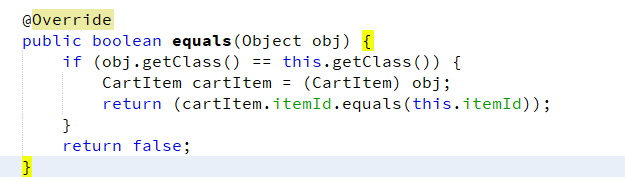
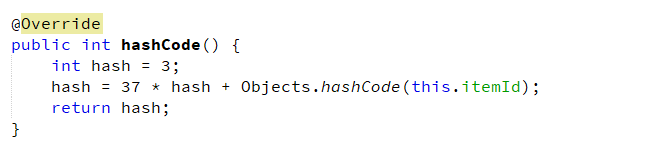
6.1P

Task 2.

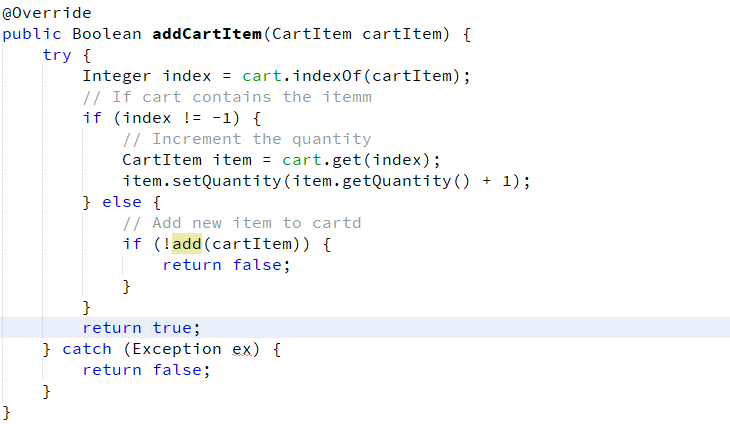
**addCartItem**

First to check if the cart contains an item I overrode the *equals* and *hashCode* methods of CartItem so that the arraylist method *indexOf* will return the items index of a cartItem with the same id if it exists, otherwise it will return -1.



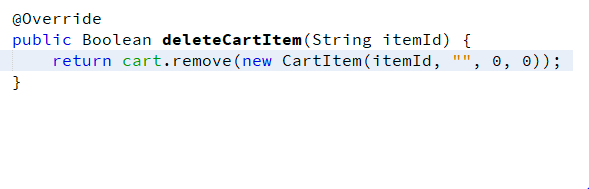


Here our method checks if the item exists and has an index in the cart. If it does, it retrieves the object and increments its quantity. Otherwise it attempts to add an item to the card. The method returns false only on an exception.



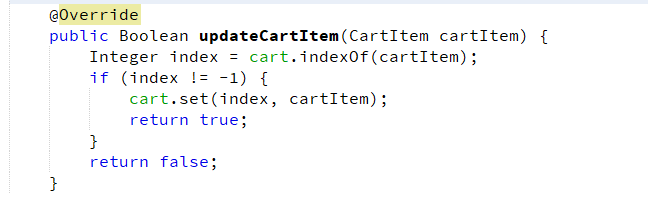
**deleteCartItem**

The arraylist already has this method inbuilt. Arraylist.remove(object) will find the first occurrence of an object & if it finds one it will delete it and return true, otherwise it will return false.  
Because I’ve overridden equals to return true if two cartItems have the same id I can just create a new cartItem with that id to pass as a parameter.



**updateCartItem**

This checks if the item exists using the index, then uses the arraylist set method to replace the item.



Task 4. Testing

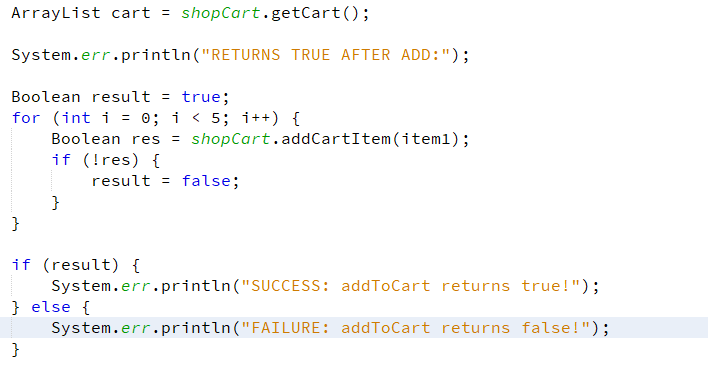
For testing purposes I changed the quantities of the cart items to 1 to be less confusing.



