

Magdalena Larment
Salesforce
Developer



Fasha



OPTIMISER UN BACK END
APEX

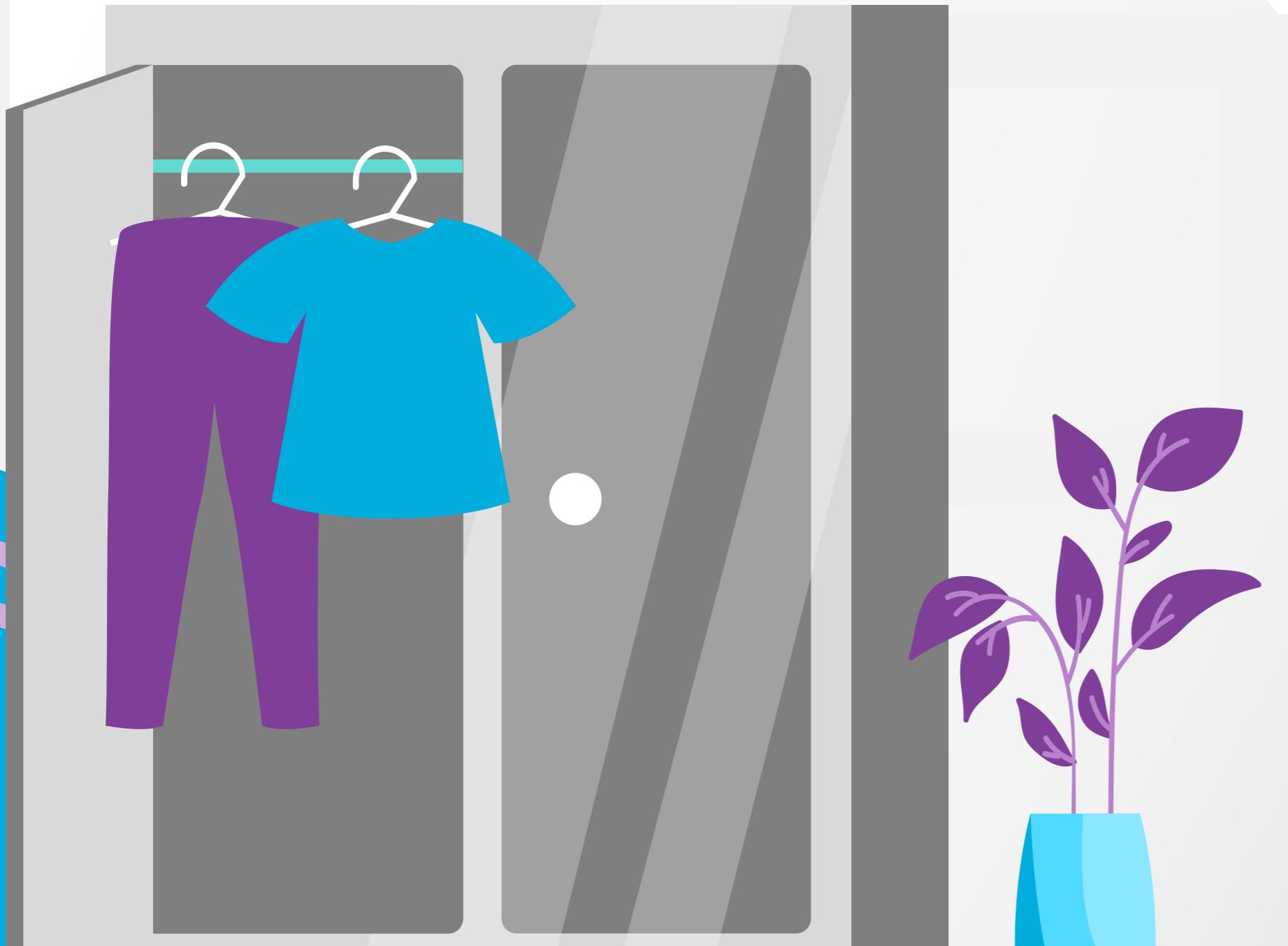
Sommaire

- INTRO** Rappel du projet
- 1** Le Trigger
- 2** Le Batch
- 3** Les Classes
- 4** Les Tests
- CONCLU** Démo et Conclusion





