



Shaun Vulaj
SER 316
Assignment# 2
02/09/2022

Task 2

5 things that you think need changing to improve the code:

- **BearWorkshop class : addbear() & removebear() :**
 - This is a style format, the brackets in this method does not follow the formatting of the rest of the system.
- **BearWorkshop class : Checkout() :**
 - Temporary variables are created randomly throughout the method and not at the top of the method. This makes the method difficult to read.
- **BearWorkshop class : getRawCost() ->**
 - I think that the bearprice variable should be moved to the top of the method and all of the calculations should be done in this temporary variable instead of the bear.price variable.
- **NoiseMaker class :**
 - Enums should follow the format style of making declarations at the top of the class.
- **NoiseMaker class : Constructor() :**
 - I believe that in the switch statement should offer the option for all locations since all customers won't just opt for the small selections currently offered and that are implemented but never used.
- **BearWorkshop class : getCost() :**
 - There should be a temporary variable here to avoid any possible headaches in the system. Currently the method keeps changing the "price" of the bear when we could more safely perform these calculations through a temporary viable.
- **BearWorkshop class : getRawCost() :**
 - Price is set to zero right at the end of the method, this doesn't seem to accomplish anything but setting the bears price to zero, which doesn't make much sense in this system.

Task 3

Test cases:

Does the cart support x number of bears in the cart?

Test cases
0-5 bears
5+ bears
10 bears

Are clothes buy 2 get one free?

Test case
3 clothes with varying prices(Expensive to cheapest)
6 clothes all same price
9 clothes all same price

Do bears accept 10+ accessories?

Test case
1 bear: 0 accessories
1 bear: 0-5 accessories
1 bear: 5-9 accessories
1 bear: 10 accessories

Does the ink become free if bear exceeds \$70 value?

Test case
Is free – above \$70 (single item)
Is free – at \$70 (single item)
Is free – at \$70 (varying items)
Is free – above \$70 (varying items)
Is free – below \$70.

Task 3

List in detail which errors you found and tell me which test case(s) helped you fine tune it (eg. implementation X does not use a discount if there are more than X bears in the workshop/cart).

Test #	What happened	Class that had error or revealed there was an Easter egg
Test A	Tested: 0-5 bears in cart. Tested 4 bears. No errors in any class	Errors in classes: NA Easter Eggs:
Test B	Tested: 5+ bears in cart. Tested 7 bears. No errors in any class	Errors in classes: NA Easter Eggs:
Test C	<p>Tested: 10 bears in cart. Tested 10 bears. This test was a boundary test and seems to point to an issue where the cart cannot exceed 10 bears. Was re-tested at 9 bears and broke at 9 as well.</p> <p>Conclusion: The cart cannot hold 9 or more bears at a given time.</p> <p><u>Output:</u></p> <pre> testcase name="givenDiscount" classname="TestC" status="skipped" time="0.003" hauns-MacBook-Air.local" time="0.003"> <properties/> <testcase name="oneBearTest3clothings[0]" classname="GivenBlackBox" time="0.03"/> <testcase name="threeBearsSaveOnCheapest[0]" classname="GivenBlackBox" time="0.0"/> <testcase name="testC[0]" classname="GivenBlackBox" time="0.022"> <failure message="java.lang.AssertionError: expected:<123.0>; but was:<164.0>;" type="java.lang.AssertionError"> java.lang.AssertionError: expected:<123.0>; but was:<164.0>; </pre>	Errors in classes: 0 Easter Eggs: #4

