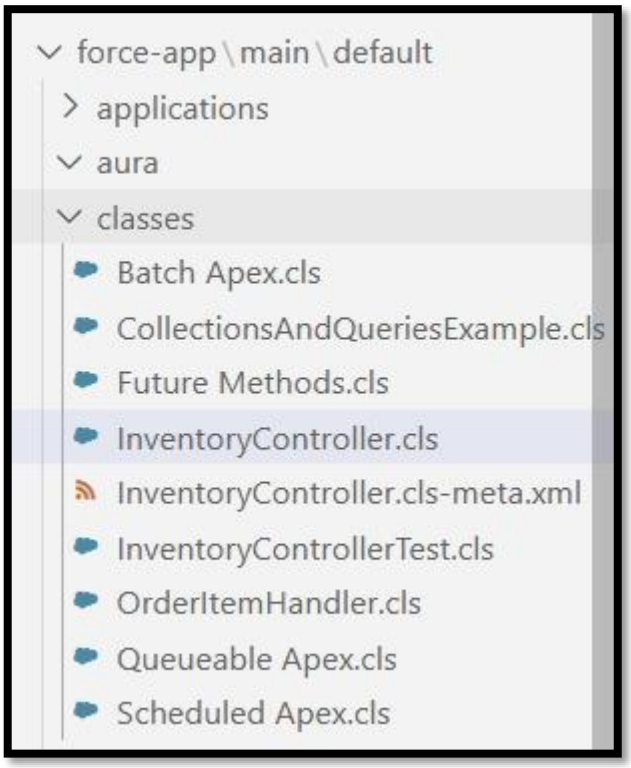
Project Title : “ Inventory & Order Tracking Dashboard “

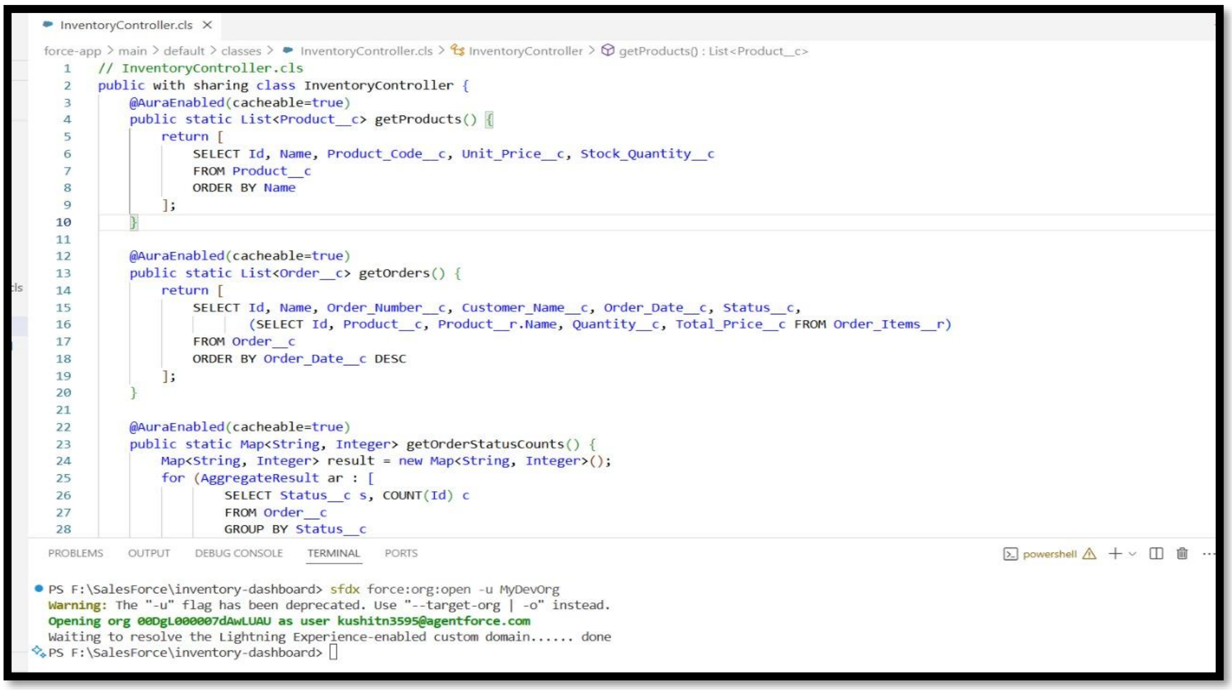
Phase 5: Apex Programming (Developer)

1. Classes & Objects

- What: Apex classes are like Java classes — they group related code (methods & variables).
- created Apex classes



```
force-app \ main \ default
  applications
  aura
  classes
    Batch Apex.cls
    CollectionsAndQueriesExample.cls
    Future Methods.cls
    InventoryController.cls
    InventoryController.cls-meta.xml
    InventoryControllerTest.cls
    OrderItemHandler.cls
    Queueable Apex.cls
    Scheduled Apex.cls
```



