Phase 5 – Apex Programming (Developer)

---

1. Classes & Objects

Use Case: Encapsulate shipment-related operations in reusable code.
Explanation: Apex classes store logic for shipment updates, validations, or calculations that can be reused across triggers, flows, and other processes.

Example:

```
public class ShipmentHelper {
    public static void markDelivered(List<Shipment__c> shipments) {
        for(Shipment__c s : shipments){
            s.Status__c = 'Delivered';
        }
        update shipments;
    }
}
```

---

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

Use Case: Perform automatic actions when shipment records are created or updated.

Example – After Insert Trigger:

```
trigger ShipmentTrigger on Shipment__c (after insert) {
    for(Shipment__c s : Trigger.New){
        if(s.Shipment_Value__c > 50000){
            System.debug('High-value shipment created: ' + s.Name);
        }
    }
}
```

- Before Triggers: Validate data before saving.
- After Triggers: Execute logic after record commit.
  Screenshot: Trigger setup page.

---

3. Trigger Design Pattern

Use Case: Organize trigger logic for maintainability and scalability.

Example:

```
trigger ShipmentTrigger on Shipment__c (before update, after update) {
   if(Trigger.isBefore && Trigger.isUpdate){
      ShipmentTriggerHandler.beforeUpdate(Trigger.New, Trigger.OldMap);
   }
}
public class ShipmentTriggerHandler {
   public static void beforeUpdate(List<Shipment__c> newList, Map<Id, Shipment__c>
oldMap){
      for(Shipment__c s : newList){
         if(s.Status__c < oldMap.get(s.Id).Status__c){
            s.Status__c.addError('Cannot downgrade shipment status');
         }
      }
   }
}
```

---

4. SOQL & SOSL

Use Case: Query shipment and customer data.

Examples:

```
// SOQL – get shipments for a customer
List<Shipment__c> shipments = [SELECT Id, Name, Status__c FROM Shipment__c
WHERE Customer__c = :customerId];


// SOSL – search shipment by tracking number
```

```
List<List<SObject>> results = [FIND 'TRACK123*' IN ALL FIELDS RETURNING
Shipment__c(Id, Name, Tracking_Number__c)];
```

---

5. Collections: List, Set, Map

Use Case: Store and process multiple shipment records efficiently.

```
List<Shipment__c> shipmentList = new List<Shipment__c>();

Set<Id> shipmentIds = new Set<Id>();

Map<Id, Shipment__c> shipmentMap = new Map<Id, Shipment__c>([SELECT Id,
Status__c FROM Shipment__c]);
```

---

6. Control Statements

Use Case: Execute logic based on conditions.

```
for(Shipment__c s : shipmentList){
    if(s.Status__c == 'Delayed'){
        System.debug('Notify logistics team: ' + s.Name);
    }
}
```

---

7. Batch Apex

Use Case: Process large numbers of shipments (bulk update).

```
global class BatchUpdateShipmentStatus implements Database.Batchable<SObject>{
    global Database.QueryLocator start(Database.BatchableContext BC){
        return Database.getQueryLocator([SELECT Id, Status__c FROM Shipment__c WHERE
Status__c = 'Dispatched']);
    }
    global void execute(Database.BatchableContext BC, List<Shipment__c> scope){
```

```
        for(Shipment__c s : scope){
            s.Status__c = 'In Transit';
        }
        update scope;
    }
    global void finish(Database.BatchableContext BC){}
}
```

---

8. Queueable Apex

Use Case: Perform asynchronous operations like sending notifications.

```
public class ShipmentNotificationQueueable implements Queueable {
    public List<Id> shipmentIds;
    public ShipmentNotificationQueueable(List<Id> ids){ this.shipmentIds = ids; }
    public void execute(QueueableContext context){
        System.debug('Notify shipments: ' + shipmentIds);
    }
}
System.enqueueJob(new ShipmentNotificationQueueable(shipmentIds));
```

---

9. Scheduled Apex

Use Case: Daily check for overdue shipments.

```
global class DailyOverdueCheck implements Schedulable {
    global void execute(SchedulableContext sc){
        List<Shipment__c> overdue = [SELECT Id FROM Shipment__c WHERE Status__c != 'Delivered' AND Expected_Delivery_Date__c < TODAY];
        System.debug('Overdue shipments: ' + overdue.size());
    }
}
```

---

10. Future Methods

Use Case: Send email notifications asynchronously after shipment updates.

```
public class ShipmentEmailService {
   @future(callout=true)
   public static void sendShipmentEmail(Id shipmentId){
      Shipment__c s = [SELECT Id, Customer_Email__c FROM Shipment__c WHERE Id =
:shipmentId];
      System.debug('Send email to: ' + s.Customer_Email__c);
   }
}
```

---

11. Exception Handling

Use Case: Handle errors during shipment updates gracefully.

```
try{
   update shipmentList;
} catch(DmlException e){
   System.debug('Error updating shipments: ' + e.getMessage());
}
```

---

12. Test Classes

Use Case: Validate Apex code for deployment.

```
@isTest
private class ShipmentHelperTest {
   @isTest static void testMarkDelivered(){
      Shipment__c s = new Shipment__c(Name='Test', Status__c='Dispatched');
      insert s;
      ShipmentHelper.markDelivered(new List<Shipment__c>{s});
      s = [SELECT Status__c FROM Shipment__c WHERE Id = :s.Id];
```

```
        System.assertEquals('Delivered', s.Status__c);
    }
}
```
