

## **Phase 5: Apex Programming (Developer)**

---

### **1. Classes & Objects**

#### **Theory:**

In Apex, classes are blueprints that define objects. They can contain variables (fields), methods, and constructors. Objects are instances of these classes. This allows Salesforce developers to encapsulate business logic and reuse it across the org.

#### **In my project:**

- JobApplicationController.cls → handles UI logic for job applications.
- JobController.cls → handles job object operations.
- TaxCalculation.cls → encapsulates tax calculation logic.
- EventReminder.cls → defines logic for event reminders.

### **Control Statements**

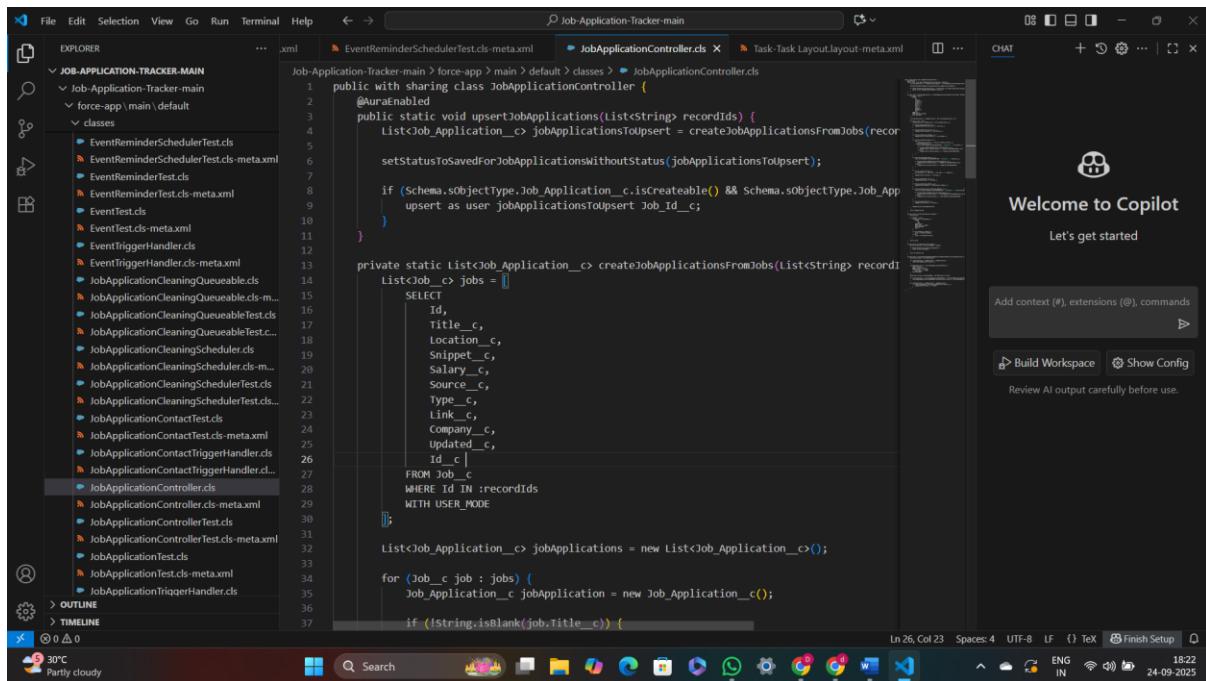
#### **Theory:**

Apex supports control statements like if-else, for, while, switch, which help in conditional and iterative logic.