Intro



Projet mené pour la Société Fasha



Rappels du projet

Rappel du projet

02-06-2023

Date démarrage

02-07-2023

Date de fin

35 000€

Coût du projet

BESOINS EXPRIMÉS

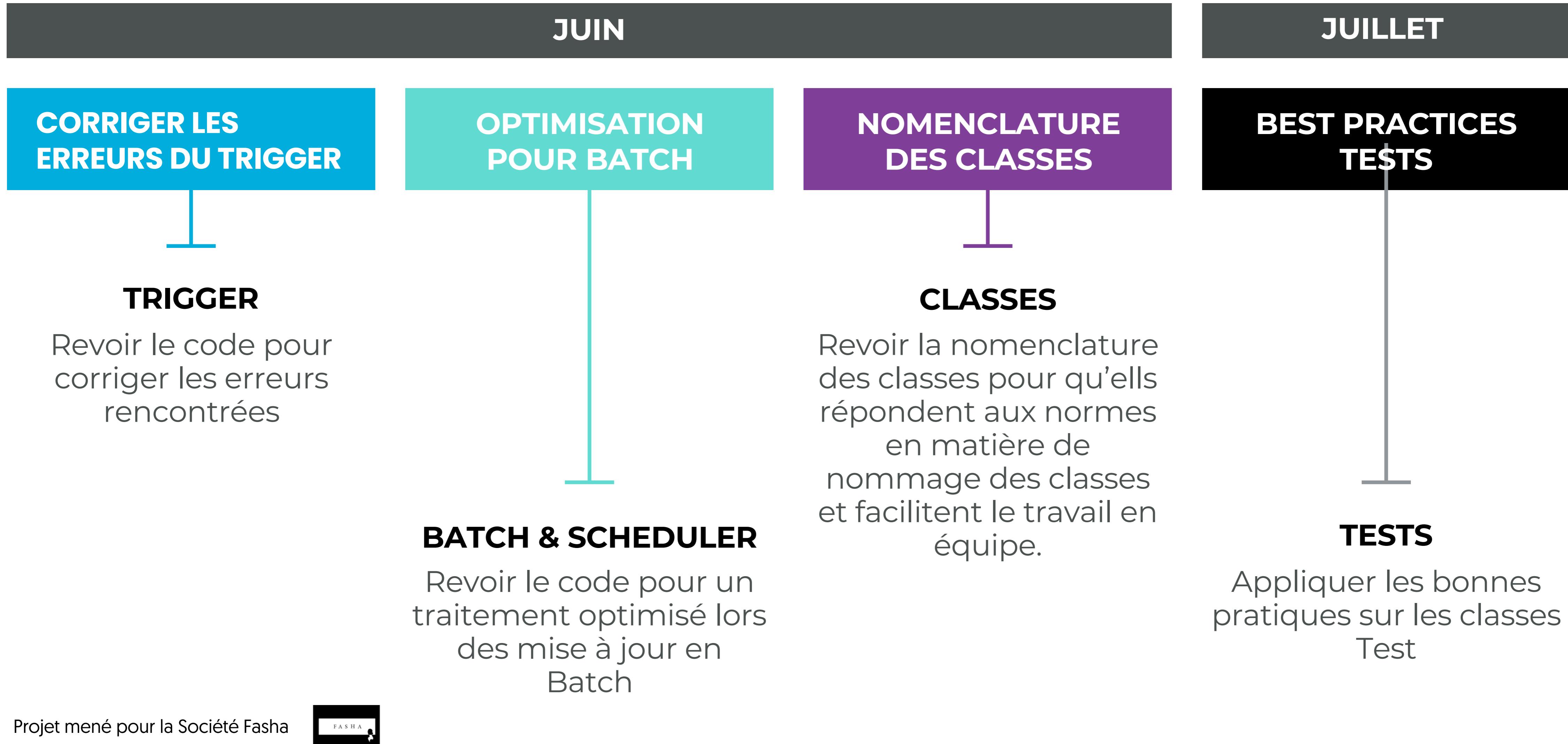
- Optimiser l'application pour le traitement des données en masse
- Corriger les bugs
- Réorganiser le code

PROBLÈMES RENCONTRES SUR L'APPLICATION

- ✓ Code pas optimisé pour le traitement des données en Batch
- ✓ Problème de trigger (résultant en des erreurs)
- ✓ Mauvaises pratiques de Tests
- ✓ Nomenclature à revoir



