# Software Quality Assurance and Testing

# Workbook for Experiment 1

Enter your name and student number below.

**Name: Zenkai Young**

**Student Number: 20175181**

Answer the following questions.

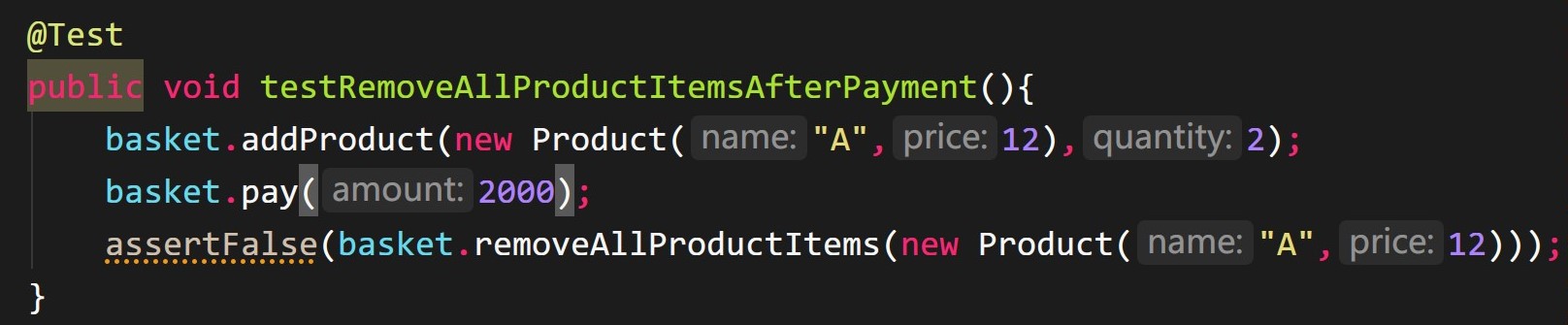
## Question 1

What problems did you find with your tests. List each problem and explain which method you found the problem in and the test that shows the problem.

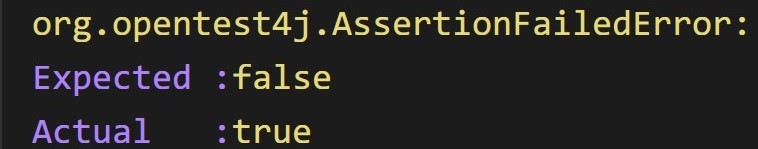
1. After payment, we still can remove items in Basket:

In removeAllProdcutItems(Product product). After payment is done, we should not be able to remove items in Basket. But actually, we can. Because after this function check addItemsEnabled, it does nothing where it should return false.

I use this test below to test this condition.



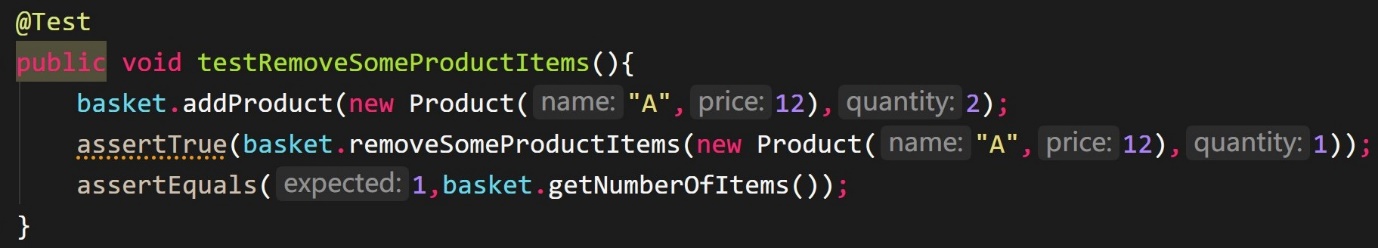
And the results:



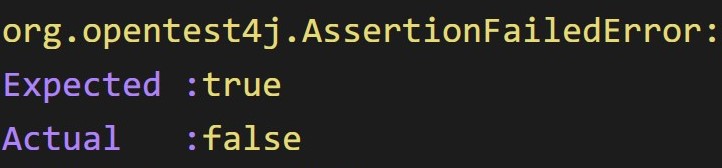
1. We remove some products items exceptionally.

