MSc. in Applied Software Engineering

PC#2 Mockito Assessment 2023 – Friday 17th November

Time : 11.15 – 13.15 – At 13.15 upload your Test class to moodle. Moodle will close automatically at 13.20

Notes on marking scheme:-

1. **Your test code must compile. Automatic zero for non-compiling code.**
2. **Do not modify any of the code you have been given. Complete the class CheckOutServiceImplTest.java**
3. **Set up the project with the packages as given.**
4. **You test should be passing before it will be marked. No marks for partially completed/failing tests. Tests should include asserts and verifys as specified in the test descriptions below. Name the tests as specified below.**
5. **What to submit :- 1. Your java test class CheckOutServiceImplTest.java**

shop is an online shopping application that implements a shopping cart.

`CheckoutServiceImpl.java` implements the logic for checking out a customer's shopping cart. The class encapsulates the logic for checking out items in a shopping cart, including checking item availability, processing payment based on the customer's chosen payment method, and updating inventory and notifications accordingly. The `checkout` method is the main function of this class

It is used to process a customer's checkout. It takes a Customer object as a parameter.

- It first retrieves the customer's shopping cart using `customer.getCart()`.
- It checks if the cart is empty. If the cart is empty, it throws a `CartEmptyException` indicating that the customer cannot proceed with an empty cart.
- It then iterates through the items in the cart and checks if each item is available in the inventory by calling `inventoryService.isAvailable(item.getProductId(), item.getQuantity())`. If **any item** is not available in sufficient quantity, it throws an `InventoryException` indicating that the product is out of stock or doesn't have enough quantity.
- It retrieves the customer's payment strategy and calculates the total amount of the items in the cart.
- Depending on the payment type chosen by the customer (CreditCard or PayPal), it processes the payment using the `paymentGateway`. If any payment-related exception occurs, it catches it and throws a `PaymentException` indicating an error processing the payment.

- After successful payment processing, it deducts the purchased items from the inventory using `inventoryService.deductItem(item.getProductId(), item.getQuantity())`.
- Finally, it notifies the `notificationService` that the order has been processed by calling `notificationService.notifyOrderProcessed(cart)`.

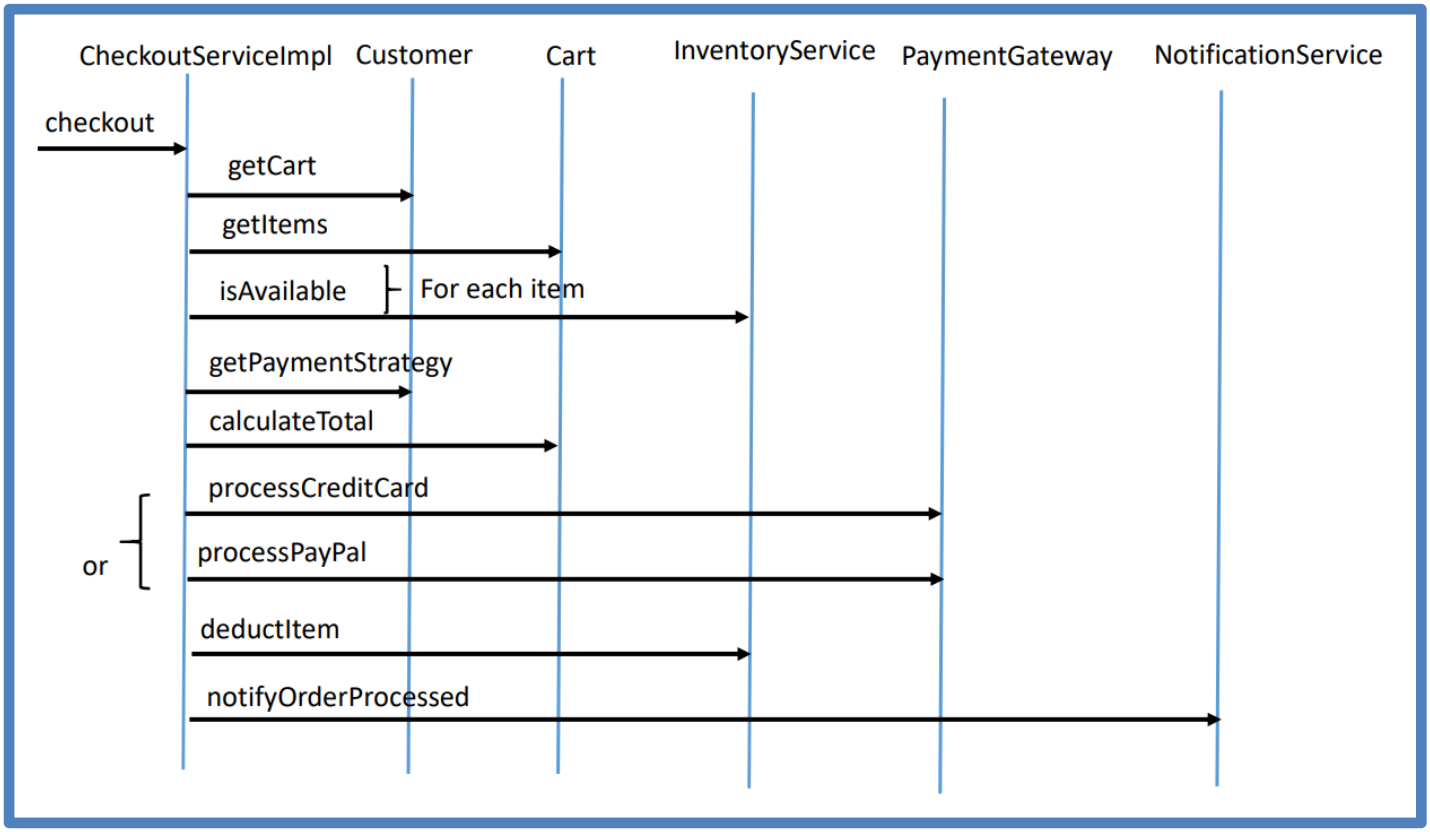
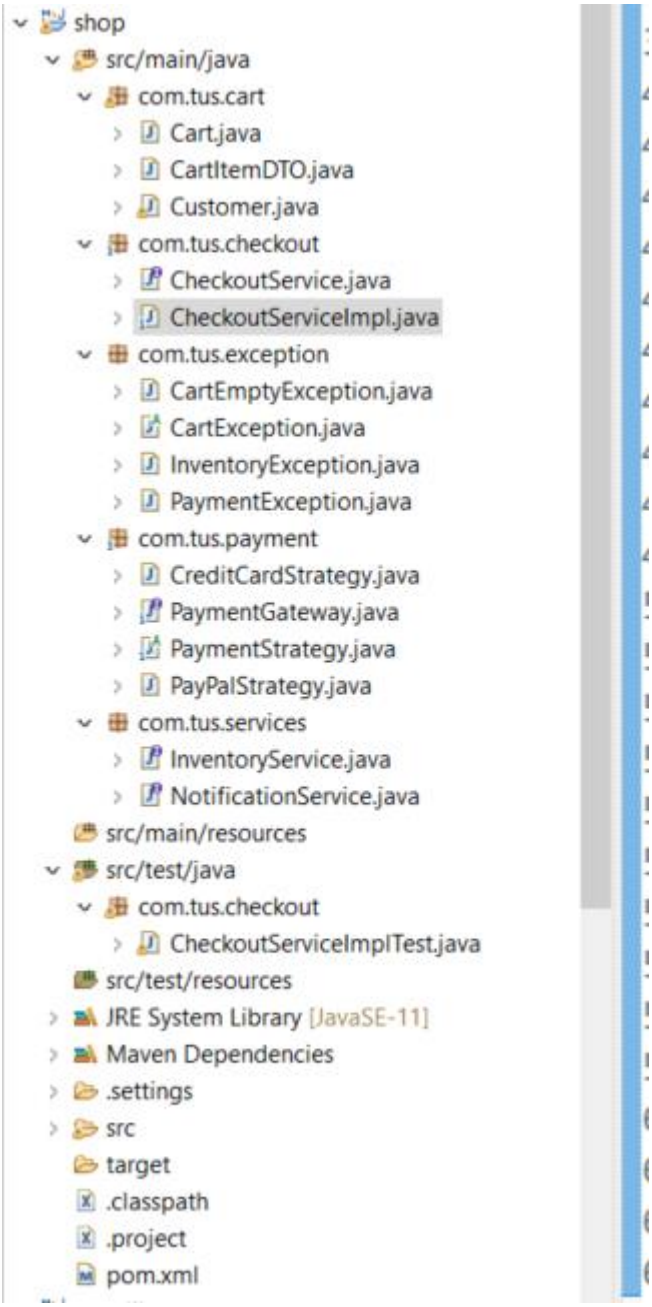


