

## Overview

- I tested the Java application with JUnit 5 and got 81% coverage of the source code. I wrote 24 tests and documented the bugs that I came across. The failing tests are highlighted in red

## Challenges

- Mocking system input in Java was difficult for me. I ended up calling `System.setIn()` in the `@BeforeEach` clause or in the test case itself

## Test Directory (Java/Test)

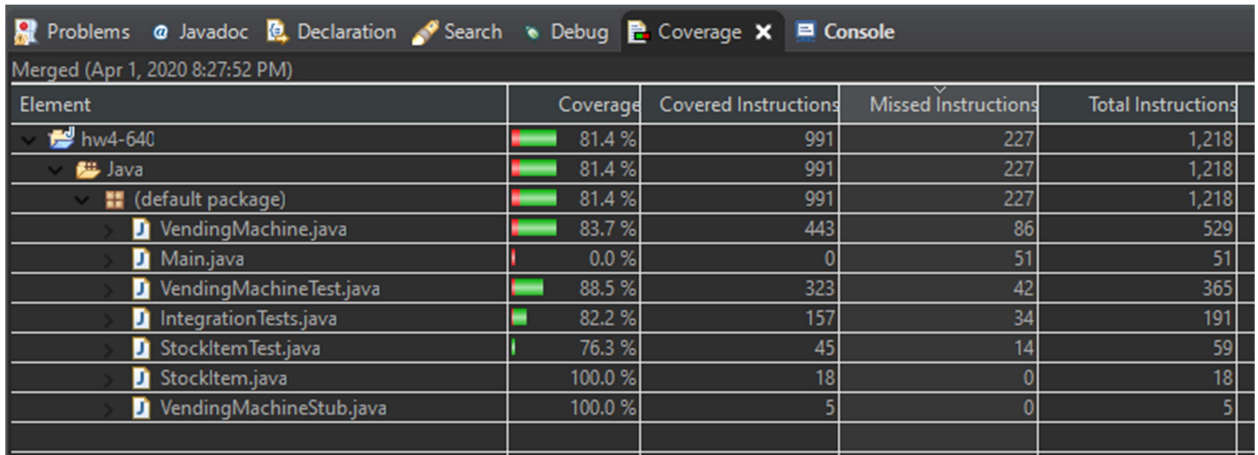









- Test Files
  - `Integration_Tests.java`
  - `VendingMachine_Tests.java`
  - `StockItem_Tests.java`
- Stub File
  - `VendingMachine_Stub.java`
    - Inherits from `VendingMachine`
    - Overrides `getFileName()` with filepath string literal
- (source files are copied over too)

## Jar Directory (Java/Test/Jar)

## Bugs

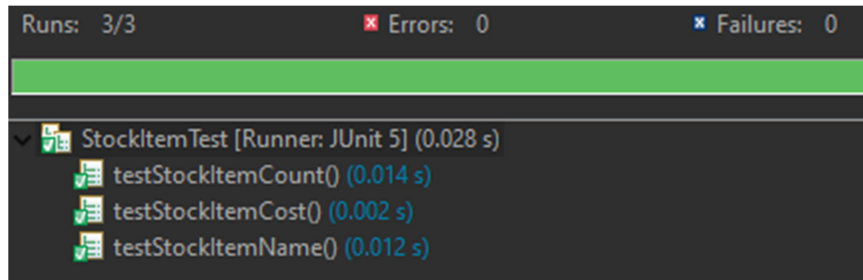
1. When the machine says "Press (e)xit, (r)estock, or anything else to continue:" ; an exception is raised if the user only clicks enter
2. When the user makes a selection, the machine does not check to see if the user has inserted enough money to make this selection. Therefore, a user may purchase an item without paying for it
3. When the user inserts a coin value (such as .03) that does not match an expected value (0.5,.10,.25,1), the system does not enforce this expectation, and adds .03 to the balance
4. When an item is purchased, and the new machine balance still exceeds the cost, the change is not dispensed

## Coverage (JUnit 5 test runner – Eclipse IDE)

Problems Javadoc Declaration Search Debug Coverage X Console				
Merged (Apr 1, 2020 8:27:52 PM)				
Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
hw4-640	 81.4 %	991	227	1,218
Java	 81.4 %	991	227	1,218
(default package)	 81.4 %	991	227	1,218
VendingMachine.java	 83.7 %	443	86	529
Main.java	 0.0 %	0	51	51
VendingMachineTest.java	 88.5 %	323	42	365
IntegrationTests.java	 82.2 %	157	34	191
StockItemTest.java	 76.3 %	45	14	59
StockItem.java	 100.0 %	18	0	18
VendingMachineStub.java	 100.0 %	5	0	5