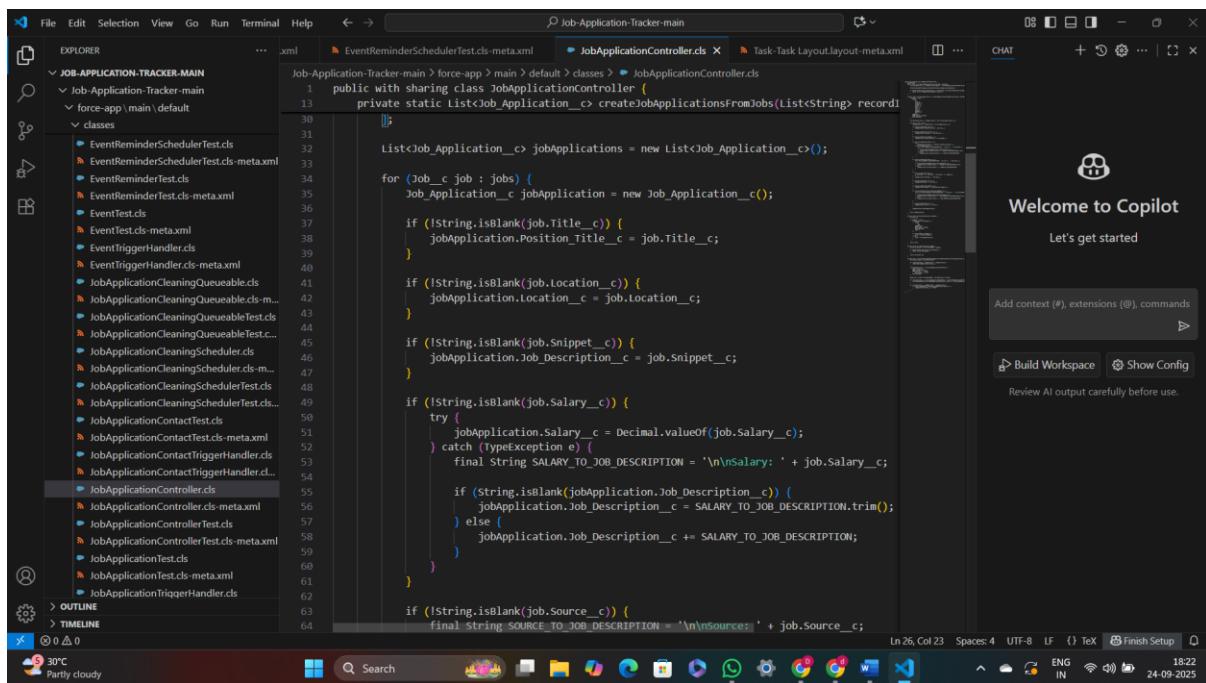
#### **In your project:**

- TaxCalculation.cls → uses if-else to calculate taxes.

## JobApplicationController.cls:



```
1 public with sharing class JobApplicationController {
2     @AuraEnabled
3     public static void upsertJobApplications(List<String> recordIds) {
4         List<Job_Application__c> jobApplicationsToUpser = createJobApplicationsFromJobs(recordIds);
5
6         setStatusToSavedOrJobApplicationsWithoutStatus(jobApplicationsToUpser);
7
8         if (Schema.sObjectType.Job_Application__c.isCreateable() && Schema.sObjectType.Job_App...
9             upsert as user jobApplicationsToUpser Job_Id__c;
10        }
11    }
12
13    private static List<Job_Application__c> createJobApplicationsFromJobs(List<String> record...
14        List<Job__c> jobs = [
15            SELECT
16                Id,
17                Title__c,
18                Location__c,
19                Snippet__c,
20                Salary__c,
21                Source__c,
22                Type__c,
23                Link__c,
24                Company__c,
25                Updated__c,
26                Id__c
27            FROM Job__c
28            WHERE Id IN :recordIds
29            WITH USER_MODE
30        ];
31
32        List<Job_Application__c> jobApplications = new List<Job_Application__c>();
33
34        for (Job__c job : jobs) {
35            Job_Application__c jobApplication = new Job_Application__c();
36
37            if (!String.isBlank(job.Title__c)) {
38                jobApplication.Position__title__c = job.Title__c;
39            }
40
41            if (!String.isBlank(job.Location__c)) {
42                jobApplication.Location__c = job.Location__c;
43            }
44
45            if (!String.isBlank(job.Snippet__c)) {
46                jobApplication.Job_Description__c = job.Snippet__c;
47            }
48
49            if (!String.isBlank(job.Salary__c)) {
50                try {
51                    jobApplication.Salary__c = Decimal.valueOf(job.Salary__c);
52                } catch (TypeException e) {
53                    final String SALARY_TO_JOB_DESCRIPTION = '\n\nSalary: ' + job.Salary__c;
54
55                    if (String.isNotBlank(jobApplication.Job_Description__c)) {
56                        jobApplication.Job_Description__c += SALARY_TO_JOB_DESCRIPTION.trim();
57                    } else {
58                        jobApplication.Job_Description__c += SALARY_TO_JOB_DESCRIPTION;
59                    }
60                }
61            }
62
63            if (!String.isBlank(job.Source__c)) {
64                final String SOURCE_TO_JOB_DESCRIPTION = '\n\nSource: ' + job.Source__c;
65
66                if (String.isNotBlank(jobApplication.Job_Description__c)) {
67                    jobApplication.Job_Description__c += SOURCE_TO_JOB_DESCRIPTION.trim();
68                } else {
69                    jobApplication.Job_Description__c += SOURCE_TO_JOB_DESCRIPTION;
