

---

## Phase 5 – Apex Programming

---

### PROJECT:- DEEPCODE CRM

---

#### 1. Apex Classes & Objects

**Definition:** Apex Classes are templates that define **custom business logic**. Objects are Salesforce database tables, and fields are columns.

#### Key Points:

- Classes contain **methods** (functions) and **variables**
- Encapsulate logic → reusable across triggers, Visualforce, Lightning

#### Video Handler:



The screenshot shows the Salesforce Class Editor interface. At the top, there are four tabs: 'Class Body' (which is selected), 'Class Summary', 'Version Settings', and 'Trace Flags'. Below the tabs, the class code is displayed in a scrollable area:

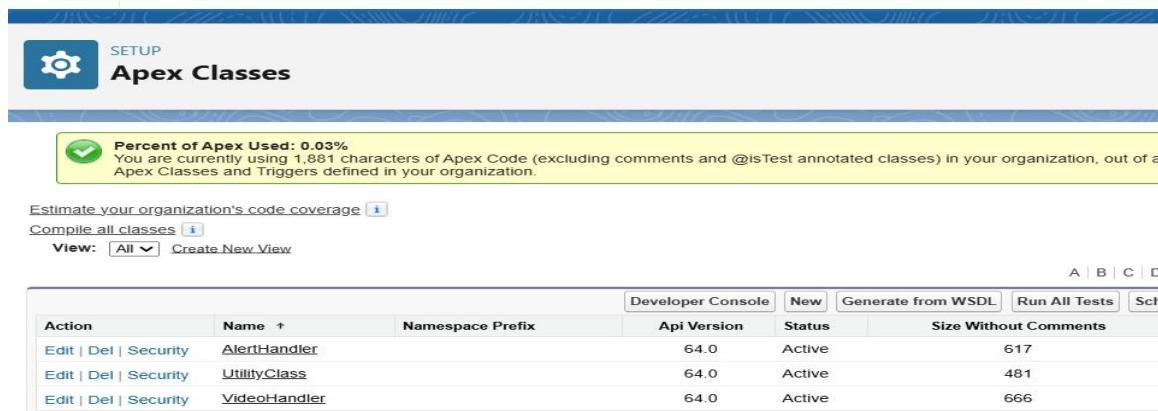
```
1 public class VideoHandler {
2     public static void updateVideoStatus(List<Video__c> videos) {
3         for(Video__c v : videos) {
4             if(String.isNotBlank(v.Confidence_Score__c)) {
5                 Integer score = 0;
6                 try {
7                     score = Integer.valueOf(v.Confidence_Score__c);
8                 } catch (Exception e) {
9                     score = 0; // agar text me number na ho
10                }
11
12                if(score >= 50) {
13                    v.Status__c = 'Approved';
14                } else {
15                    v.Status__c = 'Review';
16                }
17            } else {
18                v.Status__c = 'Review';
19            }
20        }
21    }
}
```

## Alert Handler:



```
1 public class AlertHandler {
2
3     // Method to send alert based on type
4     public static void processAlerts(List<Alert__c> alerts) {
5         for(Alert__c a : alerts) {
6             if(a.Alert_Type__c == 'System') {
7                 a.Status__c = 'Pending Review';
8             } else if(a.Alert_Type__c == 'Video') {
9                 a.Status__c = 'Check Video';
10            } else {
11                a.Status__c = 'User Action Required';
12            }
13        }
14        update alerts;
15    }
16
17    // Fetch alerts by type
18    public static List<Alert__c> getAlertsByType(String alertType) {
19        return [SELECT Id, Name, Alert_Type__c FROM Alert__c WHERE Alert_Type__c = :alertType];
20    }
}
```

## All Apex Classes :



Percent of Apex Used: 0.03%  
You are currently using 1,881 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of a total of 1,881 characters.

