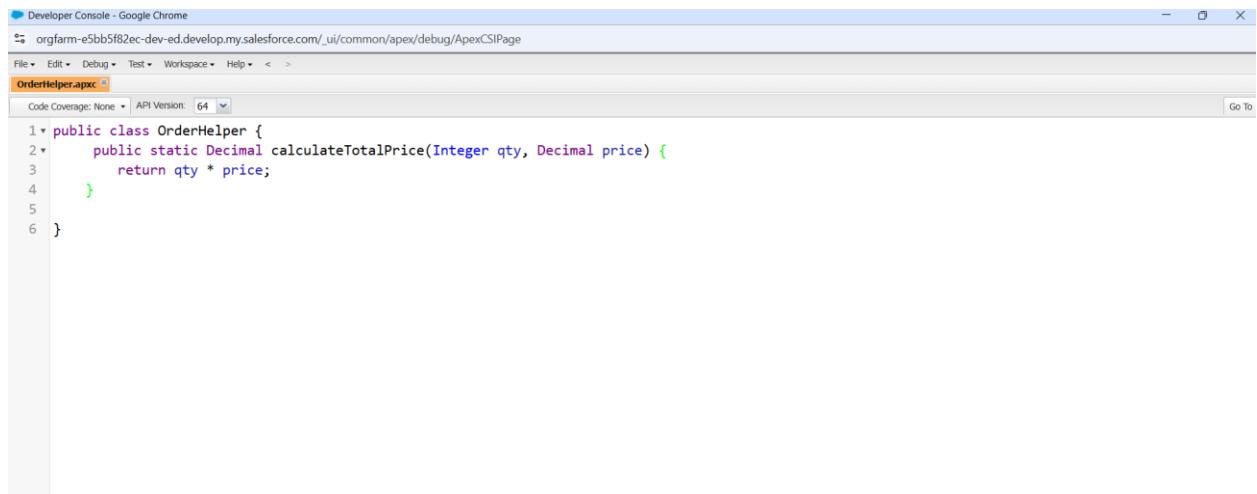


# Phase 5 – Apex Programming (Complete Guide)

This guide explains all Apex concepts required for **Phase 5** of the Farmers Market & Supply Chain Project. It combines Classes, Triggers, SOQL, SOSL, Collections, Control Statements, Batch Apex, Queueable Apex, Scheduled Apex, Future Methods, Exception Handling, and Test Classes.

## ◊ 1) Classes & Objects

- Apex classes group reusable logic and methods.
- 

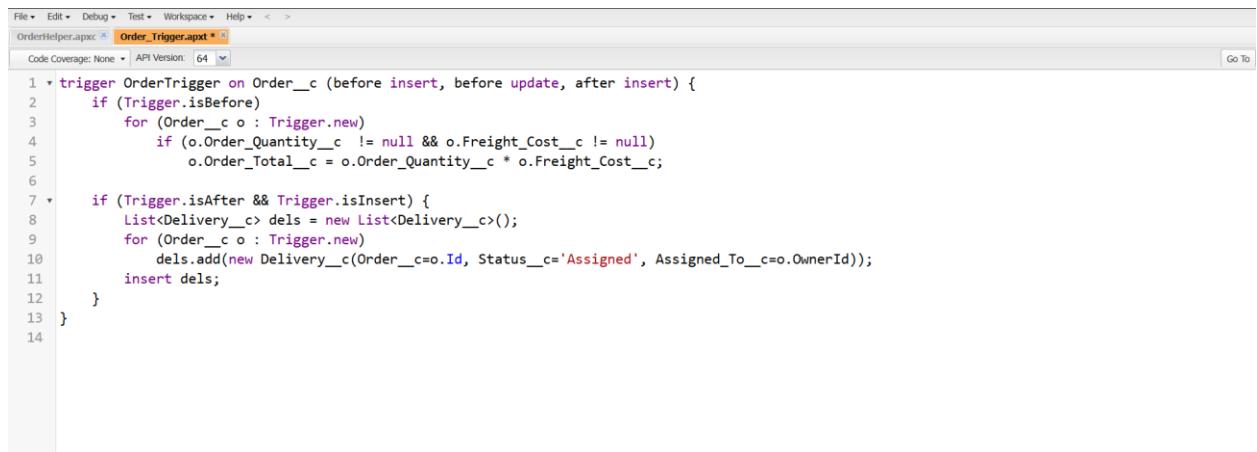


The screenshot shows the Salesforce Developer Console interface. The title bar reads "Developer Console - Google Chrome" and the URL is "orgfarm-e5bb5f82ec-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage". The menu bar includes File, Edit, Debug, Test, Workspace, Help, and a search bar. The tabs at the top show "OrderHelper.apxc" and "Order\_Trigger.apxt". The main area displays the Apex code for the OrderHelper class:

```
1 * public class OrderHelper {  
2     public static Decimal calculateTotalPrice(Integer qty, Decimal price) {  
3         return qty * price;  
4     }  
5 }  
6 }
```

## ◊ 2) Apex Triggers

- Execute logic before/after DML events (insert, update, delete).