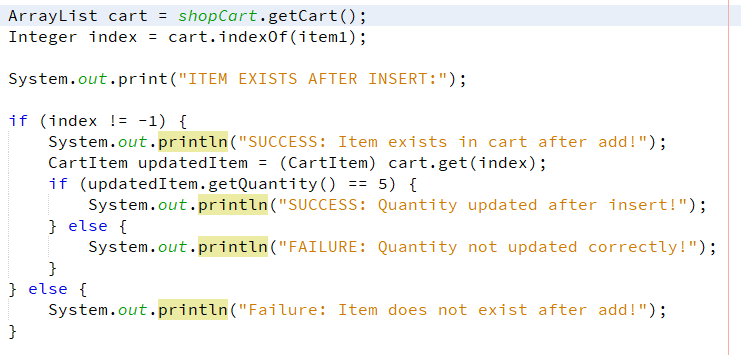
**addToCart**

This test adds 5 items to the cart. It then checks if the item exists by finding its index. It then uses the index to retrieve the item and check that its quantity is equal to 5.

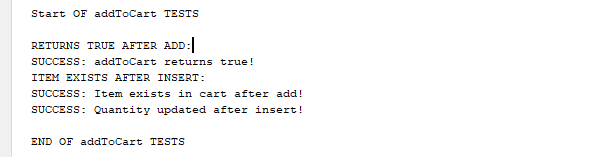
First we test that it returns true on a successful add:



Then we check that the item exists in the cart and its quantity has been updated appropriately.

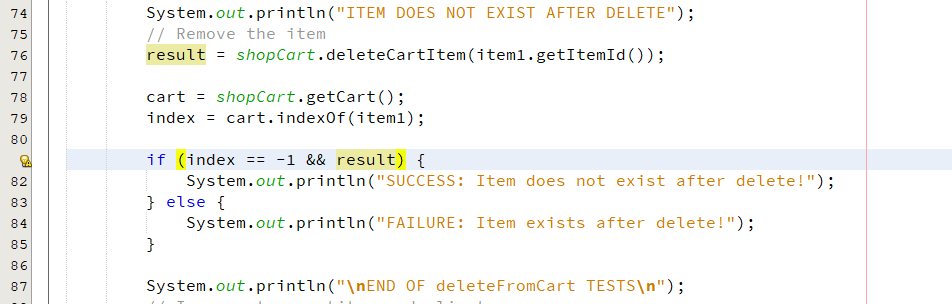


This is the console output from the test



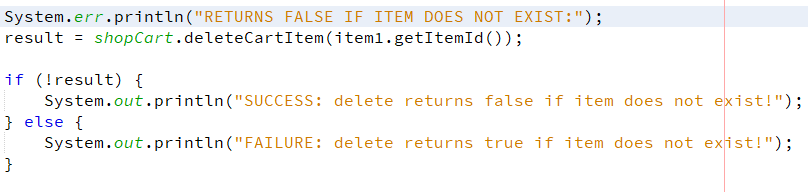
**deleteFromCart**

This is the test:

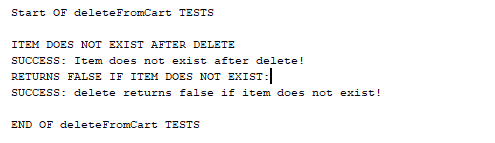


It calls the delete method and checks that it returns true, and then checks that indexOf returns -1 indicating the item does not exist.

The second test checks that it returns false if the item does not exist.

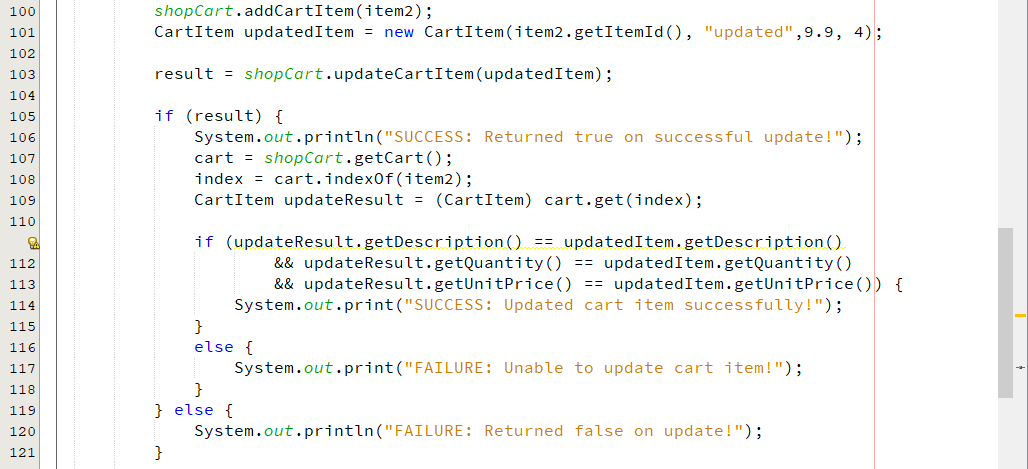


Console output:

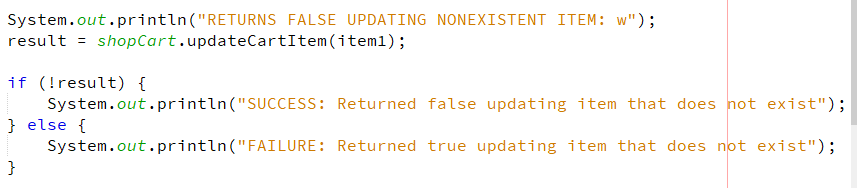


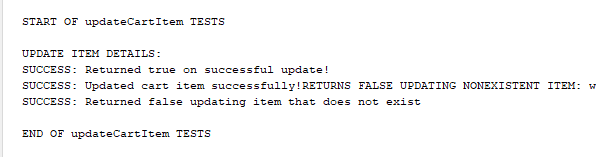
**Update item**

In this test I create a new item with the same id as item 2. If the update returns true it checks if the item is updated by checking its fields.



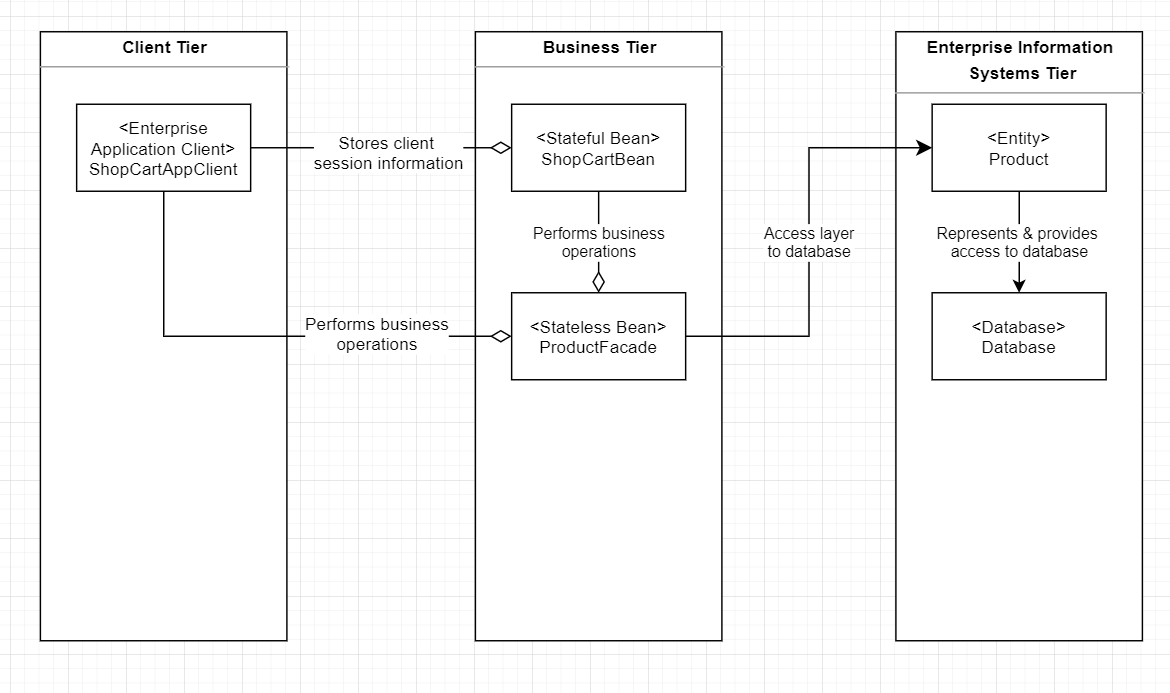
The second test attempts to update an item not in the cart and checks if it returns false.





Task 5

5.1 Design



5.2

ShopCartClient

This is in the client tier as this software will exist on the client’s hardware, rather than on a server instance. Its role is providing an interface for either the user or their system to interact with the enterprise server & database. It does this by making calls to the servers’ remote interfaces over the network.

ShopCartBean

Its role is to store the client’s session information over the duration of the client’s interaction with the system, and to persist this information if necessary.

ProductFacade

This bean would provide business method calls either directly to client, or to other parts of the enterprise system.

Product

The product entity here represents all entities in the system. These are used to define & access the database via objects. These will be used directly by the beans to edit the data in the database, & indirectly represented by DTO’s to provide read-only access to the database’s information.

Database

The information is stored here and is only access via the entities. This is to simplify accessing data as it is directly represented as objects.