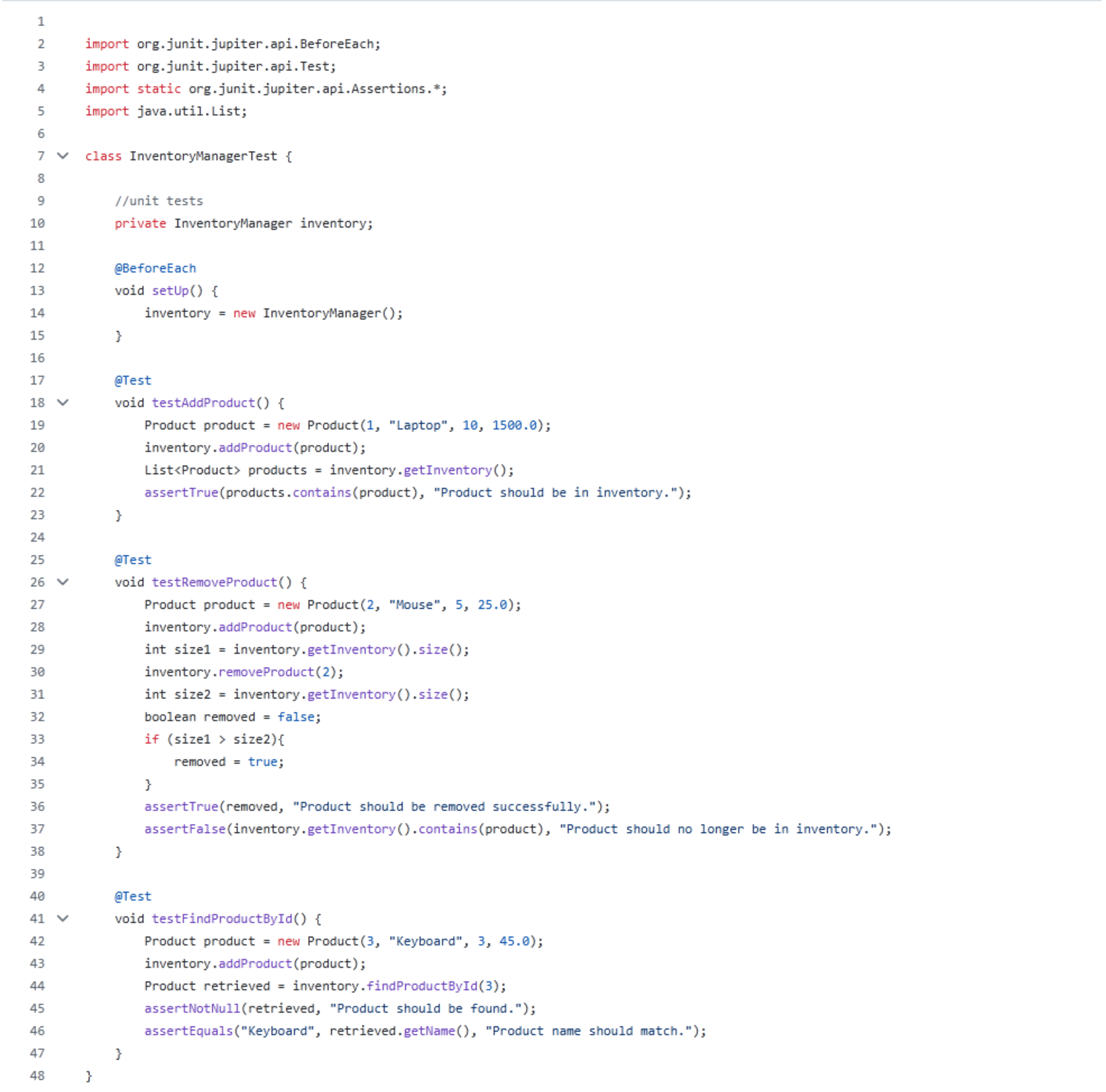
CT70A3000 Software Maintenance

D3- Individual project - Inventory

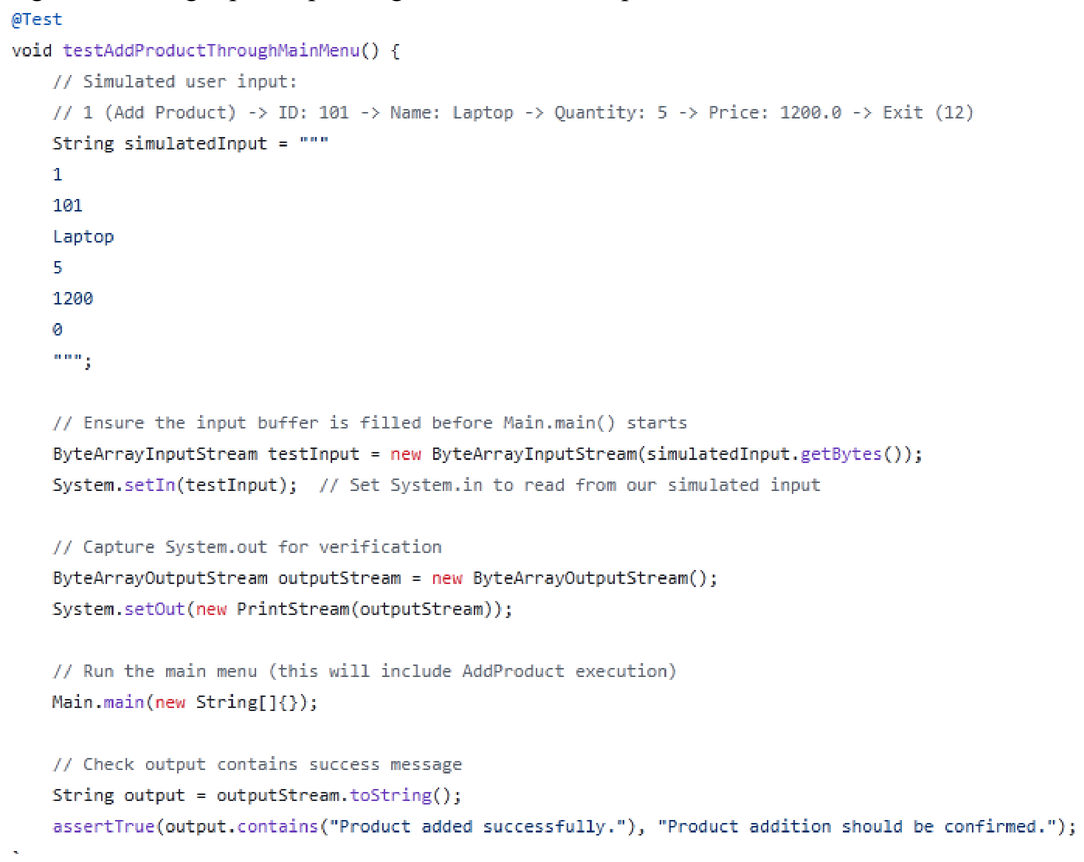
Jaakko Lipponen

1. Unit tests
   Three unit tests were performed on the inventory project, these being testAddProduct, testRemoveProduct, and testFindProductById. As these are core to the functionality and highest up in functional requirements. The test were placed in a tests folder in its own class containing all the unit tests. Linked here.