The screenshot shows the Salesforce Developer Console interface. The title bar reads "Developer Console - Google Chrome" and the URL is "orgfarm-e5bb5f82ec-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage". The menu bar includes File, Edit, Debug, Test, Workspace, Help, and a search bar. The tabs at the top show "OrderHelper.apxc" and "Order\_Trigger.apxt". The main area displays the Apex trigger code:

```
1 * trigger OrderTrigger on Order_c (before insert, before update, after insert) {  
2     if (Trigger.isBefore)  
3         for (Order_c o : Trigger.new)  
4             if (o.Order_Quantity__c != null && o.Freight_Cost__c != null)  
5                 o.Order_Total__c = o.Order_Quantity__c * o.Freight_Cost__c;  
6  
7     if (Trigger.isAfter && Trigger.isInsert) {  
8         List<Delivery_c> dels = new List<Delivery_c>();  
9         for (Order_c o : Trigger.new)  
10             dels.add(new Delivery_c(Order_c=o.Id, Status_c='Assigned', Assigned_To_c=o.OwnerId));  
11         insert dels;  
12     }  
13 }  
14 }
```

## ◊ 3) SOQL (Salesforce Object Query Language)

- Query data from Salesforce.

### Example:

The screenshot shows the Salesforce IDE interface. In the top-left pane, there is an 'Enter Apex Code' editor containing the following Apex code:

```
1 List<List<SObject>> results =
2     [FIND 'Mango' IN ALL FIELDS
3      RETURNING Order__c(Id, Name), Produce__c(Id, Name), Farmer__c(Id, Name)];
4
5 System.debug('Results: ' + results);
6
```

In the bottom-right pane, there is a 'Logs' tab showing three log entries:

User	Application	Operation	Time	Status	Read	Size
Thirishamahi Bandari	Unknown	/services/data/v64.0/tooling/executeA...	9/25/2025, 2:48:09 PM	Success		2.93 KB
Thirishamahi Bandari	Unknown	/services/data/v64.0/tooling/executeA...	9/25/2025, 2:46:12 PM	Success	Unread	2.91 KB
Thirishamahi Bandari	Unknown	/services/data/v64.0/tooling/executeA...	9/25/2025, 2:45:02 PM	Success	Unread	2.98 KB

The screenshot shows the Salesforce IDE interface. In the top-left pane, there is a 'Query Editor' tab with the following query:

```
SELECT Id, Name FROM Produce__c WHERE Name = 'Mango'
```

In the bottom-right pane, there is a 'Query Results - Total Rows: 1' grid showing one row of data:

Id	Name
a02gk000000SLuxBQAS	Mango

The screenshot shows the Salesforce Query Editor interface. In the top-left pane, there is a 'Query Editor' tab with the following query:

```
SELECT Id, Name FROM Produce__c WHERE Name = 'Mango'
```

In the bottom-right pane, there is a 'History' panel showing the executed queries:

```
Executed =
SELECT Id, Name FROM Produce__c WHERE Name = 'Mango'
SELECT Id, Name FROM Produce__c WHERE Name = 'Tomato'
```

## ◊ 4) SOSL (Salesforce Object Search Language)

- Search text across multiple objects.

The screenshot shows the Salesforce Developer Console interface. At the top, there are tabs for OrderHelper.apxc, Order\_Trigger.apxt, Produce\_c@2:17 PM, Produce\_c@2:33 PM, Log executeAnonymous @9/25/2025, 2:48:09 PM, Log executeAnonymous @9/25/2025, 2:53:23 PM, and Log executeAnonymous @9/25/2025, 2:57:27 PM. Below the tabs, the 'Execution Log' section displays a table with columns: Timestamp, Event, and Details. A single entry is shown: 14:57:27:002 USER\_DEBUG [2][DEBUG]List Example: (Tomato, Mango). Below this is the 'Logs' tab, which shows a table of log entries from the user Thirshamahi Bandari. The table includes columns for User, Application, Operation, Time, Status, Read, and Size. The logs show various tooling operations like /services/data/v64.0/tooling/executeA... with success status and sizes ranging from 2.93 KB to 2.94 KB.

