QAP 1

SDAT and Dev Ops

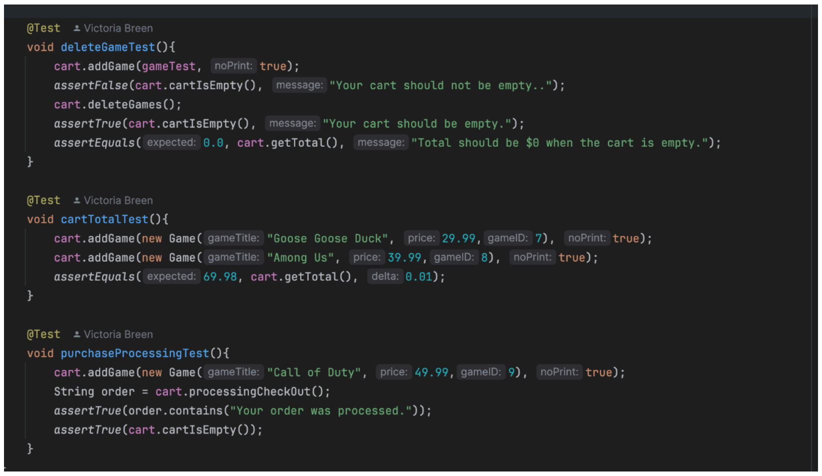
Game Store System

Victoria Breen

February 7, 2025

Documentation:

1. Clean Code Practices

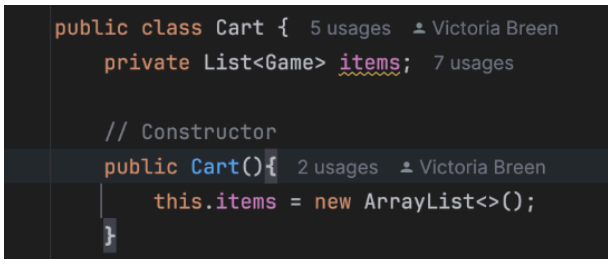


```java
@Test  Victoria Breen
void deleteGameTest(){
    cart.addGame(gameTest,  noPrint: true);
    assertFalse(cart.cartIsEmpty(),  message: "Your cart should not be empty..");
    cart.deleteGames();
    assertTrue(cart.cartIsEmpty(),  message: "Your cart should be empty.");
    assertEquals( expected: 0.0, cart.getTotal(),  message: "Total should be $0 when the cart is empty.");
}

@Test  Victoria Breen
void cartTotalTest(){
    cart.addGame(new Game( gameTitle: "Goose Goose Duck",  price: 29.99, gameID: 7),  noPrint: true);
    cart.addGame(new Game( gameTitle: "Among Us",  price: 39.99, gameID: 8),  noPrint: true);
    assertEquals( expected: 69.98, cart.getTotal(),  delta: 0.01);
}

@Test  Victoria Breen
void purchaseProcessingTest(){
    cart.addGame(new Game( gameTitle: "Call of Duty",  price: 49.99, gameID: 9),  noPrint: true);
    String order = cart.processingCheckOut();
    assertTrue(order.contains("Your order was processed."));
    assertTrue(cart.cartIsEmpty());
}
```

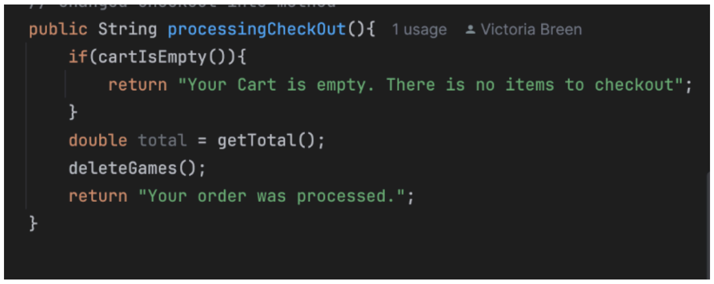
The first practice is all of my three unit tests follow the one specific responsibility rule for the cart class. They are required to validate a single expected outcome, which makes it easier to debug.



```java
public class Cart {  5 usages  Victoria Breen
    private List<Game> items;  7 usages

    // Constructor
    public Cart(){  2 usages  Victoria Breen
        this.items = new ArrayList<>();
    }
```

The second practice was using List rather than arrays. This makes List<Game> more flexible and dynamic, it also prevents items from being reassigned somewhere else in the code.



```java
public String processingCheckOut(){  1 usage  Victoria Breen
    if(cartIsEmpty()){
        return "Your Cart is empty. There is no items to checkout";
    }
    double total = getTotal();
    deleteGames();
    return "Your order was processed.";
}
```

The last practice I used was descriptive naming regarding variable and method names. For example, processingCheckout is the method that handles the purchasing processing in the game system.

2. My project is a game store management system that allows the user to add/delete games (However, I did preload eight games that I normally play myself), can view all of the games available in the system, and can display what has been added to the cart and calculate the cart total price. When you run the code, it will give you four options to choose from; the user is to input the number according to the option they would like to complete in the main menu. The test cases I used all focus on the cart class and the different methods it could run.

3. The two dependencies I used in this project were **junit-jupiter-api** and **junit-jupiter-engine.** These allowed for Junit testing in the project. Both of them came from https://mvnrepository.com.

4. Overall, I didn't struggle a lot with this QAP. I did have a couple problems when it came to the start of the project and the github actions but I rewatched some of the classes and it fixed up the confusion. The main issue for me was myself. I've been trying to catch up in this class as I have been struggling in another class and that has taken most of my time.