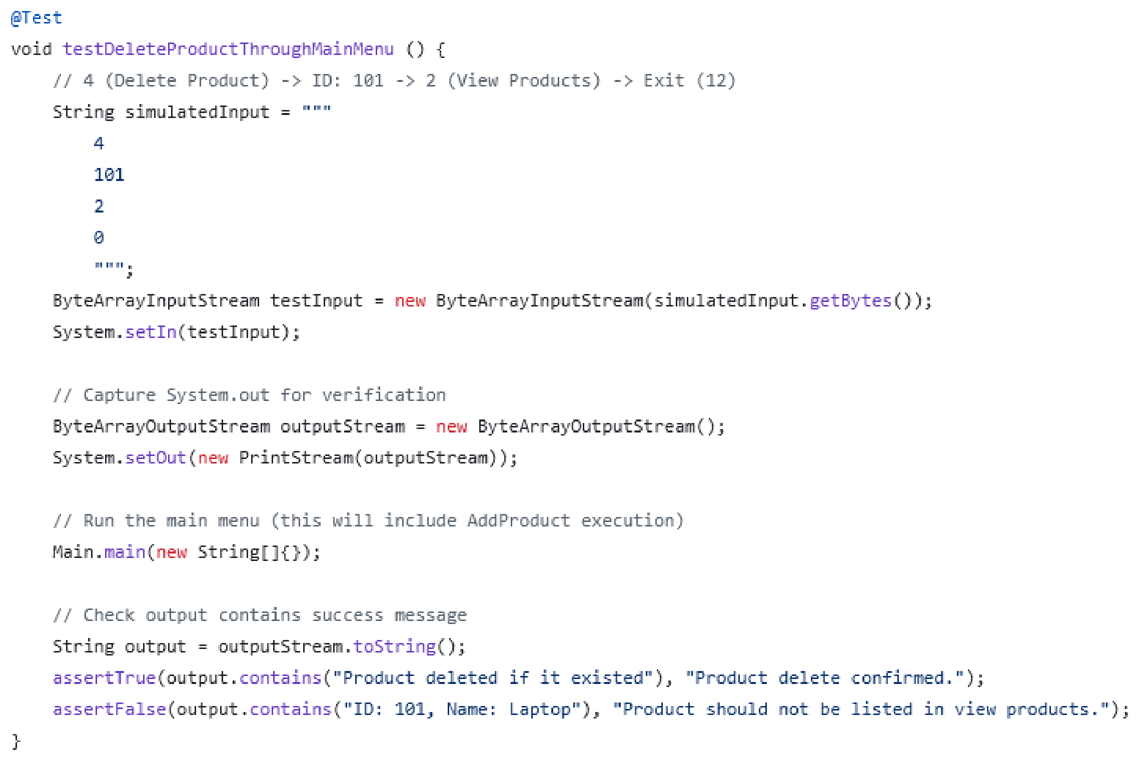


```java

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import java.util.List;

class InventoryManagerTest {

    //unit tests
    private InventoryManager inventory;

    @BeforeEach
    void setUp() {
        inventory = new InventoryManager();
    }

    @Test
    void testAddProduct() {
        Product product = new Product(1, "Laptop", 10, 1500.0);
        inventory.addProduct(product);
        List<Product> products = inventory.getInventory();
        assertTrue(products.contains(product), "Product should be in inventory.");
    }

    @Test
    void testRemoveProduct() {
        Product product = new Product(2, "Mouse", 5, 25.0);
        inventory.addProduct(product);
        int size1 = inventory.getInventory().size();
        inventory.removeProduct(2);
        int size2 = inventory.getInventory().size();
        boolean removed = false;
        if (size1 > size2){
            removed = true;
        }
        assertTrue(removed, "Product should be removed successfully.");
        assertFalse(inventory.getInventory().contains(product), "Product should no longer be in inventory.");
    }

    @Test
    void testFindProductById() {
        Product product = new Product(3, "Keyboard", 3, 45.0);
        inventory.addProduct(product);
        Product retrieved = inventory.findProductById(3);
        assertNotNull(retrieved, "Product should be found.");
        assertEquals("Keyboard", retrieved.getName(), "Product name should match.");
    }
}
```