Test D	<p><u>Tested:</u> 3 clothes with varying prices(Expensive to cheapest). This test was a boundary test to see if the standard discount of buy 2 get 1 free actually works.</p> <p><u>Conclusion:</u> It appears that the discount of “free” was given to the first article of clothing in the clothing list for a single bear.</p> <p><u>Output:</u></p> <pre> shauns-MacBook-Air.local" time="0.096"> <properties/> <testcase name="oneBearTest3clothings[0]" classname="GivenBlackBox" time="0.045"/> <testcase name="threeBearsSaveOnCheapest[0]" classname="GivenBlackBox" time="0.0"/> <testcase name="testD[0]" classname="GivenBlackBox" time="0.024"> <failure message="java.lang.AssertionError: expected:<2.0>gt; but was:<6.0>gt;" type="java.lang.AssertionError"> java.lang.AssertionError: expected:<2.0>gt; but was:<6.0>gt; </pre>	<p>Errors in classes:</p> <p>0 & 2</p> <p>Easter Eggs: #2</p>
Test E	<p><u>Tested:</u> 6 clothes all same price. When tested at 6 clothes on 1 bear, Easter eggs 7 & 8 were flagged.</p> <p><u>Conclusion:</u> Coming from the previous test all clothing prices were made the same. The two issues here I believe are from an error being generated from every piece of clothing being the same price(so system doesn't know how to pick which ones get discounted). The second is most likely from the bear exceeding \$70 which is specification detail given to us by the publisher.</p> <p><u>Output:</u></p> <pre> <testcase name="testE[4]" classname="GivenBlackBox" time="0.001"/> <testcase name="oneBearNoSavings[4]" classname="GivenBlackBox" time= <testcase name="testOfNoisesInABears_NoiseMakerCollection[4]" classr <system-out><![CDATA[Easter Egg #7: — Quote Cory House Easter Egg #7: — Quote Cory House Easter Egg #8: Thanks for playing, this is the last easter egg :-) Easter Egg #8: Thanks for playing, this is the last easter egg :-) </pre>	<p>Errors in classes: NA</p> <p>Easter Eggs: 7 & 8</p>
Test F	<p><u>Tested:</u> 9 clothes all same price. When tested at 9 clothes on 1 bear, Easter eggs 7 & 8 were flagged.</p> <p><u>Conclusion:</u> This test furthers my assumptions from Test E.</p> <p><u>Output:</u></p> <pre> <testcase name="testF[4]" classname="GivenBlackBox" time="0.001"/> <testcase name="oneBearNoSavings[4]" classname="GivenBlackBox" time="0.0"/> <testcase name="testOfNoisesInABears_NoiseMakerCollection[4]" classname="GivenBlackBox" time="0.001"/> <system-out><![CDATA[Easter Egg #7: — Quote Cory House Easter Egg #7: — Quote Cory House Easter Egg #8: Thanks for playing, this is the last easter egg :-) Easter Egg #8: Thanks for playing, this is the last easter egg :-)]]></system-out> </pre>	<p>Errors in classes: NA</p> <p>Easter Eggs: 7 & 8</p>
Test G	<p><u>Tested:</u> 0 accessories on one bear. There were no errors as expected.</p> <p><u>Conclusion:</u> Did not expect to find anything here but figured I should still test in case of possible root issues.</p> <p><u>Output:</u></p> <pre> <testcase name="testG[4]" classname="GivenBlackBox" time="0.0"/> <testcase name="oneBearNoSavings[4]" classname="GivenBlackBox" time="0.0"/> <testcase name="testOfNoisesInABears_NoiseMakerCollection[4]" classname="GivenBlackBox" </pre>	<p>Errors in classes: NA</p> <p>Easter Eggs:</p>
Test H	<p><u>Tested:</u> 1 bear: 0-5 accessories. Easter egg #3 was found. Tested 5 bears with all NoiseMaker accessories.</p>	<p>Errors in classes: 3</p>

	<p><u>Conclusion:</u> Easter egg #3 seems to pointing to an issue where the first accessory (in this case NoiseMakers) was not discounted but the other 3 accessories were resulting in a \$40 discount.</p> <p><u>Output:</u></p> <pre><testcase name="testH[3]" classname="GivenBlackBox" time="0.006"> <failure message="java.lang.AssertionError: expected:<0.0>; but java.lang.AssertionError: expected:<0.0>; but was:<40.0>;</pre>	<p>Easter Eggs:</p> <p>3</p>
Test I	<p><u>Tested:</u> 1 bear: 5-9 accessories. Tested 8 bears with all NoiseMaker accessories.</p> <p><u>Conclusion:</u> No new errors.</p> <p><u>Output:</u></p> <pre><testcase name="testI[3]" classname="GivenBlackBox" time="0.005"> <failure message="java.lang.AssertionError: expected:<0.0>; but java.lang.AssertionError: expected:<0.0>; but was:<70.0>;</pre>	<p>Errors in classes: 3</p> <p>Easter Eggs:</p>
Test J	<p><u>Tested:</u> 1 bear: 10 accessories. Tested exactly 10 bears with all NoiseMaker accessories.</p> <p><u>Conclusion:</u> Errors begin to occur when we begin to exceed 9 accessories on a single bear. I believe one of the errors are from having 10 items on a single bear. The 10% discount seems to be active once we have 10 items but it is hard to tell if the condition is set for 10 or 10+ accessories which could give the error; this could also be a conversion issue because the discount is \$14 but we get \$14.10000000001 as seen in the image below.</p> <p><u>Output:</u></p> <pre><testcase name="testJ[0]" classname="GivenBlackBox" time="0.04"> <failure message="java.lang.AssertionError: expected:<41.0>; but was:< 14.100000000000001>;" type="java.lang.AssertionError">java.lang.AssertionError: expected:<41.0>; but was:<14.100000000000001>;</pre>	<p>Errors in classes:</p> <p>0 - 4</p> <p>Easter Eggs:</p> <p>1 & 5</p>
Test K	<p><u>Tested:</u> 1 bear, discount = free ink when price of bear is above \$70 (with single item). Changed price of bear manually to \$71.</p> <p><u>Conclusion:</u> It appears as though an error is thrown when the price of the bear exceeds \$70 with only embroidery on the bear. We expected to save the 5 dollars from the embroidery but instead the system said that it expected save the total cost of the bear excluding the embroidery. This makes me think there is something wrong.</p> <p><u>Output:</u></p> <pre><testcase name="testK[0]" classname="GivenBlackBox" time="0.008"> <failure message="java.lang.AssertionError: expected:<5.0>; but java.lang.AssertionError: expected:<5.0>; but was:<71.0>;</pre>	<p>Errors in classes:</p> <p>0 - 4</p> <p>Easter Eggs:</p> <p>6</p>
Test L	<p><u>Tested:</u> 1 bear, discount = free ink when price of bear is at \$70 (with single item). Changed price of bear manually to \$65 so that with \$5 embroidery total cost hits \$70.</p> <p><u>Conclusion:</u> this furthers my ideas from Test K.</p>	<p>Errors in classes:</p> <p>0 - 4</p> <p>Easter Eggs:</p>

