Bug Report & Test Results

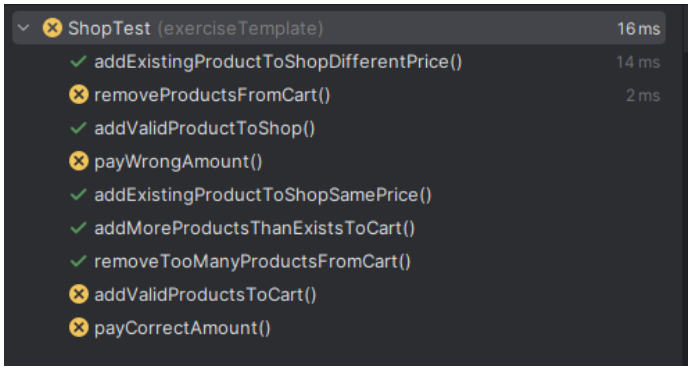
Test	Method(s)	Input	Expected Output/State	Pass/Fail	Post Bugfix
addValidProductToShop	addProduct	"Pancakes", 0, 1.70	howManyInStock=0	pass	pass
addExistingProductToShopDifferentPrice	addProduct	"Egg", 10, 0.19	false	pass	pass
addExistingProductToShopSamePrice	addProduct	"Egg", 10, 0.20	howManyInStock=20	fail	pass
addValidProductsToCart	addToCart	"Egg", 5	howManyInStock=5, howManyInCart=5	fail	pass
addMoreProductsThanExistsToCart	addToCart	"Egg", 5	false	pass	pass
removeTooManyProductsFromCart	addToCart, removeFromCart	"Egg", 11	false	pass	pass
removeProductsFromCart	addToCart, removeFromCart	Egg, 5	howManyInStock=10, howManyInCart=0	fail	pass
payCorrectAmount	addToCart, payCart	1,00	true, cart is empty	fail	pass
payWrongAmount	addToCart, payCart	0,99	false, 5 eggs in cart	fail	pass

UseCaseTest	Method(s)	Input	Expected Output/State	Pass/Fail	Post Bugfix
buyMultipleProductsIndecisiveCustomer	addToCart, removeFromCart, payCart	"Egg", 4, "Toilet Paper", 20.00	true	fail	pass

Tests that need to be checked up	Method(s)	Input	Expected Output/State	Pass/Fail	Post Bugfix
addNewProductNegativeValueNegativePrice	addProduct	"Pancakes", -1, -1.70	false	fail	fail
addNegativeAmountToCart	addToCart	"Egg", -1	false	fail	fail

The final test suite has 9 Junit tests and one use case test. Two tests were excluded, as the system's desired behaviour was not specified.

Bug fixes



Initially, some Junit tests fail. These tests fail, as they expect a state of 5 eggs in the cart, but the methods return -5:

- payWrongAmount()
- addValidProductsToCart()
- payCorrectAmount()

One test fails as the cart should be empty, but the items are still left:

- removeProductFromCart()

Two lines were changed as bugs were found:

The method addToCart subtracted an amount, which led to negative values for items in the shopping cart. This code was changed, as positive integers are more intuitive in the cart.

The removeFromCart method had a more-than condition that prevented it from removing items if the removed items were as many as the total number of items in the cart. The condition was changed to more than.

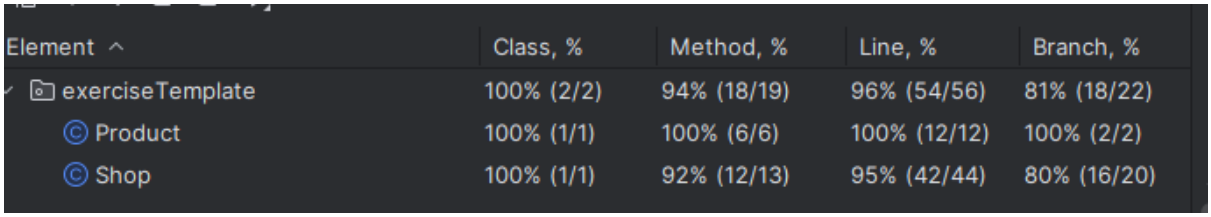
Additional Considerations

Tests that need to be checked up	Method(s)	Input	Expected Output/State	Pass/Fail	Post Bugfix
addNewProductNegativeValueNegativePrice	addProduct	"Pancakes", -1, -1.70	false	fail	fail
addNegativeAmountToCart	addToCart	"Egg", -1	false	fail	fail

The system currently allows negative prices and product inventories, and it is beneficial to specify the desired system behaviour before further development.

- Is negative product price allowed?
- Is negative product stock allowed?
- Are negative products allowed in the cart?

Code Coverage After Bugfixes



Element	Class, %	Method, %	Line, %	Branch, %
exerciseTemplate	100% (2/2)	94% (18/19)	96% (54/56)	81% (18/22)
Product	100% (1/1)	100% (6/6)	100% (12/12)	100% (2/2)
Shop	100% (1/1)	92% (12/13)	95% (42/44)	80% (16/20)

After fixing these minor bugs the chosen tests pass and the code coverage is decent, considering that the Junit tests do not cover all the paths discovered in the Shop class.

Final Thoughts

It is recommended to write more Junit tests based on the test design document and clarify the requirements further. These tests result in a decent line of code coverage, but do not cover all methods, nodes and edges. Furthermore, a more detailed examination of allowed values is recommended with boundary value analysis and input space partitioning.