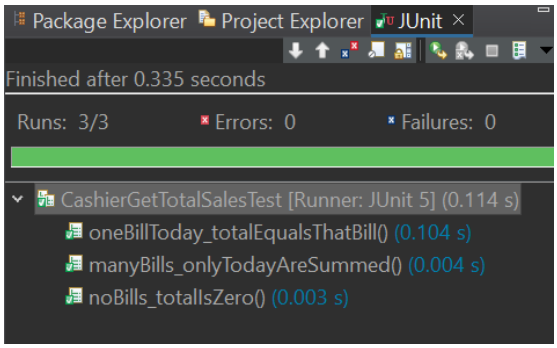
Testing Analysis(Part 2)-Antea Koxherri

1. Boundary Value Testing : Method: Category.checkStockAlert()

Purpose

The purpose of Boundary Value Testing is to verify that the `checkStockAlert()` method behaves correctly at critical stock limits, specifically when the item stock is equal to, below, or above the minimum stock level. Since the method relies on numerical comparison, boundary testing is suitable to ensure correct alert triggering.



Tested Boundary Values & Results:

1)No bills available (lower boundary)

When the cashier has no bills, the method returns a total sales value of **0.0**.
This confirms that the method correctly handles the lower boundary case without errors.

2)Exactly one bill for today

When a single bill exists for the current day, the method returns the total amount of that bill.This confirms that the method correctly handles the minimum non-zero boundary.

3)Multiple bills for today

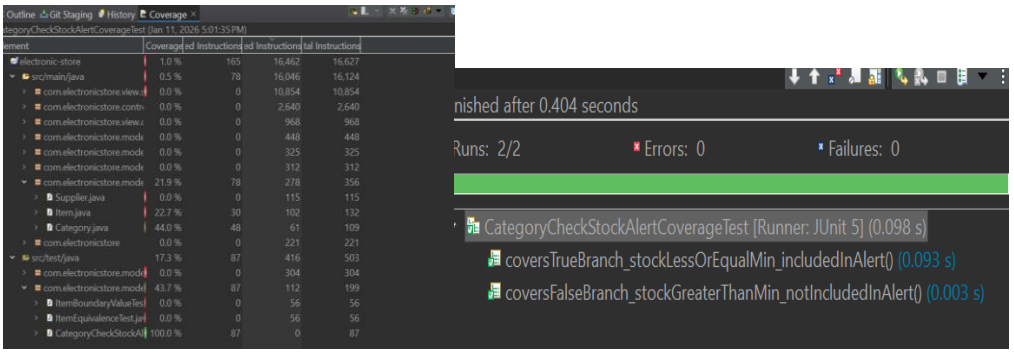
When multiple bills exist for the current day, the method returns the sum of all bill totals. This confirms that the method correctly aggregates values beyond the minimum boundary.

2. Code Coverage Testing – Category.checkStockAlert()

Purpose of testing :The purpose of Code Coverage Testing is to verify that all decision paths in the checkStockAlert() method are executed.
The condition stockQuantity ≤ minStockLevel was tested for both outcomes:

TRUE: item included in the alert list , FALSE: item excluded from the alert list.

Coverage Results : Both branches were executed successfully using two test cases.
The coverage analysis shows:Statement Coverage: **100% ,** Branch Coverage: **100% ,**Condition Coverage: **100%.**




```java
public void addItem(Item item) {
    if (!items.contains(item)) {
        items.add(item);
    }
}

public void removeItem(Item item) {
    items.remove(item);
}

// Business Methods
public List<Item> checkStockAlert() {
    return items.stream()
            .filter(item -> item.getStockQuantity() <= minStockLevel)
            .toList();
}
```

```java
public Category(String id, String name, int minStockLevel, String sector) {
    this.id = id;
    this.name = name;
    this.minStockLevel = minStockLevel;
    this.sector = sector;
    this.items = new ArrayList<>();
}
```

3. Equivalence Class Testing – Cashier.viewDailyBills()

Purpose of Testing : The purpose of this test is to verify that the Cashier.viewDailyBills() method correctly filters bills based on their date and returns consistent results for different input scenarios.

Tested Classes & Results

1) Bills for today:

This test verifies the behavior of the method when bills from the current day exist.

The method returned a list containing only today's bills, confirming correct filtering based on date.

2) Bills from previous days:

This test checks the case where only bills from earlier dates are present.

The method returned an empty list, confirming that outdated bills are correctly excluded.

3) No bills available:

This test verifies the behavior when no bills are present.

The method returned an empty list, confirming safe handling of empty input