Phase 5: Apex Programming (Developer) – AI Career Coach Salesforce Integration

1. Classes & Objects

- **Concept:** Apex is an object-oriented language. Classes define blueprints; objects are instances.
- **Usage in AI Career Coach:** Define classes for `Student`, `CareerProfile`, `AssessmentResult`.
- **Sample Apex Class:**

```
public class Student {
    public String name;
    public Integer age;

    public Student(String name, Integer age) {
        this.name = name;
        this.age = age;
    }

    public void displayInfo() {
        System.debug('Name: ' + name + ', Age: ' + age);
    }
}
```

- **Screenshot suggestion:** Capture `Developer Console` → `Classes` tab showing the `Student` class.

---

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

- **Concept:** Triggers automate actions on record changes.
- **Example Use Case:** Automatically create a `CareerProfile` when a `Student` is inserted.
- **Sample Trigger:**

```
trigger CreateCareerProfile on Student__c
(after insert) {
    List<CareerProfile__c> profiles = new
List<CareerProfile__c>();
    for(Student__c s : Trigger.new) {
        profiles.add(new
CareerProfile__c(Student__c = s.Id,
Status__c='New'));
    }
    insert profiles;
}
```

- **Screenshot suggestion:** `Setup → Object Manager → Student__c → Triggers → Show CreateCareerProfile trigger`.

---

3. Trigger Design Pattern

- **Concept:** Use “One Trigger per Object” and separate logic into **Handler Classes**.
- **Example:**

```
trigger StudentTrigger on Student__c (before
insert, after insert) {
    if(Trigger.isAfter && Trigger.isInsert){

StudentTriggerHandler.createProfiles(Trigger.ne
w);
    }
}
```

- **Screenshot suggestion:** `Handler class` in `Developer Console` and linked `Trigger`.

---

4. SOQL & SOSL

- **Concept:** Query Salesforce data using SOQL (object-specific) or SOSL (multi-object search).
- **Example:**

```
List<Student__c> students = [SELECT Id, Name
FROM Student__c WHERE Age__c > 20];
List<List<SObject>> searchResults = [FIND 'AI*'
IN ALL FIELDS RETURNING CareerProfile__c(Name,
Status__c)];
```

- **Screenshot suggestion:** `Developer Console → Query Editor` showing query execution.

---

5. Collections: List, Set, Map

- **Usage:** Store multiple records efficiently.

```
List<String> names = new
List<String>{'Alice','Bob'};
Set<Integer> ages = new Set<Integer>{21,22};
Map<Id, Student__c> studentMap = new Map<Id,
Student__c>([SELECT Id, Name FROM Student__c]);
```

- **Screenshot suggestion:** Capture `Debug Logs` showing collection contents.

---

6. Control Statements

- **Usage:** `if-else`, loops, `switch`, for automation.

```
for(Student__c s : [SELECT Name, Age__c FROM
Student__c]){
    if(s.Age__c > 22){
        System.debug(s.Name + ' is eligible for
mentorship');
    }
```

```
}
```

- **Screenshot suggestion:** `Developer Console → Logs` showing conditional logic in action.

---

7. Batch Apex

- **Use Case:** Process large datasets asynchronously.

```
global class StudentBatch implements
Database.Batchable<SObject> {
    global Database.QueryLocator
start(Database.BatchableContext BC) {
        return Database.getQueryLocator('SELECT
Id FROM Student__c');
    }
    global void
execute(Database.BatchableContext BC,
List<Student__c> scope){
        for(Student__c s : scope){
            s.Status__c = 'Processed';
        }
        update scope;
    }
    global void
finish(Database.BatchableContext BC){}
}
```

- **Screenshot suggestion:** `Setup → Apex Classes → Schedule/Execute Batch`.

---

8. Queueable Apex

- **Use Case:** Chain asynchronous processes.

```
public class CareerQueueable implements
Queueable {
    public void execute(QueueableContext
context){
        System.debug('Queueable executed for
career updates');
    }
}
```

- **Screenshot suggestion:** `Developer Console → Execute Anonymous → System.enqueueJob(new CareerQueueable());`.

---

9. Scheduled Apex

- **Use Case:** Run periodic jobs.

```
global class CareerScheduler implements
Schedulable {
    global void execute(SchedulableContext sc){
        System.debug('Scheduled job executed');
    }
}
```

- **Screenshot suggestion:** `Setup → Apex Classes → Schedule Apex → Add CareerScheduler`.

---

10. Future Methods

- **Use Case:** Run tasks asynchronously (like sending emails).

```
@future
public static void sendEmail(String email){
    System.debug('Email sent to: ' + email);
}
```

- **Screenshot suggestion:** `Developer Console → Execute Anonymous → sendEmail('student@example.com');.`

---

11. Exception Handling

```
try {
    insert new Student__c(Name=null);
} catch(DmlException e){
    System.debug('Error: ' + e.getMessage());
}
```

- **Screenshot suggestion:** `Debug Log` showing handled exception.

---

12. Test Classes

- **Requirement:** Minimum 75% code coverage.

```
@IsTest
private class TestStudent {
    static testMethod void testCreateProfile()
{
        Student__c s = new
Student__c(Name='Test');
        insert s;
        System.assertEquals(1, [SELECT count()
FROM Student__c WHERE Name='Test']);
    }
}
```

- **Screenshot suggestion:** `Setup → Apex Test Execution → Run TestStudent`.

---

13. Asynchronous Processing

- **Summary:** Combines Batch, Queueable, Scheduled, and Future methods for scalable automation.
- **Screenshot suggestion:** `Monitor → Apex Jobs` showing all async jobs for AI Career Coach.