70                }
71            }
72        }
73    }
74}
```



```
1 public with sharing class JobApplicationController {
2     @AuraEnabled
3     public static List<Job_Application__c> createJobApplicationsFromJobs(List<String> record...
4        List<Job__c> jobs = [
5            SELECT
6                Id,
7                Title__c,
8                Location__c,
9                Snippet__c,
10               Salary__c,
11               Source__c,
12               Type__c,
13               Link__c,
14               Company__c,
15               Updated__c,
16               Id__c
17           FROM Job__c
18           WHERE Id IN :recordIds
19           WITH USER_MODE
20       ];
21
22       List<Job_Application__c> jobApplications = new List<Job_Application__c>();
23
24       for (Job__c job : jobs) {
25           Job_Application__c jobApplication = new Job_Application__c();
26
27           if (!String.isBlank(job.Title__c)) {
28               jobApplication.Position__title__c = job.Title__c;
29           }
30
31           if (!String.isBlank(job.Location__c)) {
32               jobApplication.Location__c = job.Location__c;
33           }
34
35           if (!String.isBlank(job.Snippet__c)) {
36               jobApplication.Job_Description__c = job.Snippet__c;
37           }
38
39           if (!String.isBlank(job.Salary__c)) {
40               try {
41                   jobApplication.Salary__c = Decimal.valueOf(job.Salary__c);
42               } catch (TypeException e) {
43                   final String SALARY_TO_JOB_DESCRIPTION = '\n\nSalary: ' + job.Salary__c;
44
45                   if (String.isNotBlank(jobApplication.Job_Description__c)) {
46                       jobApplication.Job_Description__c += SALARY_TO_JOB_DESCRIPTION.trim();
47                   } else {
48                       jobApplication.Job_Description__c += SALARY_TO_JOB_DESCRIPTION;
49                   }
50               }
51
52               if (!String.isBlank(jobApplication.Job_Description__c)) {
53                   final String SOURCE_TO_JOB_DESCRIPTION = '\n\nSource: ' + job.Source__c;
54
55                   if (String.isNotBlank(jobApplication.Job_Description__c)) {
56                       jobApplication.Job_Description__c += SOURCE_TO_JOB_DESCRIPTION.trim();
57                   } else {
58                       jobApplication.Job_Description__c += SOURCE_TO_JOB_DESCRIPTION;
59                   }
60               }
61           }
62
63           if (!String.isBlank(job.Source__c)) {
64               final String SOURCE_TO_JOB_DESCRIPTION = '\n\nSource: ' + job.Source__c;
65
66               if (String.isNotBlank(jobApplication.Job_Description__c)) {
67                   jobApplication.Job_Description__c += SOURCE_TO_JOB_DESCRIPTION.trim();
68               } else {
69                   jobApplication.Job_Description__c += SOURCE_TO_JOB_DESCRIPTION;
70               }
71           }
72       }
73   }
74}
```