	<p>Output:</p> <pre><testcase name="testL[0]" classname="GivenBlackBox" time="0.009"> <failure message="java.lang.AssertionError: expected:<5.0> but java.lang.AssertionError: expected:<5.0> but was:<65.0>;</pre>	6
Test M	<p><u>Tested:</u> 1 bear, discount = free ink when price of bear is at \$70 (with varying items item). Changed price of bear manually to \$65 so that with \$5 embroidery total cost hits \$70 and to see if embroidery is free.</p> <p><u>Conclusion:</u> This test seems to shed light on the idea that embroidery is being ignored when it comes to discounts. We can see in the output that \$20 has been saved and not the \$5 from the embroidery like we anticipated.</p> <p>Output:</p> <pre><testcase name="testM[2]" classname="GivenBlackBox" time="0.005"> <failure message="java.lang.AssertionError: expected:<5.0> but java.lang.AssertionError: expected:<5.0> but was:<20.0>;</pre>	<p>Errors in classes:</p> <p>2</p> <p>Easter Eggs:</p> <p>6,7,8</p>
Test N	<p><u>Tested:</u> : 1 bear, discount = free ink when price of bear is above \$70 (with varying items item). Changed price of bear manually to \$80 so that we can see if bear with \$5 embroidery activates our discount for the embroidery.</p> <p><u>Conclusion:</u> This test furthers my ideas from the previous test. It appears that the embroidery never gets calculated in the savings.</p> <p>Output:</p> <pre><testcase name="testN[2]" classname="GivenBlackBox" time="0.011"> <failure message="java.lang.AssertionError: expected:<5.0> but java.lang.AssertionError: expected:<5.0> but was:<20.0>;</pre>	<p>Errors in classes:</p> <p>2</p> <p>Easter Eggs:</p> <p>6,7,8</p>
Test O	<p><u>Tested:</u> Ink Is free – below \$70.</p> <p><u>Conclusion:</u> Embroidery is never calculated. Whether it's with varying items, same items, below or above the \$70 mark. My second idea would be that embroidery does not work well when being calculated with other costs.</p> <p>Output:</p> <pre><testcase name="testO[2]" classname="GivenBlackBox" time="0.011"> <failure message="java.lang.AssertionError: expected:<0.0> but java.lang.AssertionError: expected:<0.0> but was:<4.0>;</pre>	<p>Errors in classes:</p> <p>2</p> <p>Easter Eggs:</p> <p>6</p>

• **Which BearWorkshop implementation adheres to the specification best? Is there one that passed all your test? If so which one is it and why?**

I think the best implementations were classes 0 & 1. They come up only a few times when testing and had almost no errors when running the tests. I am assuming that these classes had good implementations of the calculateSavings() because they were able to manage their bears properly. When I say this, I mean specifically that the accessories on the bears could be properly sorted and data retrieval was not impeded. Most of the problems found had to do with the variations of accessories on a given bear and the number of accessories on a single bear.

Task 4

The image shows a terminal window on the left and a web browser on the right. The terminal window displays the output of the command `gradle jacocoTestReport`. The output indicates that the build was successful and that 2 actionable tasks were executed. The command line `shaunvulaj@shaun-MacBook-Air CodeGiven 3 % gradle jacocoTestReport` is highlighted in yellow. The web browser displays the 'Test Summary' page, which shows the following statistics:

90	22	0	0.144s
tests	failures	ignored	duration
75% successful			

The 'Failed tests' tab is selected, showing a list of test cases:

- GivenBlackBox. testC[0]
- GivenBlackBox. testD[0]
- GivenBlackBox. testD[2]
- GivenBlackBox. testH[3]
- GivenBlackBox. testI[3]
- GivenBlackBox. testJ[0]
- GivenBlackBox. testJ[1]
- GivenBlackBox. testJ[2]
- GivenBlackBox. testJ[3]
- GivenBlackBox. testJ[4]
- GivenBlackBox. testK[0]
- GivenBlackBox. testK[1]
- GivenBlackBox. testK[2]
- GivenBlackBox. testK[3]
- GivenBlackBox. testK[4]
- GivenBlackBox. testL[0]
- GivenBlackBox. testL[1]
- GivenBlackBox. testL[2]
- GivenBlackBox. testL[3]

Figure 1* index.html file showing Jacoco test results from task 3. ID is highlighted in yellow in the command line.