Figure 1 – Positive Flow

You are given the classes and interfaces below.



Complete the test class called `CheckoutServiceImplTest.java` that will test `CheckoutServiceImpl.java`.

Test 1 `testEmptyCartException`

Write a test that checks that CartEmptyException is thrown when there are no items added to the cart. Test that the correct message is received in the exception. The payment type “PayPal” should be used.
Verify that there is no interaction with the mocks for PaymentGateway (processPayPal), InventoryService (deductItem) or NotificationService (notifyOrderProcessed).

Test 2 `testInventoryNotAvailableException`

Write a test that checks that InventoryException is thrown when inventory is not available for the one item in the cart. Check that the correct message is provided in the exception. Verify that there is no interaction with the mocks for PaymentGateway (processPayPal), InventoryService (deductItem) or NotificationService (notifyOrderProcessed).

Test 3 `testPayPalProcessingException`

Write a test that checks that PaymentException is re-thrown when processing a PayPal payment. Check that the correct message is provided in the exception. Verify the correct interaction with the mocks for PaymentGateway (processPayPal). Verify that there is no interactions with the mocks for InventoryService (deductItem) or NotificationService (notifyOrderProcessed).

Test 4 `testCreditCardProcessingException`

Write a test that checks that PaymentException is re-thrown when processing a Credit Card Payment .Check that the correct message is provided in the exception. Verify the interaction with the mocks for PaymentGateway (processCreditCard). Verify that there is no interaction with the mocks for InventoryService (deductItem) or NotificationService (notifyOrderProcessed).

Test 5 `testPayPalOneCartItemSuccess`

Write a test that checks the correct interactions with the mocks for PaymentGateway (processPayPal), InventoryService (deductItem) and NotificationService (notifyOrderProcessed) when there is one item in the cart and payment type is paypal

Test 6 `testCreditCardOneCartItemSuccess()`

Write a test that checks the correct interactions with the mocks for PaymentGateway (processCrdeitCard), InventoryService (deductItem) and NotificationService (notifyOrderProcessed) when there is one item in the cart and payment type is credit card

Test 7 `testPayPalTwoCartItemSuccess()`

Write a test that checks the correct interactions with the mocks for PaymentGateway (processPayPal), InventoryService (deductItem) and NotificationService (notifyOrderProcessed) when there are two items in the cart and payment type is paypal