## JobController.cls:

The screenshot shows the JobController.cls file in a code editor. The code implements a sharing class with methods for querying and deleting jobs. The Copilot AI interface is visible on the right, displaying a welcome message and a text input field for adding context.

```
public with sharing class JobController {
    @AuraEnabled(cacheable=true)
    public static List<Job_c> queryJobs() {
        return [
            SELECT
                Id,
                title_c,
                Company_c,
                Location_c,
                Type_c,
                Salary_c,
                Snippet_c,
                Link_c,
                Source_c,
                Updated_c,
                Id_c
            FROM Job_c
            WITH USER_MODE
        ];
    }

    @AuraEnabled
    public static void deleteJobs(List<String> recordIds) {
        if (Schema.sObjectType.Job_c.isDeleteable()) {
            delete as user [SELECT Id FROM Job_c WHERE Id IN :recordIds WITH USER_MODE];
        }
    }

    @AuraEnabled
    public static void deleteAllJobs() {
        if (Schema.sObjectType.Job_c.isDeleteable()) {
            delete as user [SELECT Id FROM Job_c WITH USER_MODE];
        }
    }
}
```

## TaxCalculation.cls:

The screenshot shows the TaxCalculation.cls file in a code editor. The code contains two static methods for calculating federal and social security taxes based on salary. The Copilot AI interface is visible on the right, displaying a welcome message and a text input field for adding context.

```
public with sharing class TaxCalculation {
    @AuraEnabled(cacheable=true)
    public static Decimal calculateFederalTax(Decimal salary) {
        Decimal federalTax = 0;

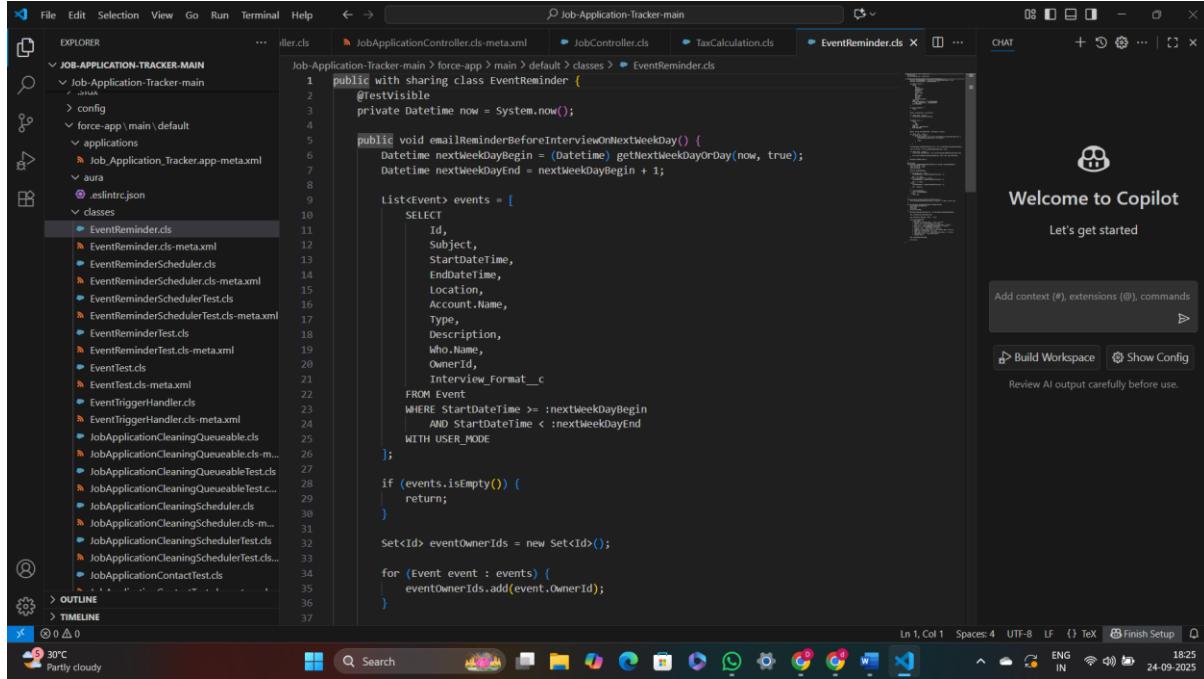
        Decimal salaryMinusStandardDeduction = salary - 13850;

        if (salaryMinusStandardDeduction == null || salaryMinusStandardDeduction <= 0) {
            return federalTax;
        } else if (salaryMinusStandardDeduction <= 11000) {
            federalTax = (salaryMinusStandardDeduction * 0.10);
        } else if (salaryMinusStandardDeduction <= 44725) {
            federalTax = ((salaryMinusStandardDeduction - 11000) * 0.12 + 1100);
        } else if (salaryMinusStandardDeduction <= 95375) {
            federalTax = ((salaryMinusStandardDeduction - 44725) * 0.22 + 5147);
        } else if (salaryMinusStandardDeduction <= 182100) {
            federalTax = ((salaryMinusStandardDeduction - 95375) * 0.24 + 16290);
        } else if (salaryMinusStandardDeduction <= 231250) {
            federalTax = ((salaryMinusStandardDeduction - 182100) * 0.32 + 37104);
        } else if (salaryMinusStandardDeduction <= 578125) {
            federalTax = ((salaryMinusStandardDeduction - 231250) * 0.35 + 52832);
        } else if (salaryMinusStandardDeduction > 578125) {
            federalTax = ((salaryMinusStandardDeduction - 578125) * 0.37 + 174238.25);
        }

        return federalTax.setScale(2, System.RoundingMode.HALF_UP);
    }

    @AuraEnabled(cacheable=true)
    public static Decimal calculateSocialSecurityTax(Decimal salary) {
        Decimal socialSecurityTax = 0.00;
        if (salary > 0){
            socialSecurityTax = (salary * 0.062);
            socialSecurityTax.setScale(2, System.RoundingMode.HALF_UP);
        }
        return socialSecurityTax;
    }
}
```

## EventReminder.cls:



```
1 public with sharing class EventReminder {
2     @restVisible
3     private Datetime now = System.now();
4
5     public void emailReminderBeforeInterviewOnNextWeekDay() {
6         Datetime nextWeekDayBegin = Datetime.getWeekdayOrDay(now, true);
7         Datetime nextWeekDayEnd = nextWeekDayBegin + 1;
8
9         List<Event> events = [
10             SELECT
11                 Id,
12                 Subject,
13                 StartDateTime,
14                 EndDateTime,
15                 Location,
16                 Account.Name,
17                 Type,
18                 Description,
19                 Who.Name,
20                 OwnerId,
21                 Interview_Format__c
22             FROM Event
23             WHERE StartDateTime > :nextWeekDayBegin
24                 AND StartDateTime < :nextWeekDayEnd
25             WITH USER_MODE
26         ];
27
28         if (events.isEmpty()) {
29             return;
30         }
31
32         Set<Id> eventOwnerIds = new Set<Id>();
33
34         for (Event event : events) {
35             eventOwnerIds.add(event.OwnerId);
36         }
37     }
38 }
```

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

### Theory:

Triggers are pieces of Apex code that run before or after records are inserted, updated, deleted, or undeleted. They are used to enforce business rules or automate processes.

In your project:

- JobApplicationTrigger.trigger → fires on job applications.
- EventTrigger.trigger → fires on events.
- JobApplicationContactTrigger.trigger → fires on job application contacts.

