implementing the solution to DEFECT\_5, the system displayed the implemented error message when the user choice option 4(Quit). The system would still quit as expected, just displayed said message before it did so.

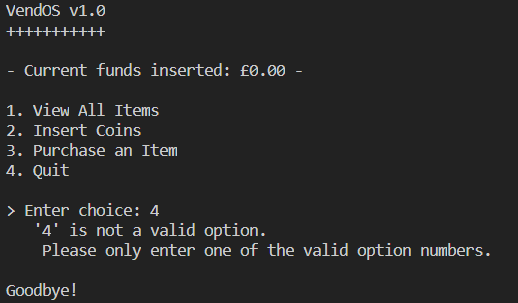
Defect Reason

The switch case in processChoice() was ignoring option 4 as I had assumed it would just terminate with the while loop, but it instead runs processChoice() before terminating, therefore processChoice(4) would result in the default case, which outputs a message saying the choice was invalid.

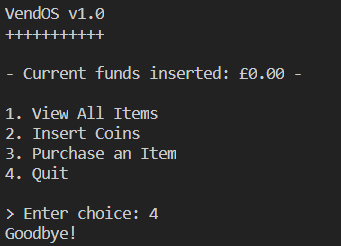
Defect Solution

Originally I decided to update the processChoice() method to catch option 4 to ensure it is definitely caught and the system quits WITHOUT showing the 'is not a valid option' output.  
  
I instead decided that it would be best to ONLY run the processChoice() method IF choice != options.length. This allows for more expandability and future proofing if more menu options were to be added to the menu in future.

Defect Screenshot



Fixed Screenshot



DEFECT\_7

Defect Description

With the machine containing no VendItem(s), the listAll() method would just output a blank area, rather than a String informing the user of the machine containing no stock.

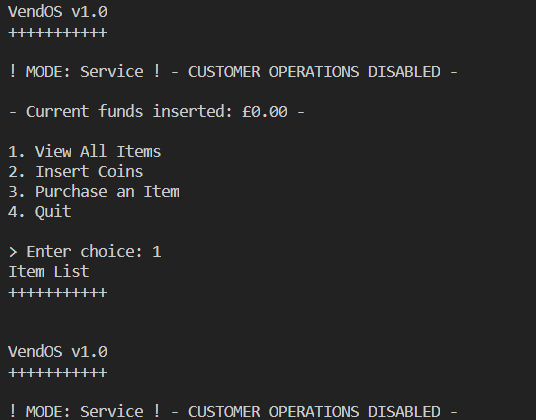
Defect Reason

The message was not implemented.

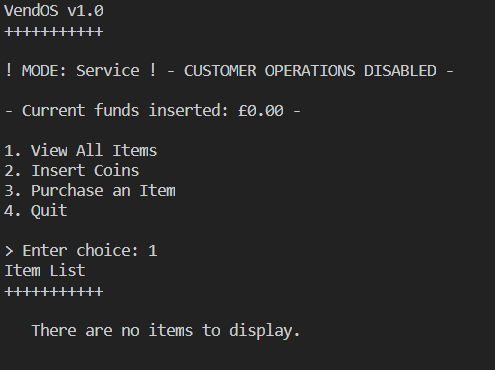
Defect Solution

Implement the message to inform the user that no items(VendItem(s)) have been added to the machine if none have been added.

Defect Screenshot



Fixed Screenshot



DEFECT\_8

Defect Description

When asked to input an integer and the user just presses enter, they can keep pressing enter constantly, adding a new line to the input. Any output that was previously displayed will eventually disappear if the user keeps pressing enter. Not really a huge problem but can mean that any informative text will hide if the user presses enter enough times.

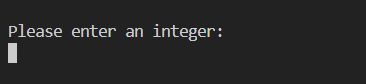
Defect Reason

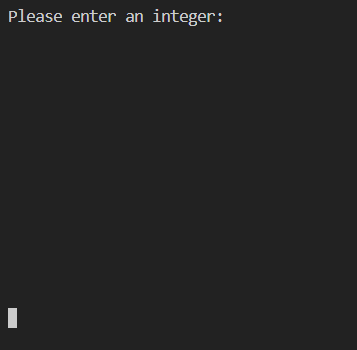
A scanner's nextInt() method entirely ignores whitespace and just waits until an integer is input. The user could press enter as much as they want and new output will only be written when the user does enter an integer or an invalid input such as an actual String.

Defect Solution

Read the input as a String as Strings count whitespace, then try to convert it to an integer.

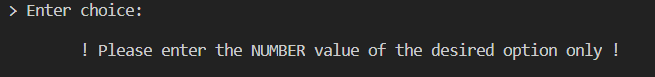
Defect Screenshot





Fixed Screenshot

NOTE: Fixed screenshot is from VendingMachineApp, and enter was pressed rather than choosing an option.



DEFECT\_9

Defect Description

Same problem as DEFECT\_8 but with double input.

Defect Reason

Same problem as DEFECT\_8 but with double input.

Defect Solution

Read the input as a String as Strings count whitespace, then try to convert it to a double.

Defect Screenshot

See DEFECT\_8.

Fixed Screenshot

See DEFECT\_8.

DEFECT\_10

When attempting to add the last item in the machine, the system output saying no more items can be added.

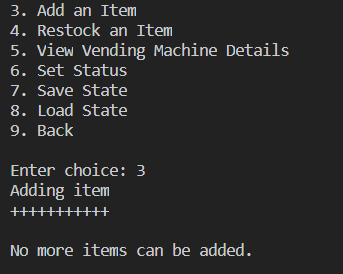
Defect Reason

Was simply an off by one error when checking if the length of the stock array has been met.

Defect Solution

Fix the off by one error.

Defect Screenshot



Fixed Screenshot