Rappel du projet





1



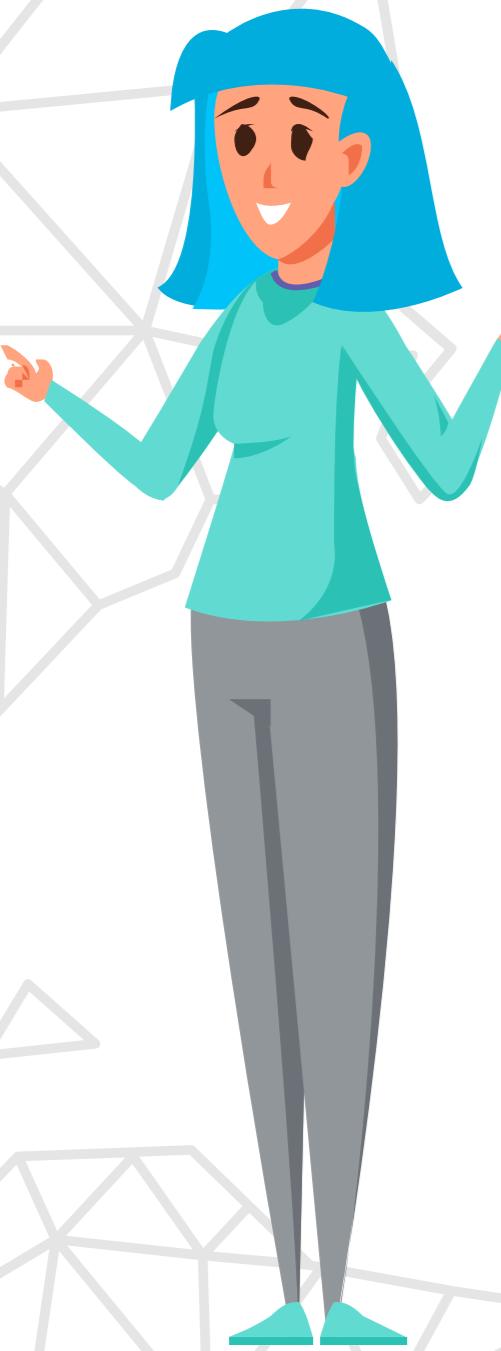
Projet mené pour la Société Fasha



Le Trigger

Correction du Trigger

Mauvaise Pratique
2 trigger par objet



2 triggers

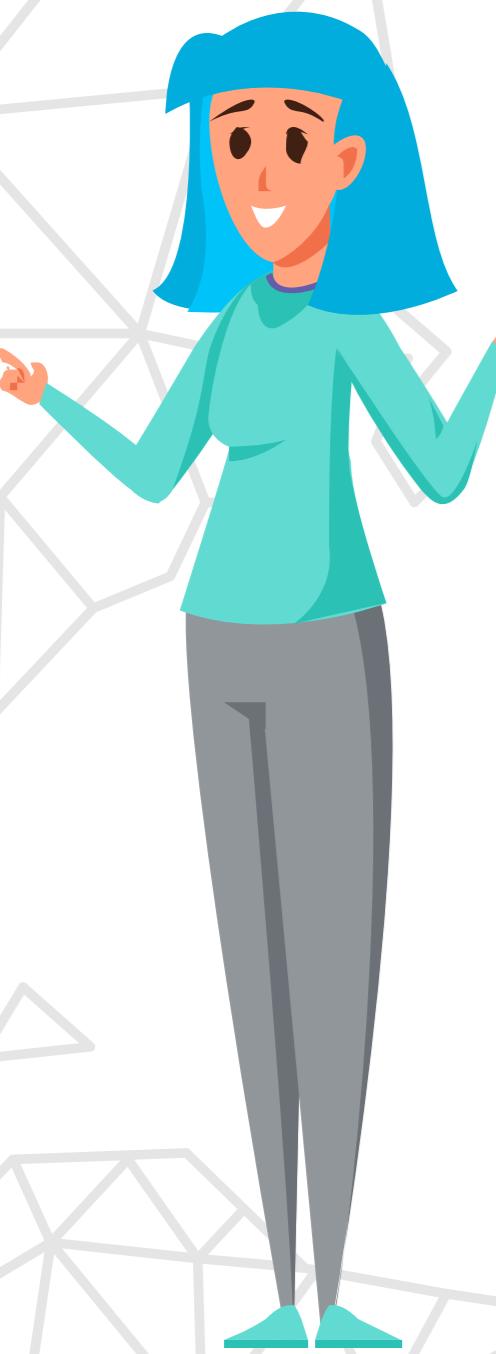
A screenshot of a GitHub repository titled "OpenClassrooms-Student-Center / Optimisez-un-backend-Apex". The repository has 5 lines (4 loc) and 159 Bytes. It contains two Apex trigger files:

```
trigger CalculMontant on Order (before update) {
    Order newOrder= trigger.new[0];
    newOrder.NetAmount__c = newOrder.TotalAmount - newOrder.ShipmentCost__c;
}
```

```
trigger UpdateAccountCA on Order (after update) {
    set<Id> setAccountIds = new set<Id>();
    for(integer i=0; i< trigger.new.size(); i++){
        Order newOrder= trigger.new[i];
        Account acc = [SELECT Id, Chiffre_d_affaire__c FROM Account WHERE Id =:newOrder.AccountId ];
        acc.Chiffre_d_affaire__c = acc.Chiffre_d_affaire__c + newOrder.TotalAmount;
        update acc;
    }
}
```