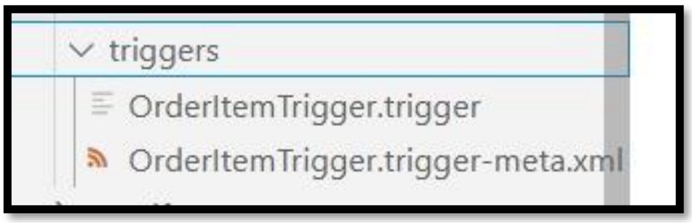
InventoryController.cls
force-app > main > default > classes > InventoryController.cls > InventoryController > getProducts() : List<Product__c>
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
powershell

```
// InventoryController.cls
public with sharing class InventoryController {
    @AuraEnabled(cacheable=true)
    public static List<Product__c> getProducts() {
        return [
            SELECT Id, Name, Product_Code__c, Unit_Price__c, Stock_Quantity__c
            FROM Product__c
            ORDER BY Name
        ];
    }

    @AuraEnabled(cacheable=true)
    public static List<Order__c> getOrders() {
        return [
            SELECT Id, Name, Order_Number__c, Customer_Name__c, Order_Date__c, Status__c,
                   (SELECT Id, Product__c, Product__r.Name, Quantity__c, Total_Price__c FROM Order_Items__r)
            FROM Order__c
            ORDER BY Order_Date__c DESC
        ];
    }

    @AuraEnabled(cacheable=true)
    public static Map<String, Integer> getOrderStatusCounts() {
        Map<String, Integer> result = new Map<String, Integer>();
        for (AggregateResult ar : [
                SELECT Status__c s, COUNT(Id) c
                FROM Order__c
                GROUP BY Status__c
```

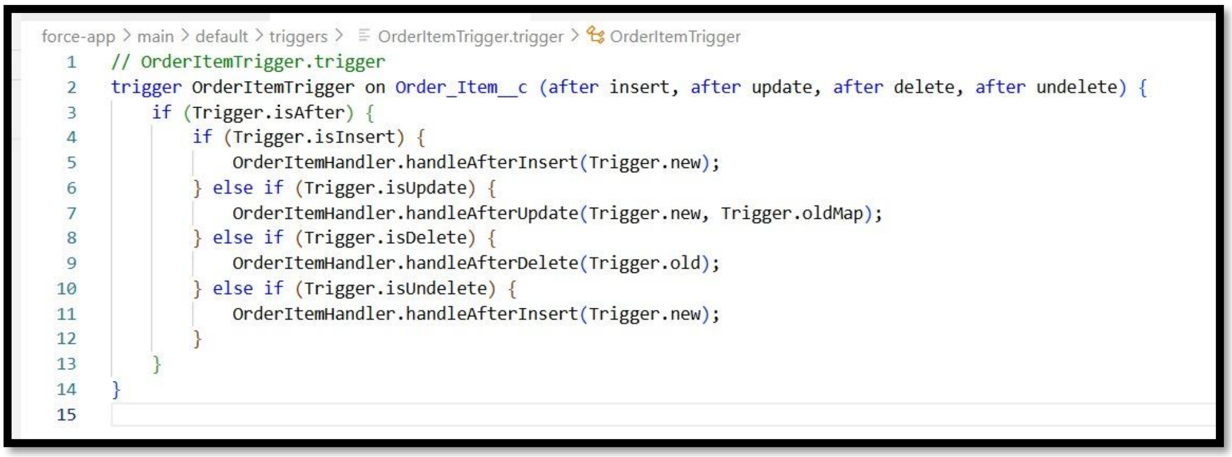
```
PS F:\SalesForce\inventory-dashboard> sfdx force:org:open -u MyDevOrg
Warning: The "-u" flag has been deprecated. Use "--target-org | -o" instead.
Opening org 00DgL000007dAwLUAU as user kushitn3595@agentforce.com
Waiting to resolve the Lightning Experience-enabled custom domain...... done
PS F:\SalesForce\inventory-dashboard>
```

2. **Apex Triggers (before/after insert/update/delete)**

- What: Triggers execute automatically when records are changed.



```
triggers
    OrderItemTrigger.trigger
    OrderItemTrigger.trigger-meta.xml
```

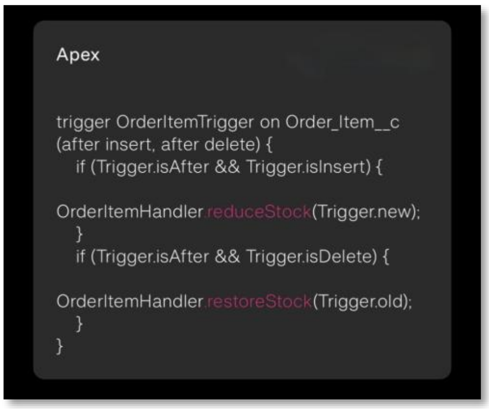


force-app > main > default > triggers > OrderItemTrigger.trigger > OrderItemTrigger

```
// OrderItemTrigger.trigger
trigger OrderItemTrigger on Order_Item__c (after insert, after update, after delete, after undelete) {
    if (Trigger.isAfter) {
        if (Trigger.isInsert) {
            OrderItemHandler.handleAfterInsert(Trigger.new);
        } else if (Trigger.isUpdate) {
            OrderItemHandler.handleAfterUpdate(Trigger.new, Trigger.oldMap);
        } else if (Trigger.isDelete) {
            OrderItemHandler.handleAfterDelete(Trigger.old);
        } else if (Trigger.isUndelete) {
            OrderItemHandler.handleAfterInsert(Trigger.new);
        }
    }
}
```