2. Integration test
   Two integration tests named DeleteIntegrationTests.java and AddIntegrationTests.java were created in the tests folder. These run the main class menu and replace the system.in with a pre-defined integration testing input responding to desired menu options.



```
@Test
void testAddProductThroughMainMenu() {
    // Simulated user input:
    // 1 (Add Product) -> ID: 101 -> Name: Laptop -> Quantity: 5 -> Price: 1200.0 -> Exit (12)
    String simulatedInput = """
    1
    101
    Laptop
    5
    1200
    0
    """;

    // Ensure the input buffer is filled before Main.main() starts
    ByteArrayInputStream testInput = new ByteArrayInputStream(simulatedInput.getBytes());
    System.setIn(testInput);  // Set System.in to read from our simulated input

    // Capture System.out for verification
    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    System.setOut(new PrintStream(outputStream));

    // Run the main menu (this will include AddProduct execution)
    Main.main(new String[]{});

    // Check output contains success message
    String output = outputStream.toString();
    assertTrue(output.contains("Product added successfully."), "Product addition should be confirmed.");
`
```



```
@Test
void testDeleteProductThroughMainMenu () {
    // 4 (Delete Product) -> ID: 101 -> 2 (View Products) -> Exit (12)
    String simulatedInput = """
        4
        101
        2
        0
        """;
    ByteArrayInputStream testInput = new ByteArrayInputStream(simulatedInput.getBytes());
    System.setIn(testInput);

    // Capture System.out for verification
    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    System.setOut(new PrintStream(outputStream));

    // Run the main menu (this will include AddProduct execution)
    Main.main(new String[]{});

    // Check output contains success message
    String output = outputStream.toString();
    assertTrue(output.contains("Product deleted if it existed"), "Product delete confirmed.");
    assertFalse(output.contains("ID: 101, Name: Laptop"), "Product should not be listed in view products.");
}
```

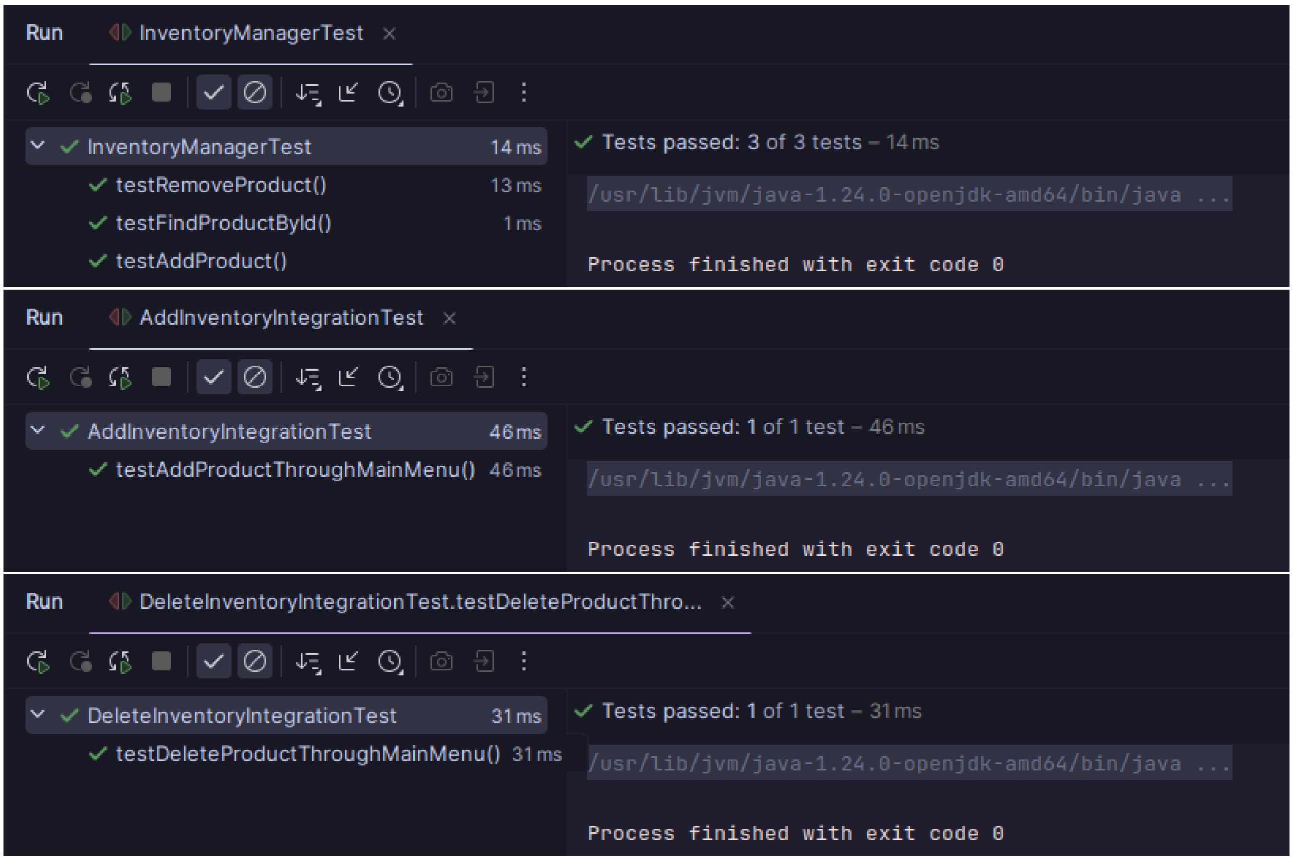
3. Add feature
   Adding a feature while maintaining maximum compatibility required moving the save and exit option from menu item 12 to menu item 0. This makes it easier to append more functions to the