## ◊ 5) Collections: List, Set, Map

### List:

The screenshot shows the Salesforce Developer Console with an Apex code editor window titled 'Enter Apex Code'. The code defines a list of Order\_c objects and iterates through them to debug either 'High value order' or 'Normal order' based on their total value. Below the code editor is the 'Logs' tab, which shows a single log entry from the user Thirshamahi Bandari at 15:18:010. The log details a USER\_DEBUG message: [6][DEBUG]Normal order: First Order. The log table has columns: User, Application, Operation, Time, Status, Read, and Size. The entry shows a success status and a size of 3.37 KB.

## Set:

The screenshot shows the Salesforce Developer Console interface. On the left, the Execution Log pane displays a log entry for a USER\_DEBUG log message at 15:05:55:003, which outputs '[2]||DEBUG|My Set: (Onion, Potato, Tomato)'. On the right, the Enter Apex Code pane contains the following Apex code:

```
1 Set<String> vegSet = new Set<String>{'Tomato', 'Potato', 'Tomato', 'Onion'};
2 System.debug('My Set: ' + vegSet);
3
```

Below the code editor is a toolbar with buttons for Open Log, Execute, and Execute Highlighted. At the bottom, there is a logs tab and a table showing a single log entry from Thrishamahi Bandari.

User	Application	Operation	Time	Status	Read	Size
Thrishamahi Bandari	Unknown	/services/data/v64.0/tooling/executeA...	9/25/2025, 3:05:55 PM	Success		2.46 KB

## Map:

The screenshot shows the Salesforce Developer Console interface. On the left, the Execution Log pane displays a log entry for a USER\_DEBUG log message at 15:08:08:002, which outputs '[7]||DEBUG|My Map: {O=Onion, P=Potato, T=Tomato}'. On the right, the Enter Apex Code pane contains the following Apex code:

```
1 Map<String, String> vegMap = new Map<String, String>();
2 vegMap.put('T', 'Tomato');
3 vegMap.put('P', 'Potato');
4 vegMap.put('O', 'Onion');
5
6 // Print the whole map
7 System.debug('My Map: ' + vegMap);
8
9 // Print keys
10 System.debug('Keys: ' + vegMap.keySet());
11
12 // Print values
13 System.debug('Values: ' + vegMap.values());
14
15 // Get a specific value
16 System.debug('Veg with key T: ' + vegMap.get('T'));
17
18
```

Below the code editor is a toolbar with buttons for Open Log, Execute, and Execute Highlighted. At the bottom, there is a logs tab and a table showing a single log entry from Thrishamahi Bandari.

User	Application	Operation	Time	Status	Read	Size
Thrishamahi Bandari	Unknown	/services/data/v64.0/tooling/executeA...	9/25/2025, 3:08:08 PM	Success		2.46 KB

## ◊ 6) Control Statements

- Core logic flow: if, else, for, while.

### Example:

```
for (Order__c o : [SELECT Id, Order_Total__c FROM Order__c]) {
    if (o.Order_Total__c > 1000) {
        System.debug('High value order: ' + o.Name);
    }
}
```

## ◊ 7) Batch Apex

- Handles **large volumes of records** in chunks.
- Useful for stock updates or recalculations in the Farmers Market project.

## ◊ 8) Queueable Apex

- For **asynchronous background processing**.
- Similar to Batch but simpler, supports job chaining.
- **Use Case:** Send order details to logistics system.

## ◊ 9) Scheduled Apex

- Runs code at specific times (daily, weekly, etc.).
- **Use Case:** Nightly job to notify farmers of low stock.
- Configurable via **Setup → Scheduled Jobs**.

## ◊ 10) Future Methods

- Lightweight async operations, annotated with `@future`.
- Good for callouts or quick notifications.
- **Use Case:** Notify farmer by email after order confirmation.

## ◊ 11) Exception Handling

- Apex supports try-catch-finally.
- Prevents entire transactions from failing.
- **Use Case:** Catch errors when updating Produce stock.

## ◊ 12) Test Classes

- Required for deployment ( $\geq 75\%$  coverage).
- Written with `@isTest`.
- Validate outcomes using `System.assert()`.
- **Use Case:** Ensure Order trigger calculates totals correctly and creates Delivery records.

**Example:**

```
@isTest
public class OrderTriggerTest {
    @isTest static void testOrderTotal() {
        Order__c o = new Order__c(Name='Test Order', Order_Quantity__c=10,
Unit_Price__c=5, Freight_Cost__c=5);
        insert o;
        System.assertEquals(55, o.Order_Total__c);
    }
}
```