### JobApplicationTrigger.trigger:

```

File Edit Selection View Go Run Terminal Help < > Job-Application-Tracker-main
EXPLORER ler.cls-meta.xml JobController.cls TaxCalculation.cls EventReminder.cls JobApplicationTrigger.trigger ...
Job-Application-Tracker-main > force-app > main > default > triggers > JobApplicationTrigger.trigger
JobApplicationTriggerHandler jobApplicationTriggerHandler = new JobApplicationTriggerHandler();
switch on Trigger.operationType {
    when BEFORE_INSERT {
        jobApplicationTriggerHandler.validateApplicationAndFollowUpDate(Trigger.new);
        jobApplicationTriggerHandler.setPrimaryContact(Trigger.new);
        TaxCalculation.calculatePayrollTaxes(Trigger.new);
    }
    when BEFORE_UPDATE {
        jobApplicationTriggerHandler.validateApplicationAndFollowUpDate(Trigger.new);
        jobApplicationTriggerHandler.setPrimaryContact(Trigger.new);
        TaxCalculation.calculatePayrollTaxes(Trigger.new);
    }
    when AFTER_INSERT {
        jobApplicationTriggerHandler.createTask_Insert(Trigger.new);
    }
    when AFTER_UPDATE {
        jobApplicationTriggerHandler.createTask_Update(Trigger.oldMap, Trigger.new);
    }
}

```

CHAT + ⚙️ ... | 🔍

Welcome to Copilot

Let's get started

Add context (#), extensions (@), commands ▶

Build Workspace Show Config

Review AI output carefully before use.

Ln 25, Col 2 Spaces: 4 UTF-8 LF Plain Text Finish Setup

30°C Party cloudy

## EventTrigger.trigger:

```

File Edit Selection View Go Run Terminal Help < > Job-Application-Tracker-main
EXPLORER EventTrigger.trigger-meta.xml EventTrigger.trigger Job_Application_Compact_Layout.compactLayout-meta.xml ...
Job-Application-Tracker-main > force-app > main > default > triggers > EventTrigger.trigger
trigger EventTrigger on Event (before insert, before update) {
    eventTriggerHandler eventTriggerHandler = new eventTriggerHandler();
}
switch on Trigger.operationType {
    when BEFORE_INSERT {
        eventTriggerHandler.validateEventDateTime(Trigger.new);
    }
    when BEFORE_UPDATE {
        eventTriggerHandler.validateEventDateTime(Trigger.new);
    }
}

```

CHAT + ⚙️ ... | 🔍

Welcome to Copilot

Let's get started

Add context (#), extensions (@), commands ▶

Build Workspace Show Config

Review AI output carefully before use.

Ln 13, Col 2 Spaces: 4 UTF-8 LF Plain Text Finish Setup

30°C Party cloudy

## JobApplicationContactTrigger.trigger:

```
trigger JobApplicationContactTrigger on Job_Application__c (after insert, after update)
{
    switch on Trigger.operationType {
        when AFTER_INSERT {
            jobApplicationContactTriggerHandler.setPrimaryContact(Trigger.new);
        }
        when AFTER_UPDATE {
            jobApplicationContactTriggerHandler.setPrimaryContact(Trigger.new);
        }
    }
}
```

### 3. Trigger Design Pattern

#### Theory:

To avoid bulky triggers, the Trigger Handler Pattern is used. Each trigger delegates logic to a handler class, keeping triggers clean and single-purposed.

#### In your project:

- **JobApplicationTriggerHandler.cls**
- **EventTriggerHandler.cls**
- **JobApplicationContactTriggerHandler.cls**

```

File Edit Selection View Go Run Terminal Help < > Job-Application-Tracker-main
trigger-meta.xml EventTrigger.trigger JobApplicationContactTrigger.trigger JobApplicationTriggerHandler.cls ... CHAT + ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ...
EXPLORER ... .trigger-meta.xml
JOB-APPLICATION-TRACKER-MAIN ...
Job-Application-Tracker-main ...
force-app/main/default ...
classes ...
EventTest.cls-meta.xml ...
EventTriggerHandler.cls ...
EventTriggerHandler.cls-meta.xml ...
JobApplicationCleaningQueueable.cls ...
JobApplicationCleaningQueueable.cls-meta.xml ...
JobApplicationCleaningQueueableTest.cls ...
JobApplicationCleaningQueueableTest.cls ...
JobApplicationCleaningScheduler.cls ...
JobApplicationCleaningScheduler.cls-meta.xml ...
JobApplicationCleaningSchedulerTest.cls ...
JobApplicationCleaningSchedulerTest.cls ...
JobApplicationContactTest.cls ...
JobApplicationContactTest.cls-meta.xml ...
JobApplicationContactTriggerHandler.cls ...
JobApplicationContactTriggerHandler.cls ...
JobApplicationController.cls ...
JobApplicationController.cls-meta.xml ...
JobApplicationControllerTest.cls ...
JobApplicationControllerTest.cls-meta.xml ...
JobApplicationTest.cls ...
JobApplicationTest.cls-meta.xml ...
JobApplicationTriggerHandler.cls ...
JobApplicationTriggerHandler.cls ...
JobController.cls ...
JobController.cls-meta.xml ...
JobControllerTest.cls ...
JobControllerTest.cls-meta.xml ...

```

```

public with sharing class JobApplicationTriggerHandler {
    public void validateApplicationAndFollowUpDate(List<Job_Application__c> triggerDotNew) {
        for (Job_Application__c newJobApplication : triggerDotNew) {
            if (newJobApplication.Application_Date__c > newJobApplication.Follow_Up_Date__c) {
                final String ERROR = 'Application Date cannot be after Follow-Up Date';
                newJobApplication.addError(ERROR);
            }
        }
    }

    public void createTask_Insert(List<Job_Application__c> triggerDotNew) {
        List<Task> tasksToInsert = new List<Task>();

        for (Job_Application__c newJobApplication : triggerDotNew) {
            if (newJobApplication.Status__c != null) {
                tasksToInsert.add(createTask(newJobApplication));
            }
        }

        if (Schema.sObjectType.Task.isCreateable()) {
            insert as user tasksToInsert;
        }
    }

    private Task createTask(Job_Application__c jobApplication) {
        String status = jobApplication.Status__c;

        Id jobApplicationId = jobApplication.Id;
        Id jobApplicationPrimaryContact = jobApplication.Primary_Contact__c;
        Id jobApplicationOwnerId = jobApplication.OwnerId;

        Task task = new Task();

        switch on status {
            when 'Saved' {
                task.Subject = 'Things to do for a job saved';
            }
        }
    }
}

```

Ln 20, Col 1 Spaces: 4 UTF-8 LF ⌂ Tex ⌂ Finish Setup

30°C Party cloudy

Welcome to Copilot Let's get started

Add context (#), extensions (@), commands

Build Workspace Show Config

Review AI output carefully before use.

OUTLINE TIMELINE

18:33 24-09-2025

## EventTriggerHandler.cls:

```

File Edit Selection View Go Run Terminal Help < > Job-Application-Tracker-main
Trigger.trigger JobApplicationContactTrigger.trigger JobApplicationTriggerHandler.cls ... CHAT + ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ...
EXPLORER ... .trigger-meta.xml
JOB-APPLICATION-TRACKER-MAIN ...
Job-Application-Tracker-main ...
force-app/main/default ...
applications ...
Job-Application_Tracker.app-meta.xml ...
aura ...
.eslintrc.json ...
classes ...
EventReminder.cls ...
EventReminder.cls-meta.xml ...
EventReminderScheduler.cls ...
EventReminderScheduler.cls-meta.xml ...
EventReminderSchedulerTest.cls ...
EventReminderSchedulerTest.cls ...
EventReminderTest.cls ...
EventReminderTest.cls-meta.xml ...
EventTest.cls ...
EventTest.cls-meta.xml ...
EventTriggerHandler.cls ...
EventTriggerHandler.cls ...
EventTriggerHandler.cls ...
EventTriggerHandler.cls ...
EventTriggerHandler.cls ...
EventTriggerHandler.cls ...
JobApplicationCleaningQueueable.cls ...
JobApplicationCleaningQueueable.cls-meta.xml ...
JobApplicationCleaningQueueableTest.cls ...
JobApplicationCleaningQueueableTest.cls ...
JobApplicationCleaningQueueableTest.cls ...
JobApplicationCleaningScheduler.cls ...
JobApplicationCleaningScheduler.cls ...
JobApplicationCleaningSchedulerTest.cls ...
JobApplicationContactTest.cls ...
JobApplicationContactTest.cls ...
JobApplicationContactTriggerHandler.cls ...

```

```

public with sharing class EventTriggerHandler {
    public void validateEventTime(List<Event> triggerDotNew) {
        List<Event> events = [
            SELECT
                StartDateTime,
                EndDateTime
            FROM Event
            WHERE Event.Id NOT IN :triggerDotNew
            WITH USER_MODE
        ];

        for (Event newEvent : triggerDotNew) {
            Datetime newEventStartDT = newEvent.StartDateTime;
            Datetime newEventEndDT = newEvent.EndDateTime;

            Long newEventStartDTInMilliseconds = newEventStartDT.getTime();
            Long newEventEndDTInMilliseconds = newEventEndDT.getTime();

            if (isWeekend(newEventStartDT) || isWeekend(newEventEndDT)) {
                newEvent.addError('Cannot schedule event on the weekend');
            } else if (!events.isEmpty()) {
                final String ERROR = validateNoEventOverlap(events, newEventStartDTInMilliseconds);

                if (error != null) {
                    newEvent.addError(ERROR);
                }
            }
        }
    }

    private Boolean isweekend(Datetime dt) {
        final String SAT = 'Sat';
        final String SUN = 'Sun';

        if (dt.format('E') == SAT || dt.format('E') == SUN) {
            return true;
        } else {
            return false;
        }
    }
}

```

Ln 22, Col 59 Spaces: 4 UTF-8 LF ⌂ Tex ⌂ Finish Setup

30°C Party cloudy

Welcome to Copilot Let's get started

Add context (#), extensions (@), commands

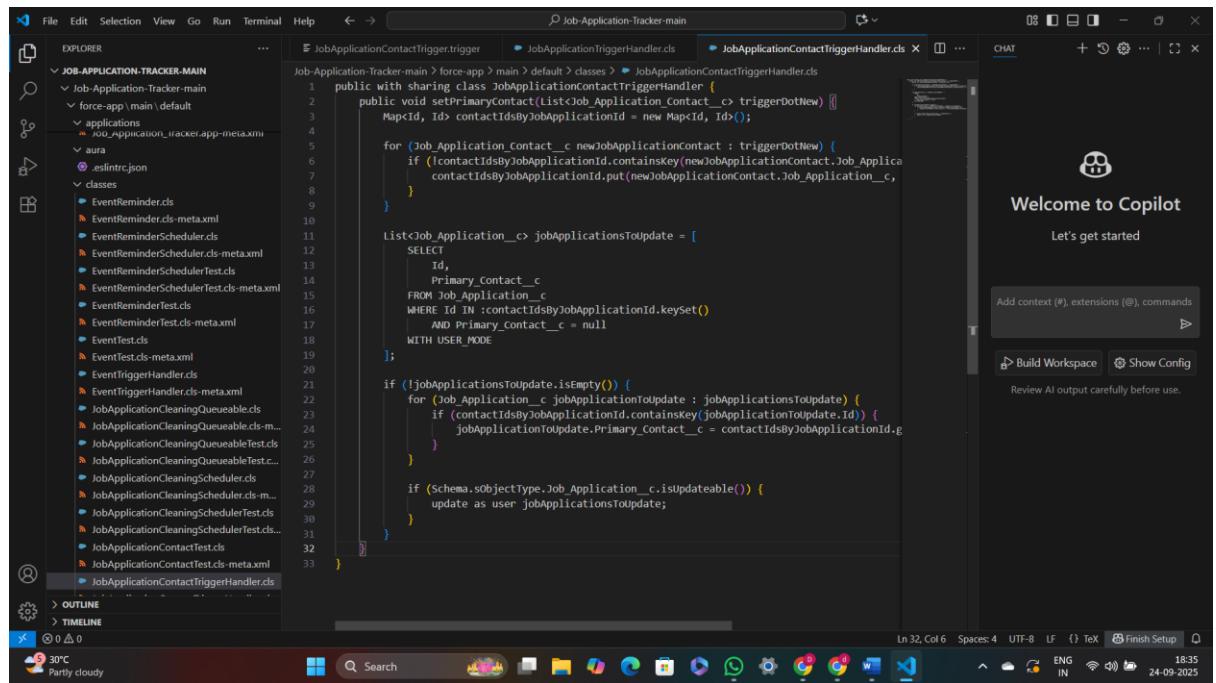
Build Workspace Show Config

Review AI output carefully before use.

OUTLINE TIMELINE

18:34 24-09-2025

## JobApplicationContactTriggerHandler.cls:



The screenshot shows the Salesforce IDE interface with the code editor open for the class `JobApplicationContactTriggerHandler.cls`. The code implements a trigger handler for the `Job_Application__c` object. It includes logic to set the primary contact for new job applications and update existing ones based on contact IDs. The code is annotated with line numbers from 1 to 33. The right side of the interface features the Copilot AI sidebar.

```
1 public with sharing class JobApplicationContactTriggerHandler {
2     public void setPrimaryContact(List<Job_Application__c> triggerNew) {
3         Map<Id, Id> contactIdsByJobApplicationId = new Map<Id, Id>();
4
5         for (Job_Application__c newJobApplicationContact : triggerNew) {
6             if (!contactIdsByJobApplicationId.containsKey(newJobApplicationContact.Job_Application__c))
7                 contactIdsByJobApplicationId.put(newJobApplicationContact.Job_Application__c,
8                     newJobApplicationContact.Id);
9         }
10
11         List<Job_Application__c> jobApplicationsToUpdate = [
12             SELECT
13                 Id,
14                 Primary_Contact__c
15             FROM Job_Application__c
16             WHERE Id IN :contactIdsByJobApplicationId.keySet()
17                 AND Primary_Contact__c = null
18             WITH USER_MODE
19         ];
20
21         if (!jobApplicationsToUpdate.isEmpty()) {
22             for (Job_Application__c jobApplicationToUpdate : jobApplicationsToUpdate) {
23                 if (contactIdsByJobApplicationId.containsKey(jobApplicationToUpdate.Id)) {
24                     jobApplicationToUpdate.Primary_Contact__c = contactIdsByJobApplicationId.get(
25                         jobApplicationToUpdate.Id);
26                 }
27             }
28
29             if (Schema.sObjectType.Job_Application__c.isUpdateable()) {
30                 update as user jobApplicationsToUpdate;
31             }
32         }
33     }
}
```

## 4. SOQL & SOSL

### Theory:

- SOQL (Salesforce Object Query Language): Used to fetch records from Salesforce objects (similar to SQL SELECT).**
- SOSL (Salesforce Object Search Language): Used for searching text across multiple objects/fields.**

### In your project:

- Queries written in Apex classes like `JobController.cls`.
- Standalone query files in `scripts/soql`:
  - `Job_Application__c.soql`
  - `Job__c.soql`
  - `account.soql`

### **Job\_Application\_\_c.soql: (Query that I had given)**

```
SELECT Name, Status__c, Application_Date__c, Follow_Up_Date__c, Notes__c,  
Rating__c, Position_Title__c, Work_Model__c, Location__c, Job_Type__c,  
Job_Description__c, Salary__c, Take_Home_Pay__c, URL__c, Job_Id__c, Company__c,  
Primary_Contact__c, CreatedDate FROM Job_Application__c ORDER BY CreatedDate  
ASC
```

### **Job\_\_c.soql:**

```
SELECT Id, Title__c, Location__c, Snippet__c, Salary__c, Source__c, Type__c, Link__c,  
Company__c, Updated__c, Id__c, CreatedDate FROM Job__c ORDER BY CreatedDate  
ASC
```

### **account.soql:**

```
SELECT Id, Name FROM Account
```

## **5. Collections: List, Set, Map**

### **Theory:**

- **List** → Ordered collection of elements (can contain duplicates).
- **Set** → Unordered collection of unique elements.
- **Map** → Key-value pairs.  
Collections make bulk processing efficient in Salesforce.

### **In your project:**

- TestDataFactory.cls → creates bulk test data using lists and maps.
- JobApplicationController.cls & JobController.cls → use collections to store query results (List<Job\_\_c>).

## TestDataFactory.cls:

```
1 public with sharing class TestDataFactory {
2     @testVisible
3     private List<Account> createAccts_setup_JobApplicationTest() {
4         List<Account> accts = new List<Account>();
5
6         Account accBlackthorn = new Account(Name = 'Blackthorn');
7         accts.add(accBlackthorn);
8
9         Account accHTS = new Account(Name = 'Hudson Technology Systems');
10
11         accts.add(accHTS);
12
13         Account accBAH = new Account(Name = 'Booz Allen Hamilton');
14
15         accts.add(accBAH);
16
17         Account accSalesforce = new Account(Name = 'Salesforce');
18
19         accts.add(accSalesforce);
20
21         return accts;
22     }
23
24     @testVisible
25     private List<Contact> createContacts_setup_JobApplicationTest() {
26         List<Contact> contacts = new List<Contact>();
27
28         Contact contactThornie = new Contact(LastName = 'Thornie');
29
30         contacts.add(contactThornie);
31
32         Contact contactHudson = new Contact(LastName = 'Hudson');
33
34         contacts.add(contactHudson);
35
36         Contact contactBooz = new Contact(LastName = 'Booz');
37     }
}
```

## 7. Batch Apex

### Theory:

**Batch Apex processes large data volumes asynchronously in smaller chunks (batches), respecting governor limits.**

### In your project:

- **JobApplicationCleaningQueueable.cls** (used like batch/queueable for cleaning old job applications).

## Queueable Apex

### Theory:

**Queueable Apex allows running jobs asynchronously, and supports job chaining for more flexibility than @future.**

### In your project:

- **JobApplicationCleaningQueueable.cls** → runs cleaning logic asynchronously.

```
Job-Application-Tracker-main > force-app > main > default > classes > JobApplicationCleaningQueueableTest.cls
```

```
1  @test
2  private class JobApplicationCleaningQueueableTest {
3      @TestSetup
4      private static void setup() {
5          List<Job_Application__c> jobApplicationsToInsert = new TestDataFactory().createJobAppli...
6
7          if (Schema.sObjectType.Job_Application__c.isCreateable()) {
8              insert as user jobApplicationsToInsert;
9          }
10     }
11
12     @IsTest
13     private static void update_two_job_applications_status_to_Closed() {
14         // GIVEN
15
16         // WHEN
17         Test.startTest();
18
19         Id jobId = System.enqueueJob(new JobApplicationCleaningQueueable());
20
21         AsyncApexJob jobInfo_WHEN = [
22             SELECT
23                 Status,
24                 TotalJobItems,
25                 JobItemsProcessed,
26                 NumberOfErrors
27             FROM AsyncApexJob
28             WHERE Id = :jobId
29             WITH USER_MODE
30             LIMIT 1
31         ];
32
33         System.assertEquals('Queued', jobInfo_WHEN.Status, 'Expect Status to be Queued');
34         System.assertEquals(0, jobInfo_WHEN.TotalJobItems, 'Expect 0 TotalJobItems');
35         System.assertEquals(0, jobInfo_WHEN.JobItemsProcessed, 'Expect 0 JobItemsProcessed');
36         System.assertEquals(0, jobInfo_WHEN.NumberOfErrors, 'Expect 0 NumberOfErrors');
37     }
38 }
```