menu in the future, while maintaining compatibility with previous versions. Total inventory value was chosen as the new feature, iterating over the items in the inventory. A total sum is calculated with product.getPrice().

```
while (true) {
    System.out.println("\nInventory Management System");
    System.out.println("0. Save and Exit");
    System.out.println("1. Add Product");
    System.out.println("2. View Products");
    System.out.println("3. Update Product Quantity");
    System.out.println("4. Delete Product");
    System.out.println("5. Search Product");
    System.out.println("6. Generate Inventory Report");
    System.out.println("7. Sort Products by Name");
    System.out.println("8. Sort Products by Price");
    System.out.println("9. Check Low Stock Products");
    System.out.println("10. Filter Products by Price Range");
    System.out.println("11. Clear Inventory");
    System.out.println("12. Total Inventory value");
    System.out.print("Choose an option: ");
    int choice = scanner.nextInt();
```

```
public class TotalValue {

    private final InventoryManager inventoryManager;

    public TotalValue(InventoryManager inventoryManager) {
        this.inventoryManager = inventoryManager;
    }

    public void execute() {
        if (inventoryManager.getInventory().isEmpty()) {
            System.out.println("No products in inventory.");
            return;
        }
        double sum = 0.0;
        for (Product product : inventoryManager.getInventory()) {
            sum += product.getPrice();
        }
        System.out.println("Total value:"+sum);

    }
}
```

4. Regression testing
   After a couple of minor improvements to the menu structure to improve future expandability with new features. Tests and existing functionality worked fine, with a small exception for save and exit which moved to 0 from 12, so old calls will not be compatible. This will, however, vastly improve future expandability and compatibility.



```
Run   InventoryManagerTest ×

InventoryManagerTest              14 ms    ✓ Tests passed: 3 of 3 tests – 14 ms
  ✓ testRemoveProduct()           13 ms    /usr/lib/jvm/java-1.24.0-openjdk-amd64/bin/java ...
  ✓ testFindProductById()          1 ms
  ✓ testAddProduct()                       Process finished with exit code 0
```

```
Run   AddInventoryIntegrationTest ×

AddInventoryIntegrationTest               46 ms    ✓ Tests passed: 1 of 1 test – 46 ms
  ✓ testAddProductThroughMainMenu()       46 ms    /usr/lib/jvm/java-1.24.0-openjdk-amd64/bin/java ...

                                                   Process finished with exit code 0
```

```
Run   DeleteInventoryIntegrationTest.testDeleteProductThro... ×

DeleteInventoryIntegrationTest               31 ms    ✓ Tests passed: 1 of 1 test – 31 ms
  ✓ testDeleteProductThroughMainMenu()       31 ms    /usr/lib/jvm/java-1.24.0-openjdk-amd64/bin/java ...

                                                      Process finished with exit code 0
```

AI usage disclaimer: Not being familiar with Java testing I have brainstormed test ideas and generated example Junit code from which I can apply patterns to these tasks with ChatGPT-4o.