Estimate your organization's code coverage [Link](#)  
Compile all classes [Link](#)

View: All [Link](#) Create New View

Action	Name <a href="#">+</a>	Namespace Prefix	Api Version	Status	Size Without Comments
Edit   Del   Security	<a href="#">AlertHandler</a>		64.0	Active	617
Edit   Del   Security	<a href="#">UtilityClass</a>		64.0	Active	481
Edit   Del   Security	<a href="#">VideoHandler</a>		64.0	Active	666

**Tip:** Always use **bulk-safe** logic – process multiple records using lists.

## 2. Apex Triggers

**Definition:** Triggers automate logic **before or after record events** (insert, update, delete).

### Trigger Design Pattern:

- Trigger calls **Handler Class**
- Keeps logic **centralized & maintainable**
- Supports **bulkification** **Video Trigger**

```
trigger VideoTrigger on Video__c (before insert, before update) {
```

```
    VideoHandler.updateVideoStatus(Trigger.new);
```

```
}
```

Namespace Prefix

Apex Trigger Version Settings Trace Flags

```
1 trigger VideoTrigger on Video__c (before insert, before update) {
2   VideoHandler.updateVideoStatus(Trigger.new);
3}
```

Edit Delete Download Show Dependencies

Use **before triggers** to update fields before saving, **after triggers** for related objects.

### 3. SOQL & SOSL

#### SOQL (Salesforce Object Query Language):

- Fetch records from a **single object**
- Can filter, order, and aggregate

```
List<Video__c> approvedVideos = [SELECT Id, Name, Status__c  
                                FROM Video__c  
                                WHERE Status__c = 'Approved'];
```

Logs Tests Checkpoints **Query Editor** View State Progress Problems

```
public class VideoHelper {  
    public static List<Video__c> getApprovedvideos() {  
        return [SELECT Id, Name, Status__c FROM Video__c WHERE Status__c = 'Approved'];  
    }  
}
```

Any query errors will appear here...

#### SOSL (Salesforce Object Search Language):

- Search **text across multiple objects/fields**
- Returns a **list of lists of sObjects**

```
List<List<sObject>> results = [FIND 'Tutorial*' IN ALL FIELDS  
                                RETURNING Video__c(Id, Name, Title__c)];
```

```
List<Video__c> videosFound = (List<Video__c>)results[0];
```

**Tip:** Use SOSL for **quick text search**, SOQL for **specific field filtering**.

---

#### 4. Collections: List, Set, Map

1. **List** – Ordered, allows duplicates

2. **Set** – Unique elements, unordered 3. **Map** – Key-Value pairs, keys

unique Example:

```
List<Video__c> videos = [SELECT Id, Name FROM Video__c];
Set<Id> videoIds = new Set<Id>();
Map<Id, Video__c> videoMap = new Map<Id, Video__c>(videos);
```

Use collections for **bulk-safe triggers** and to **avoid SOQL inside loops**.

---

#### 5. Batch Apex

**Purpose:** Process **large datasets** asynchronously in chunks (batches).

**Structure:**

1. start → Fetch records
2. execute → Process batch
3. finish → Post-processing

#### Video Batch Update

```

global class VideoBatchUpdate implements Database.Batchable<SObject> {

    // 1 Start - Query records
    global Database.QueryLocator start(Database.BatchableContext BC) {
        return Database.getQueryLocator(
            'SELECT Id, Confidence_Score__c, Status__c FROM Video__c'
        );
    }

    // 2 Execute - Process batch
    global void execute(Database.BatchableContext BC, List<Video__c> scope) {
        for(Video__c v : scope) {
            Integer score = 0;
            try {
                score = Integer.valueOf(v.Confidence_Score__c);
            } catch(Exception e){
                score = 0;
            }
            v.Status__c = (score >= 50) ? 'Approved' : 'Review';
        }
    }
}

```

Tests Checkpoints Query Editor View State Progress Problems

**Run:**

```
Database.executeBatch(new VideoBatchUpdate(), 200);
```

**Tip:** Batch Apex is essential for **50,000+ records**.

## 6. Asynchronous Apex

- **Future Methods:** Run in background @future

```
public static void sendEmailAsync(String email) { ... }
```

- **Queueable Apex:** Supports chaining jobs

```
public class QueueableExample implements
```

```
Queueable { ... }
```

- **Scheduled Apex:** Cron-like scheduling global

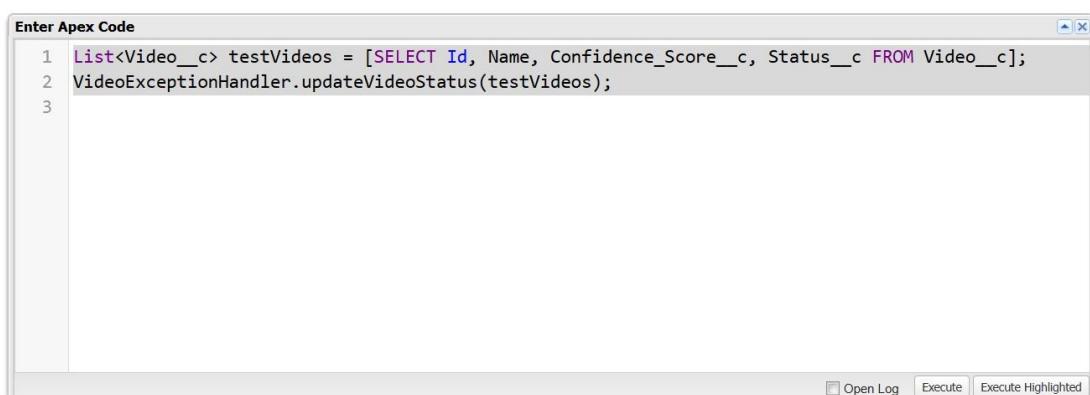
```
class ScheduledJob implements Schedulable { ... }
```

Use async apex for **heavy processing** without hitting governor limits.

## 7. Exception Handling

- Use try-catch to prevent runtime errors

```
public class VideoExceptionHandler {  
    public static void updateVideoStatus(List<Video__c> videos) {  
        for(Video__c v : videos){  
            try {  
                Integer score = Integer.valueOf(v.Confidence_Score__c);  
                v.Status__c = (score >= 50) ? 'Approved' : 'Review';  
            } catch(Exception e){  
                System.debug('Error processing Video: ' + v.Name + ' - ' + e.getMessage());  
                v.Status__c = 'Error';  
            }  
        }  
        update videos;  
    }  
}  
  
trigger VideoTriggerException on Video__c (before insert, before update) {  
    VideoExceptionHandler.updateVideoStatus(Trigger.new);  
}
```



Always log errors for debugging.

## 8. Test Classes

**Purpose:** Validate logic, ensure **code coverage ≥75%**

### Video Handler Test

```

@IsTest
public class VideoHandlerTest {
    static testMethod void testUpdateVideoStatus() {
        Video__c v1 = new Video__c(Name='V1', Confidence_Score__c='60');
        Video__c v2 = new Video__c(Name='V2', Confidence_Score__c='40');

        Test.startTest();
        insert new List<Video__c>{v1, v2};
        Test.stopTest();

        System.assertEquals('Approved', [SELECT Status__c FROM Video__c WHERE Id=:v1.Id].Status__c);
        System.assertEquals('Review', [SELECT Status__c FROM Video__c WHERE Id=:v2.Id].Status__c);
    }
}

```

**Other Test Classes:** Alert Helper Test, Utility Class Test (same format)

Overall Code Coverage		
Class	Percent	Lines
<b>Overall</b>	<b>94%</b>	17/17
VideoHandler	100%	17/17
AlertHandler	77%	14/18
UtilityClass	100%	15/15

## 9. Best Practices

- Always **bulkify triggers**
- Avoid SOQL/DML inside loops
- Use **Handler classes** for logic
- Write **robust test classes** with positive/negative scenarios
- Use **Async Apex** for large or delayed processing