Task 2:

- 1. minor redundancy- instead of having only one constructor class, with a variable for passing, there are two; one default, one with the variable. It's in nearly every class.
- 2. BearWorkshop has this but worse, in that it has a default constructor whose only function is to call a more specialized constructor with a string parameter
- 3. overall, uses a lot of for loops instead of simple multiplication
- 4. who formats like this? BearWorkshop, line 128; add and remove bear functions are needlessly complex and redundant; instead of if statements, and difficulty formatted ones, you could simply 'return this.BearCart.remove(bear)'
- 5. Too many nested if statements instead of ifs with && or ||
- 6 (bonus!). in Noisemaker, a switch statement is used when if/else would have worked fine.

Task 3

To test the 3-for-price-of-2 rule, I made several tests: one to test whether the method holds up for higher numbers of clothes, whether the method hold up for clothes with unequal prices in unexpected configurations, and whether the method can deal with high numbers of clothes in unequal prices.

- The buy2Get3ClothesDifferentPrice() method failed on classes one and three. This method tests
 whether a set of 3 clothes with different prices will choose the cheapest clothes item to take off
 as free.
- 2. The buy2Get3ClothesHiNumbers() method failed on the first four classes but succeeded on the last. This method tests whether or not the classes could deal with correctly calculating discounts for up to fifteen clothes with the same price. I got easter egg 2 from this class.
- The buy2Get3ClothesDifferentPriceHiNum() method failed on all comers. This method tested
 whether or not classes could deal with up to fifteen clothes with different prices, which they
 could not.

To test the rule for 10% off cost of bear with ten or more accessories, we would need to test whether this method works with all accessories (clothes, noisemakers, embroidery). It would also need to interface well with the 3-for-price-of-2 clothes rule.

(Note: this test also proved that the 3-for-price-of-2 clothes rule for cooperation, by testing whether or not it works with multiples and with other discounts)

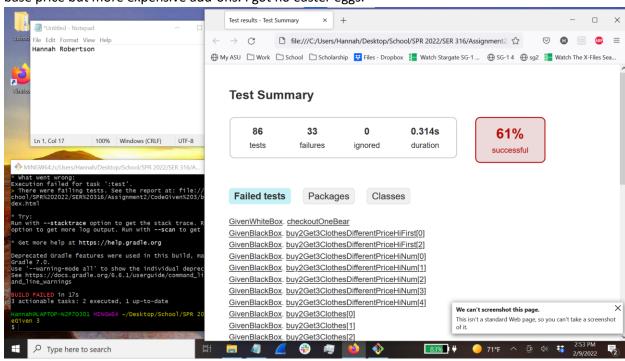
- 1. My oneBearTest10Clothes() methods proved that the 10% off with 10 paid for clothes method interacted well with the buy 2 get one free clothes method in the first 3 classes, but flagged in the fourth and fifth. I got easter eggs 1, 7, 5, and 8.
- 2. However, my oneBearTest10Noisemakers() methods proved that the method again worked on the first three classes but not on 4 and 5, oddly when it was noisemakers that were applied; the method took almost 2/3 of the price off the bear. For this one I got easter egg 3.

3. My oneBearTest10LetterEmbroidery() method failed on all tests. Are we not counting embroidery as accessories? I figured it fits the criteria of 'anything on the bear.' I didn't find any easter eggs for this method.

To test the buy 2 bears, get one free:

You would need to check that it works when bears are of equal price, that it works when all bears have accessories, and checks that it works when the price of a bear is checked with the accessories.

- 1. My threeBearsSaveOnCheapestEqual() method proves the method works well in all test classes, when three bears are equal in price it returns the correct price.
- 2. The threeBearsSaveOnCheapestAccessories() method is meant to test what happens when all three bears have accessories. The first four classes return the correct lowest price, but the fifth returns the highest price. I got easter eggs 6 and 8.
- 3. The threeBearsSaveOnCheapestAccessoriesFoam() is meant to test whether the accessories are counted, or just the base price. For this, I had two down bears (as they are more expensive as a base than the regular bears) and then added enough accessories to a regular bear to make it more expensive. All but class 3 passed this task, but 3 instead returned the bear with the lower base price but more expensive add-ons. I got no easter eggs.



Task 4:

