CS2001 Practical W05-JUnit

Tutor: Simon Dobson

Overview

In this practical I was asked to write suitable tests to test and then implement all the methods to satisfy the tests written following a Test-Driven Development process. The methods to be written are related to a shop that is able to have stock records of different products. These products can be bought from and added to the shop for example.

Design

When designing the test cases for my program I made them so that they would thoroughly test every method/action that the program undertook. To achieve this, I first wrote tests that used expected input values to test that action works as expected under normal conditions. I then wrote tests that used input values that are on the boundaries of the range of the value to test that even when the action has to deal with boundary values it still produces the expected results. To test the robustness of my program I then used input values that the method doesn't expect and check that it handles this situation as intended and doesn't fail or break.

When choosing what to test I looked at what resulting values the methods were trying to find and what different values had to be found to get the resulting values.

When implementing my program, I wrote the code to make all the tests pass, whereas before sometimes I would have to alter the code after writing it to accommodate for the unexpected input values for example but using this method of writing tests I bared that in mind whilst writing the code first time around.

To store the products in the shop I used the stock record object. This meant that values such as stock count, number of sales for a particular product could be accessed using the stock record object rather than having to keep track of those values in the shop object.

To store the stock records I used an ArrayList<IStockRecord>. I chose this way of storing the stock records because it meant adding and removing is easy using an array list. As you are able to access the array by index this meant it was easy to traverse through the array when I had to work out the most popular product for example. Array list doesn't have a fixed length so meant products are able to be added and removed without any limitation.

Testing

The tests I have chosen to test my program were selected to test all different aspects of my program. That my program can handle and normal and expected input, that boundary values result as expected and that values that aren't expected don't cause the program to break or cause any problems.

When running the stacscheck 3 out of the 3 tests pass.

What is being tested?	**Test method name**	**Expected output**	**Actual Output**	**Pass/ Fail**
The factory was able to create a non-null instance of IProduct.	ShopProduct CreationNonNull()	product != null	product != null	Pass
The factory was able to create a non-null instance of IShop.	ShopCreation NonNull()	shop != null	shop != null	Pass
The factory was able to create a non-null instance of IStockRecord.	ShopStockRecord CreationNonNull()	stockRecord != null	stockRecord !=null	Pass
The get bar code method returns the bar code string correctly.	ShopProduct CorrectBarcode()	product.getBarCode() = "121212"	product.getBarCode() = "121212"	Pass
The get description method returns the correct description value.	ShopProduct CorrectDescription()	product.getDescription() = "Washing Machine"	product.getDescription() = "Washing Machine"	Pass

That a product could be registered in the shops.	addProduct()	shop.getNumber OfProducts() = 1	shop.getNumber OfProducts() = 1	Pass
That a product with the same bar code as an existing product won't be added.	AddMultipleProducts SameBarcode()	shop.getNumber OfProducts() = 1	shop.getNumber OfProducts() = 1	Pass
That the unregister method removes the product from the shops products.	unregisterProduct()	shop.getMostPopular = null	shop.getMostPopular = null	Pass
That stock can be added for a product and correct amount of stock is recorded.	addStockToProduct()	shop.getStock Count(product .getBarCode()) = 2	shop.getStock Count(product .getBarCode()) = 2	Pass
That stock cannot be added to a product that isn't registered in the shop.	addStockTo UnregisteredShop()	shop.getStock Count(product .getBarCode()) = 0	shop.getStock Count(product .getBarCode()) = 0	Pass
That after a product has been bought the products stock count is correct.	buyAProduct CorrectStockCount()	shop.getStock Count(product .getBarCode()) = 2	shop.getStock Count(product .getBarCode()) = 2	Pass
That after a product has been bought the products number of sales is correct.	buyAProduct CorrectNumber OfSales()	shop.getNumber OfSales(product .getBarCode()) = 2	shop.getNumber OfSales(product .getBarCode()) = 2	Pass

That a product cannot be bought, and stock level remains at zero.	buyAProductNoStock()	shop.getStock Count(product .getBarCode()) = 0	shop.getStock Count(product .getBarCode()) = 0	Pass
That a product cannot be bought if it hasn't been registered in the shop.	buyAUnregistered Product()	shop.getNumber OfSales(product .getBarCode()) = 0	shop.getNumber OfSales(product .getBarCode()) = 0	Pass
The total number of products in the shop is correct.	totalNumberOf Products()	shop.getNumber OfProducts() = 4	shop.getNumber OfProducts() = 4	Pass
The total number of products in the shop is correct after removing products.	totalNumberOf ProductsAfter Removing()	shop.getNumber OfProducts() = 2	shop.getNumber OfProducts() = 2	Pass
The total stock count of the shop is correct.	totalStockCount()	shop.getTotalStockCount() = 12	shop.getTotalStockCount() = 12	Pass
The total stock count is correct after products have been bought.	totalStockCountAfter BuyingProducts()	shop.getTotalStockCount() = 9	shop.getTotalStockCount() = 9	Pass
The stock count for a particular product is correct.	stockCountParticular Product()	shop.getStock Count(product1.getBar Code()) = 3	shop.getStock Count(product1.getBar Code()) = 3	Pass
The stock count for a particular product is correct after the product has been bought.	stockCountParticular ProductAfter Purchasing()	shop.getStock Count(product1.getBar Code()) = 2	shop.getStock Count(product1.getBar Code()) = 2	Pass

If a product isn't registered in the shop its stock count will be zero.	unregisteredProduct StockCount()	shop.getStock Count(product2.getBar Code()) = 0	shop.getStock Count(product2.getBar Code()) = 0	Pass
The number of sales for a particular product is correct.	numberOfSales ParticularProduct()	shop.getNumberOf Sales(product1.getBar Code()) = 3	shop.getNumberOf Sales(product1.getBar Code()) = 3	Pass
If a product isn't registered in the shop its number of sales will be zero.	numberOfSales UnregisteredProduct()	shop.getNumberOf Sales(product2.getBar Code()) = 0	shop.getNumberOf Sales(product2.getBar Code()) = 0	Pass
The product with the most sales is returned as the most popular.	mostPopularProduct()	shop.getMostPopular() = product4	shop.getMostPopular() = product4	Pass
That if there are no products registered in the shop null is returned as most popular product.	mostPopularNo RegisteredProducts()	shop.getMostPopular() = null	shop.getMostPopular() = null	Pass
The correct product is returned from the stock record.	stockRecordCorrect Product()	shop.getProduct() = product	shop.getProduct() = product	Pass
That the register product method can handle a null value.	registerProductNull()	shop.getNumberOf Products() = 0	shop.getNumberOf Products() = 0	Pass
That the unregister product can handle a null value.	unregisterProdudtNull()	shop.getNumberOf Products() = 0	shop.getNumberOf Products() = 0	Pass

That the add stock method can handle a null value.	addStockNullBarcode()	shop.getTotalStock Count() = 0	shop.getTotalStock Count() = 0	Pass
That the buy product method can handle a null value.	buyProductNull Barcode()	shop.getNumberOf Sales(product.getBar Code()) = 0	shop.getNumberOf Sales(product.getBar Code()) = 0	Pass
That the get stock count method can handle a null value.	stockCountNull Barcode()	shop.getStockCount(null) = 0	shop.getStockCount(null) = 0	Pass
That the get number of sales method can handle a null value.	numberOfSales NullBarcode()	shop.getNumberOf Sales(null) = 0	shop.getNumberOf Sales(null) = 0	Pass

Evaluation

From the tests I have conducted above I believe I have thoroughly tested my program in many different scenarios and situations. As all my tests above passed by the expected result matching the actual result this shows that my solution fits the requirements that were given in the specification.

Conclusion

In conclusion, I found this practical to be good and it showed that using a test driven development style can lead to successful results. I enjoyed using this style to create my program as it was different to how I had made projects previously and gave me the opportunity to learn more about writing different styles of test to test my program.

At first I found it difficult to deal with the different exceptions that were to be handled by each of the methods. However, after doing some of my own research this made it clearer as to how to handle custom exceptions properly.