In removeSomeProductItems(Product product, int quantity). We can’t normally remove some products items. Because a condition statement is wrong. When this function checks item is not null, it should decreaseQuantity and return true. But on the contrary, it return false which cause us removing products items exceptionally.

I use this test below to test this condition:

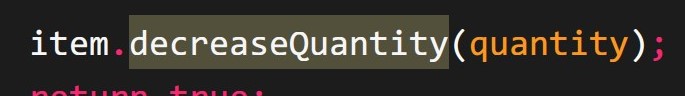


And the results:

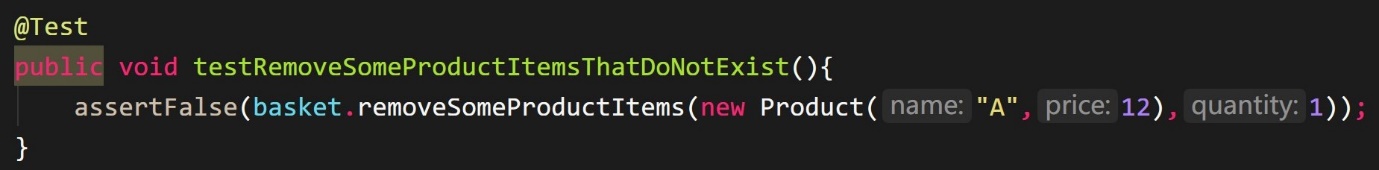


1. An unhandled NullPointerException

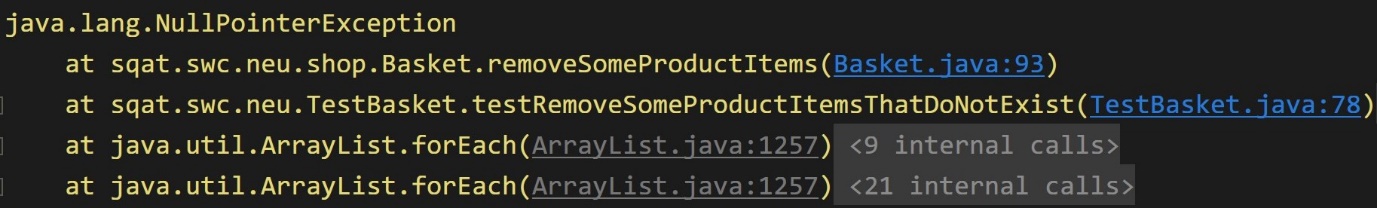
Still in removeSomeProductItems(Product product, int quantity). If the product paramenter doesn’t exist in Basket, item is null. And a null pointer can’t call method like below:



I use this test below to test this condition:



And the results:



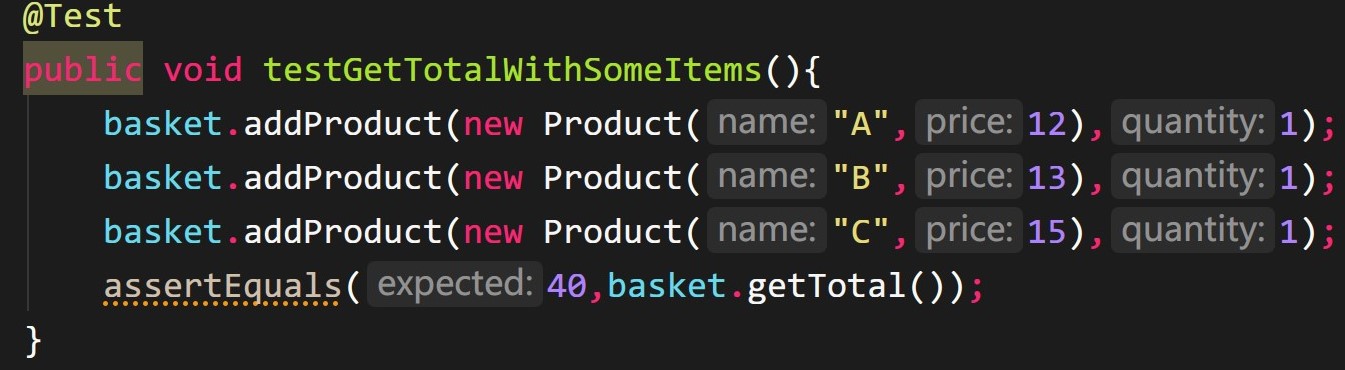
1. We can remove items that exceed the amount:

Although we can’t remove items properly, there is an issue which is removeSomeProductItems allow us to remove items more than what basket have and doesn’t do some protection.

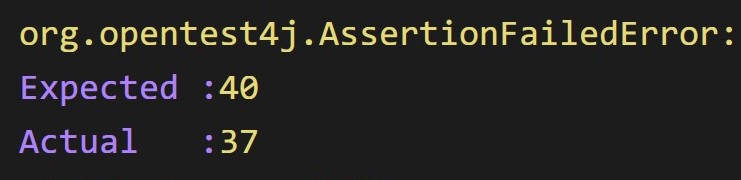
1. We get a wrong total when we call getTotal()

After we call getTotal() to calculate the total. It seems to get a wrong answer. Because in the end of getTotal, we should decrease total by priceDiscount rather than (priceDiscont + item.getQuantity()) like below:

I use this test below to test this condition:



And the results:



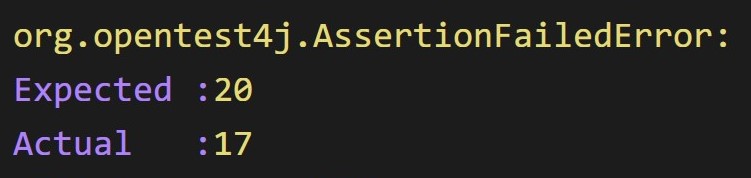
1. We get wrong total when we call getTotal() after we have some discounts

Same reason like above.

I use this test below to test this condition:



And the results:



## Question 2

Does the method containsProduct() in BasketItem contain any conditions that need to be tested? Explain your answer.

I think we should check if our member variable---product is null. Or we may get a NullPointerException.