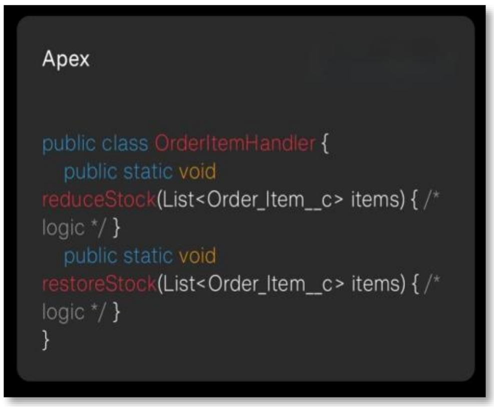
3. **Trigger Design Pattern**

- Why: Avoid writing all logic inside trigger. Use a handler class.
- Created the “OrderItemHandler “ in class folder.



Apex

```
trigger OrderItemTrigger on Order_Item__c
(after insert, after delete) {
    if (Trigger.isAfter && Trigger.isInsert) {
OrderItemHandler.reduceStock(Trigger.new);
    }
    if (Trigger.isAfter && Trigger.isDelete) {
OrderItemHandler.restoreStock(Trigger.old);
    }
}
```



Apex

```
public class OrderItemHandler {
    public static void
reduceStock(List<Order_Item__c> items) { /*
logic */ }
    public static void
restoreStock(List<Order_Item__c> items) { /*
logic */ }
}
```

4. **SOQL & SOSL**

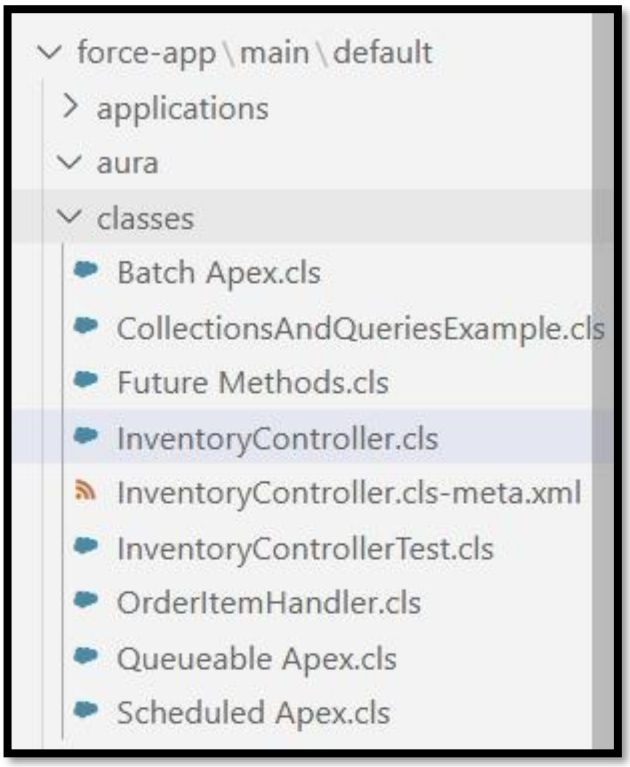
- SOQL: Salesforce Object Query Language → fetch records.

List<Product_c> prods = [SELECT Id, Name, Stock_Quantityc FROM Product_c];

- SOSL: Salesforce Object Search Language → search text across objects.

List<List<SObject>> results = [FIND 'Laptop' IN ALL FIELDS RETURNING Product_c(Name, Stock_Quantity_c)];

“Control Statements”, “Batch Apex”, “Queueable Apex”, “Scheduled Apex”, “FutureMethods” are created and placed in the class folder.



```
force-app \ main \ default
  applications
  aura
  classes
    Batch Apex.cls
    CollectionsAndQueriesExample.cls
    Future Methods.cls
    InventoryController.cls
    InventoryController.cls-meta.xml
    InventoryControllerTest.cls
    OrderItemHandler.cls
    Queueable Apex.cls
    Scheduled Apex.cls
```

5. Asynchronous Processing

- Combine Batch, Queueable, Future, Scheduled depending on need:
- Future → quick async jobs (callouts, email).
- Queueable → chainable async logic.
- Batch → large-scale record processing.
- Scheduled → run jobs on time intervals.

❖ Outcome of Phase 5

- By completing Phase 5, you will have:
- Apex classes for reusable logic.
- Clean triggers with handler pattern.
- SOQL/SOSL queries for data.
- Collections, control structures.
- Asynchronous Apex (Batch, Queueable, Scheduled, Future).
- Test classes for deployment.
- Error handling for safe execution.