## Test Cases

### StockItem\_Test.java

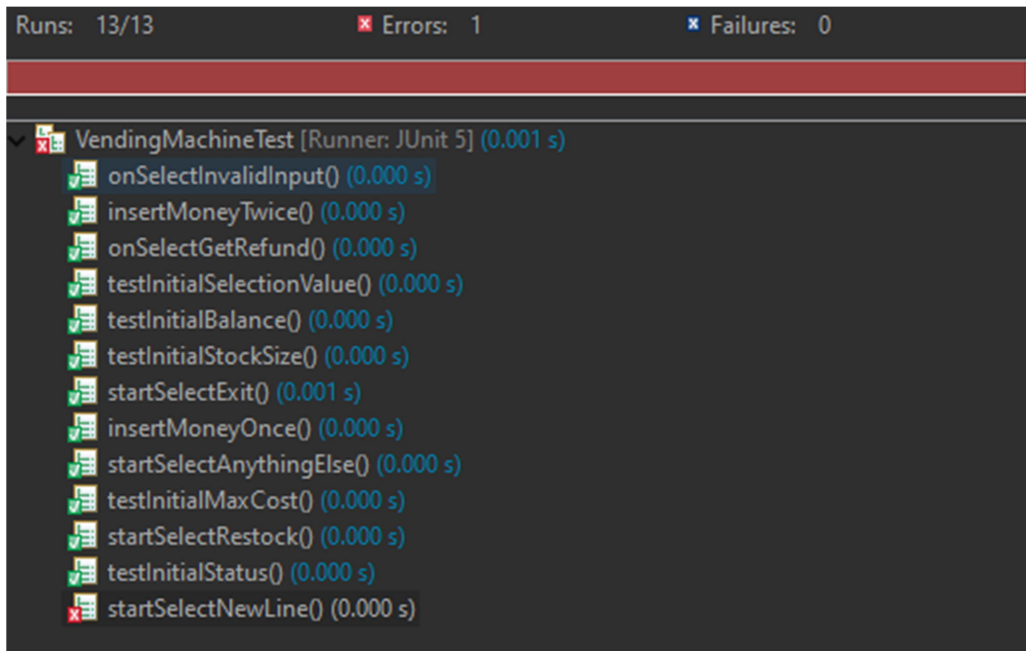


testStockItemName() ~ asserts that the "name" attribute is publicly accessible

testStockItemCost() ~ asserts that the "cost" attribute is publicly accessible

testStockItemCount() ~ asserts that the "count" attribute is publicly accessible

## VendingMachine\_Test.java



testInitialBalance() ~ asserts that balance is 0 initially

testInitialStockSize() ~ asserts that machine has 5 initial stock items

testInitialStatus() ~ asserts that initial status is "START"

testInitialMaxCost() ~ asserts that the initial value of max cost is positive

testInitialSelectionValue() ~ asserts that initial selection value is zero

onSelectGetRefund() ~ asserts that selecting "r" (refund) sets status to "CHANGE"

onSelectInvalidInput() ~ asserts that invalid input sets selection to -1

insertMoneyOnce() ~ asserts that inserting money increases balance

insertMoneyTwice() ~ asserts that inserting money twice increases balance twice

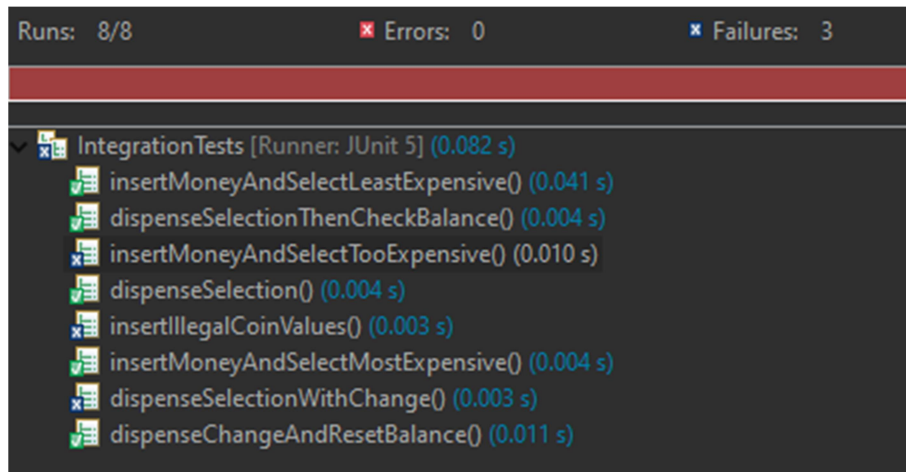
startSelectRestock() ~ asserts that inputting "r" (restock) at start sets the status to "STOCK"

startSelectExit() ~ asserts that "e" (exit) sets the state to "OFF"

startSelectAnythingElse() ~ asserts that by default, status is set to "INSERT"

startSelectNewLine() ~ asserts that the new line character sets the status to "INSERT" (default)

## Integration\_Tests.java



insertMoneyAndSelectMostExpensive() ~ asserts that the most expensive drink is selected and the status is "DISPENSE"

insertMoneyAndSelectLeastExpensive() ~ asserts that the cheapest drink is selected and the status is "DISPENSE"

dispenseSelectionThenCheckBalance() ~ asserts that balance is 0 after selection is dispensed

dispenseSelection() ~ asserts that when item is dispensed, its count attribute is decremented

dispenseChangeAndResetBalance() ~ asserts that dispenseChange clears balance and sets status to "START"

dispenseSelectionWithChange() ~ asserts that after selection is dispensed, the status is "CHANGE"

insertMoneyAndSelectTooExpensive() ~ asserts that selection is set to -1 if funds are insufficient

insertIllegalCoinValues() ~ asserts that illegal coin values are interpreted as errors