

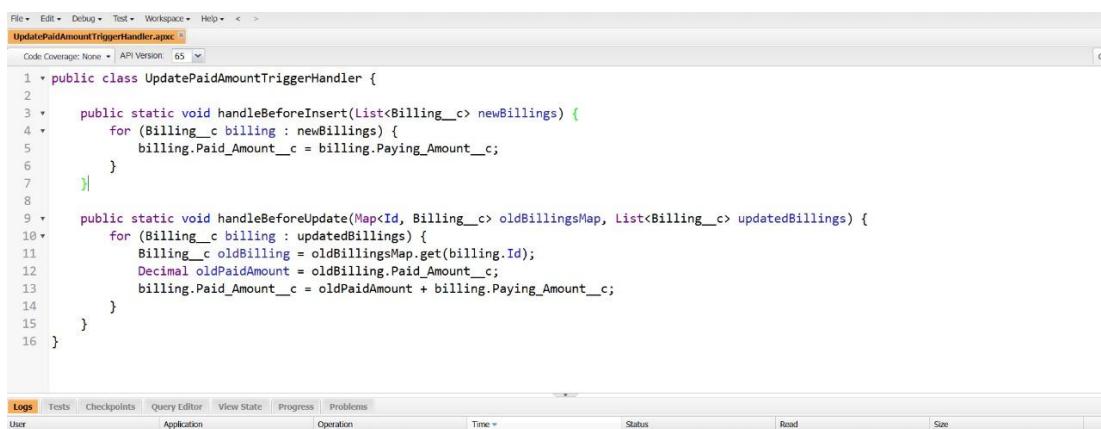
## APEX TRIGGERS AND CLASSES

<b>Date</b>	05 November 2025
<b>Team Id</b>	NM2025TMID04354
<b>Project Name</b>	CRM Application for Jewel Management

Where declarative automation cannot handle complex conditions, Apex Triggers and Apex Classes were developed.

### Examples:

- **Trigger on Order:** Automatically calculate and update total order value based on selected jewelry items.
- **Trigger on Payment:** Update the Payment Status field in the related Order record when full payment is received.
- **Trigger on Jewelry Item:** Automatically change item status to “Out of Stock” when stock quantity reaches zero.



```
File -> Edit -> Debug -> Test -> Workspace -> Help -> <- >
UpdatePaidAmountTriggerHandler.apc
Code Coverage: None | API Version: 65

1 public class UpdatePaidAmountTriggerHandler {
2
3     public static void handleBeforeInsert(List<Billing__c> newBillings) {
4         for (Billing__c billing : newBillings) {
5             billing.Paid_Amount__c = billing.Paying_Amount__c;
6         }
7     }
8
9     public static void handleBeforeUpdate(Map<Id, Billing__c> oldBillingsMap, List<Billing__c> updatedBillings) {
10        for (Billing__c billing : updatedBillings) {
11            Billing__c oldBilling = oldBillingsMap.get(billing.Id);
12            Decimal oldPaidAmount = oldBilling.Paid_Amount__c;
13            billing.Paid_Amount__c = oldPaidAmount + billing.Paying_Amount__c;
14        }
15    }
16 }
```

The screenshot shows the Salesforce Developer Console interface. The top navigation bar includes File, Edit, Debug, Test, Workspace, Help, and a back/forward button. Below the bar, the file name "UpdatePaidAmountTriggerHandler.apc" is displayed in orange, indicating it is the active file. The code editor contains the provided Apex class code. At the bottom of the screen, there is a toolbar with tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Logs tab is currently selected. Below the toolbar, there is a user interface for monitoring logs, showing columns for User, Application, Operation, Time, Status, Read, and Size.

The screenshot shows the Salesforce IDE interface. At the top, there's a menu bar with options like File, Edit, Debug, Test, Workspace, Help, and a workspace switcher. Below the menu is a toolbar with icons for Save, Undo, Redo, and others. The main area is a code editor with syntax highlighting for Apex. The code is a trigger named 'UpdatePaidAmountTrigger' on the 'Billing\_\_c' object, performing actions before insert and update. The code editor has a status bar at the bottom showing 'Code Coverage: None', 'API Version: 65', and a dropdown for 'File Encoding'. Below the code editor is a navigation bar with tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The 'Logs' tab is currently selected. The logs section is empty, showing a table with columns for User, Application, Operation, Time, and Status.

```
trigger UpdatePaidAmountTrigger on Billing__c (before insert, before update) {
    if (Trigger.isInsert) {
        UpdatePaidAmountTriggerHandler.handleBeforeInsert(Trigger.new);
    }
    else if (Trigger.isUpdate) {
        UpdatePaidAmountTriggerHandler.handleBeforeUpdate(Trigger.oldMap, Trigger.new);
    }
}
```

## Apex Classes were used to implement backend logic for:

- Generating invoices.
- Sending scheduled payment reminders.
- Running daily maintenance batch jobs.

This custom logic provides the system with flexibility and enhances automation accuracy.