## Question 3

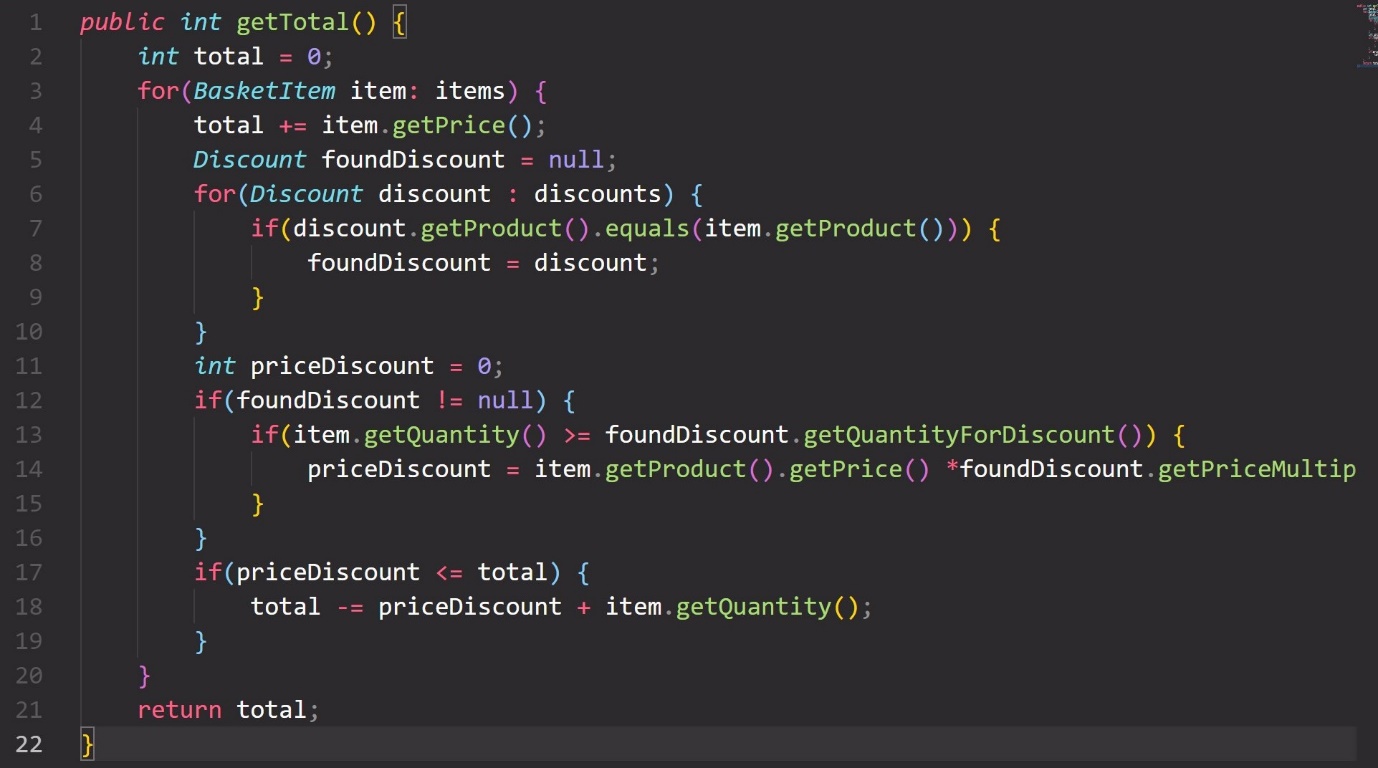
What is the Cyclomatic Complexity for the method getTotal() in the Basket class?

According to the basis path we draw in Q4, the Cyclomatic Complexity for the method getTotal() is 7.

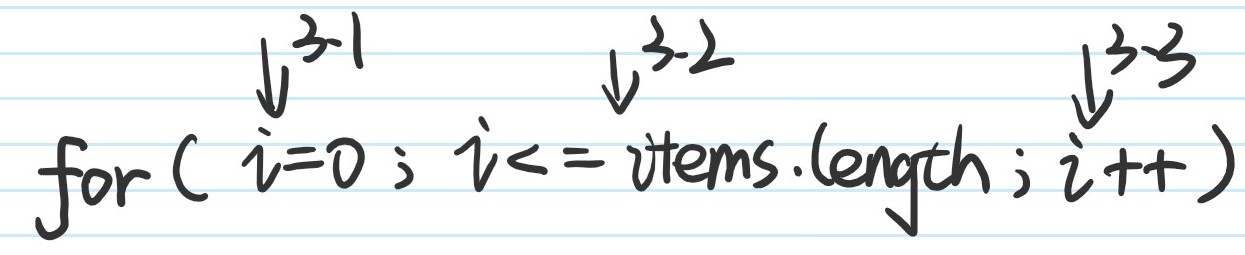
## Question 4

What are the basis paths that you identified for the method getTotal() in the Basket class?

You can draw them with a drawing tool or you can draw them on paper and take a photograph to include in this document.



Since for(BasketItem item: items) is equivalent to for(int i=0;i<=items.length;i++) we divide a for-loop like line 3 into 3 parts like below:



I draw the basis path of getTotal() like below:

