



Phase 5: Apex Programming (Developer)

1. Classes & Objects

Developed:

- CarbonEmissionUtils: Utility class for unit conversions.
- CarbonEmissionTriggerHandler: Handler class for trigger logic.
- OldEmissionsBatch: Batch class for asynchronous processing.

Verification: Setup → Apex Classes → All three classes present.

A screenshot of the Salesforce Apex Classes page. The top navigation bar shows "SETUP" and "Apex Classes". Below the header, there's a message box stating "Percent of Apex Used: 0.04%" and "You are currently using 2,318 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of an allowed limit of 6,000,000 characters. Note that the amount in use includes both Apex Classes and Triggers defined in your organization." A button "Estimate your organization's code coverage" is visible. The main area shows a table of Apex classes with columns: Action, Name, Namespace Prefix, Api Version, Status, Size Without Comments, Last Modified By, and Has Trace Flags. The classes listed are: CarbonEmissionTriggerHandler, CarbonEmissionTriggerHandler_Test, CarbonEmissionUtils, CarbonEmissionUtils_Test, and OldEmissionsBatch. All classes are in Active status, created by Vraddhi Valecha on 9/19/2025.

A screenshot of the Salesforce code editor showing the code for the OldEmissionsBatch class. The code implements the Database.Batchable<SObject> interface. It starts by querying all records where CreatedDate is less than LAST_N_DAYS:30, Status_c is not 'Approved', and Amount_c is greater than 1000. It then iterates over each record, setting High_Emission_Flag_c to true, and updates the scope. Finally, it performs post-processing logic like sending an email notification. The code editor shows syntax highlighting for Java-like keywords and Salesforce-specific annotations.

2. Apex Triggers & 3. Trigger Design Pattern

Developed: Trigger on Carbon_Emission__c for after insert, after update. Logic delegated to handler class.

Verification: Setup → Apex Triggers → CarbonEmissionTrigger contains single handler call.

The screenshot shows the 'Apex Triggers' page in the Salesforce Setup. At the top, there's a message box stating 'Percent of Apex Used: 0.04%' and 'You are currently using 2,318 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of an allowed limit of 6,000,000 characters. Note that the amount in use includes both Apex Classes and Triggers defined in your organization.' Below this, there are buttons for 'Compile all triggers' and 'View' (with options 'All', 'Create New View'). A navigation bar at the bottom includes links for A through Z and 'Other'. A table lists the trigger details:

Action	Name ↑	Namespace Prefix	sObject Type	Api Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit Del	CarbonEmissionTrigger		Carbon_Emission	64.0	Active	72	Vraddhi Valecha, 9/19/2025, 12:12 PM	<input type="checkbox"/>

The screenshot shows the 'Apex Trigger Detail' page for 'CarbonEmissionTrigger'. It displays basic information like Name, sObject Type, Status, and Last Modified By. Below this, there are tabs for 'Apex Trigger', 'Version Settings', and 'Trace Flags'. The 'Apex Trigger' tab shows the trigger code:

```
1 trigger CarbonEmissionTrigger on Carbon_Emission__c (before insert) {  
2  
3}
```

At the bottom, there are buttons for 'Edit', 'Delete', 'Download', and 'Show Dependencies'.

The screenshot shows the Salesforce IDE with the 'CarbonEmissionTrigger.apxc' file open. The code editor displays the following trigger definition:

```
1 trigger CarbonEmissionTrigger on Carbon_Emission__c (after insert, after update) {  
2     // This trigger has no logic. It only calls a handler class.  
3     CarbonEmissionTriggerHandler.handleAfterInsertUpdate(Trigger.new);  
4 }
```

4. SOQL, 6. Control Statements, 7. Collections

Implemented:

- SOQL: Queries in batch class and tests.
- Collections: List<Carbon_Emission__c> to manage records.
- Control Statements: if for checks, for loops for processing.

Verification: Code in CarbonEmissionTriggerHandler and OldEmissionsBatch.

Apex Class
CarbonEmissionTriggerHandler

< Back to List

Apex Class Detail

Name	CarbonEmissionTriggerHandler	Status	Active
Namespace Prefix		Code Coverage	0% (0/16)
Created By	Vraddhi Valecha , 9/19/2025, 12:12 PM	Last Modified By	Vraddhi Valecha , 9/19/2025, 12:13 PM

Class Body

```
1 public class CarbonEmissionTriggerHandler {
2     public static void handleAfterInsertUpdate(List<Carbon_Emission__c> newRecords) {
3         // List to hold records that qualify for approval
4         List<Carbon_Emission__c> recordsForApproval = new List<Carbon_Emission__c>();
5
6         // Loop through the triggered records and check the criteria
7         for (Carbon_Emission__c record : newRecords) {
8             if (record.Amount__c > 1000 && record.Status__c != 'Approved') {
9                 recordsForApproval.add(record);
10            }
11        }
12
13        // If we found any qualifying records, process them for approval
14        if (!recordsForApproval.isEmpty()) {
15            submitRecordsForApproval(recordsForApproval);
16        }
17    }
18
19
20    private static void submitRecordsForApproval(List<Carbon_Emission__c> records) {
21        // Create a list of approval requests
22        List<Approval.ProcessSubmitRequest> approvalRequests = new List<Approval.ProcessSubmitRequest>();
23
24        for (Carbon_Emission__c record : records) {
25            Approval.ProcessSubmitRequest req = new Approval.ProcessSubmitRequest();
26            req.setObjectId(record.Id); // Point the request to the specific record
27            approvalRequests.add(req);
28        }
29
30        // Submit all approval requests
31        try {
32            List<Approval.ProcessResult> results = Approval.process(approvalRequests);
33            // You can loop through 'results' to check if each was successful
34            } catch (Exception e) {
35                // Exception handling: Log the error so an admin can see it
36                System.debug('Approval submission failed: ' + e.getMessage());
37            }
38        }
39    }
```

```
// Query locator to gather all the records to process
global Database.QueryLocator start(Database.BatchableContext bc) {
    String query = 'SELECT Id, Name, High_Emission_Flag__c FROM Carbon_Emission__c ' +
        'WHERE CreatedDate < LAST_N_DAYS:30 ' +
        'AND Status__c != \'Approved\' ' +
        'AND Amount__c > 1000';
    return Database.getQueryLocator(query);
}
```

```
// Loop through the triggered records and check the criteria
for (Carbon_Emission__c record : newRecords) {
    if (record.Amount__c > 1000 && record.Status__c != 'Approved') {
        recordsForApproval.add(record);
    }
}
```

```
// List to hold records that qualify for approval
List<Carbon_Emission__c> recordsForApproval = new List<Carbon_Emission__c>();
```

5. Batch Apex

Developed: OldEmissionsBatch to flag old unapproved records.

Verification: Setup → Apex Jobs → Completed job for OldEmissionsBatch

[Click here to go back to the classic view](#)

Apex Batch Jobs

Monitor the status of all Apex Batch jobs.

09/12/2025, 01:17 AM - 09/20/2025, 01:17 AM

Action	Class Id	Namespace Prefix	Class Name	Queued	Processing	Aborted
More info	01pgL000005ebHJ		OldEmissionsBatch	0	0	0

SETUP Apex Jobs

[Click here to go to the new batch jobs page](#) [Help for this Page](#)

Apex Jobs

Monitor the status of all Apex jobs, and optionally, abort jobs that are in progress.

Percent of Asynchronous Apex Used: 0% You have currently used 0 asynchronous Apex operations out of an allowed 24-hour organization limit of 250,000. To learn about how this limit is calculated and what contributes to it, see the [Lightning Platform Apex Limits](#) topic.

Action	Submitted Date	Job Type	Status	Status Detail	Total Batches	Batches Processed	Failures	Submitted By	Completion Date	Apex Class	Apex Method	Apex Job ID
All	9/19/2025, 12:26 PM	Batch Apex	Completed		0	0	0	Valecha_Vradhhi	9/19/2025, 12:26 PM	OldEmissionsBatch		707gL00000ENc5W

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

CarbonEmissionUtils.apxc CarbonEmissionUtils_Test.apxc CarbonEmissionTrigger.apxt * CarbonEmissionTriggerHandler.apxc CarbonEmissionTriggerHandler_Test.apxc OldEmissionsBatch.apxc

Code Coverage: None API Version: 64

```
1 global class OldEmissionsBatch implements Database.Batchable<SObject> {
2
3     // Query locator to gather all the records to process
4     global Database.QueryLocator start(Database.BatchableContext bc) {
5         String query = 'SELECT Id, Name, High_Emission__c FROM Carbon_Emission__c ' +
6             'WHERE CreatedDate < LAST_N_DAYS:30 ' +
7             'AND Status__c != \'Approved\' ' +
8             'AND Amount__c > 1000';
9         return Database.getQueryLocator(query);
10    }
11
12    // The actual processing logic for each "chunk" of data
13    global void execute(Database.BatchableContext bc, List<Carbon_Emission__c> scope) {
14        for (Carbon_Emission__c record : scope) {
15            record.High_Emission__c = true; // Flag the record
16        }
17        update scope; // Update all records in this batch
18    }
19
20    // Post-processing logic (e.g., send an email)
21    global void finish(Database.BatchableContext bc) {
22        // You can send an email here to notify an admin the job is done.
23        // Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
24        // ... (email setup code) ...
25        // Messaging.sendEmail(new List<Messaging.SingleEmailMessage>{mail});
26    }
27 }
```

Logs Tests Checkpoints Query Editor View State Progress Problems

Name	Line	Problem
------	------	---------

8. Test Classes

Developed:

- Test classes for utilities and triggers (>75% coverage).

Verification:

- Code in CarbonEmissionUtils_Test and CarbonEmissionTriggerHandler_Test.
- Setup → Apex Test Execution → Successful test runs.

The screenshot shows the Salesforce Apex Classes page for the CarbonEmissionUtils_Test class. The page has a header with 'SETUP' and 'Apex Classes'. Below the header, the class name 'CarbonEmissionUtils_Test' is displayed along with its details: Name (CarbonEmissionUtils_Test), Namespace Prefix (empty), Status (Active), and Created By (Vraddhi Valecha, 9/19/2025, 12:06 PM). The last modified by field also shows Vraddhi Valecha, 9/19/2025, 12:06 PM. There are tabs for Class Body, Class Summary, Version Settings, and Trace Flags, with 'Class Body' selected. The code editor contains the following test code:

```
1 @isTest
2 private class CarbonEmissionUtils_Test {
3
4     @isTest
5     static void testConvertTonsToKg() {
6         // Test the conversion from tons to kg
7         Decimal result = CarbonEmissionUtils.convertToKilograms(5, 'tons');
8         System.assertEquals(5000, result, '5 tons should equal 5000 kg');
9     }
10
11    @isTest
12    static void testConvertPoundsToKg() {
13        // Test the conversion from pounds to kg
14        Decimal result = CarbonEmissionUtils.convertToKilograms(100, 'pounds');
15        System.assertEquals(45.3592, result, '100 pounds should equal ~45.3592 kg');
16    }
17
18    @isTest
19    static void testNoConversion() {
20        // Test that kg is passed through unchanged
21        Decimal result = CarbonEmissionUtils.convertToKilograms(250, 'kg');
22        System.assertEquals(250, result, '250 kg should remain 250 kg');
23    }
24 }
```

Below the code editor are standard edit, delete, download, run test, and show dependencies buttons.

The screenshot shows the Salesforce Apex Classes page for the CarbonEmissionTriggerHandler_Test class. The page has a header with 'SETUP' and 'Apex Classes'. Below the header, the class name 'CarbonEmissionTriggerHandler_Test' is displayed along with its details: Name (CarbonEmissionTriggerHandler_Test), Namespace Prefix (empty), Status (Active), and Created By (Vraddhi Valecha, 9/19/2025, 12:13 PM). The last modified by field also shows Vraddhi Valecha, 9/19/2025, 12:19 PM. There are tabs for Class Body, Class Summary, Version Settings, and Trace Flags, with 'Class Body' selected. The code editor contains the following test code:

```
1 @isTest
2 private class CarbonEmissionTriggerHandler_Test {
3
4     @isTest
5     static void testTriggerSubmitsRecordForApproval() {
6         Emission__Source__c testSource = new Emission__Source__c(
7             Name = 'Test Air Travel Source'
8         );
9         insert testSource;
10
11        // 1. CREATE TEST DATA
12        Carbon_Emission__c testEmission = new Carbon_Emission__c(
13            Name = 'Test Private Jet',
14            Amount__c = 15000, // Definitely over 1000!
15            Status__c = 'New',
16            Source__c = testSource.Id
17        );
18
19        // 2. PERFORM TEST (Inserting the record will fire the trigger)
20        // Start test()
21        Test.startTest();
22        insert testEmission;
23        Test.stopTest(); // Stop the test context
24
25        // 3. VERIFY RESULTS (Query the Approval Process to see if it was created)
26        // Query for ProcessInstance (represents an approval process)
27        List<ProcessInstance> approvalProcesses = [
28            SELECT Id, Status, TargetObjectId
29            FROM ProcessInstance
30            WHERE TargetObjectId = :testEmission.Id
31        ];
32
33        // ASSERT: Check that exactly one approval process was started
34        System.assertEquals(1, approvalProcesses.size(), 'An approval process should have been started for the high-value emission.');
35    }
36 }
```

The screenshot shows the Apex Test Results page in the Salesforce Setup. At the top, there's a setup icon and the title "Apex Test Execution". Below the title, it says "Apex Test Results". A link "Help for this Page" with a question mark icon is also present. The main area displays a table of test results:

Ac...	Name ↑	Method Name	Time Started	Pass/Fail	Error Message	Run Time
View	CarbonEmissionTrig...	testTriggerSubmitsR...	9/19/2025, 12:19 PM	Fail	System.DmlExcepti...	163
View	CarbonEmissionUtil...	testConvertPoundsT...	9/19/2025, 12:10 PM	Pass		31
View	CarbonEmissionUtil...	testConvertTonsToKg	9/19/2025, 12:10 PM	Pass		7
View	CarbonEmissionUtil...	testNoConversion	9/19/2025, 12:10 PM	Pass		6

9. Exception Handling

Developed:

- try-catch block in handler for error handling.

Verification:

- try-catch in handler code.

```
29
30     // Submit all approval requests
31     try {
32         List<Approval.ProcessResult> results = Approval.process(approvalRequests);
33         // You can loop through 'results' to check if each was successful
34     } catch (Exception e) {
35         // Exception handling: Log the error so an admin can see it
36         System.debug('Approval submission failed: ' + e.getMessage());
37     }
38 }
39 }
```