Correction du Trigger

Bonnes Pratiques
1 trigger par objet

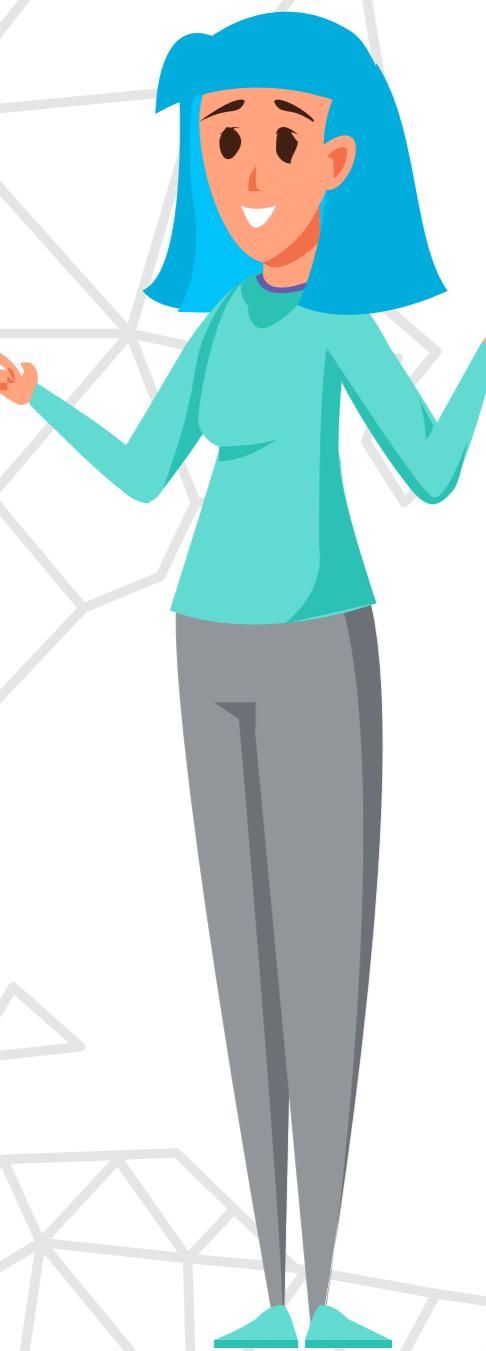
A screenshot of the Visual Studio Code interface. The title bar says "OrderTrigger.trigger - Fasha_OC_P9 - Visual Studio Code". The left sidebar shows a project structure under "FASHA_OC_P9": force-app > main > default > triggers > OrderTrigger.trigger. The main editor pane displays the following code:

```
trigger OrderTrigger on Order (after update, before update) {
    new OrderTriggerHandler().run();
}
```

The "OrderTrigger.trigger" file is highlighted in the sidebar. The status bar at the bottom shows "FASHA" and a small profile icon.

Correction du Trigger

Bonnes Pratiques
1 trigger par objet



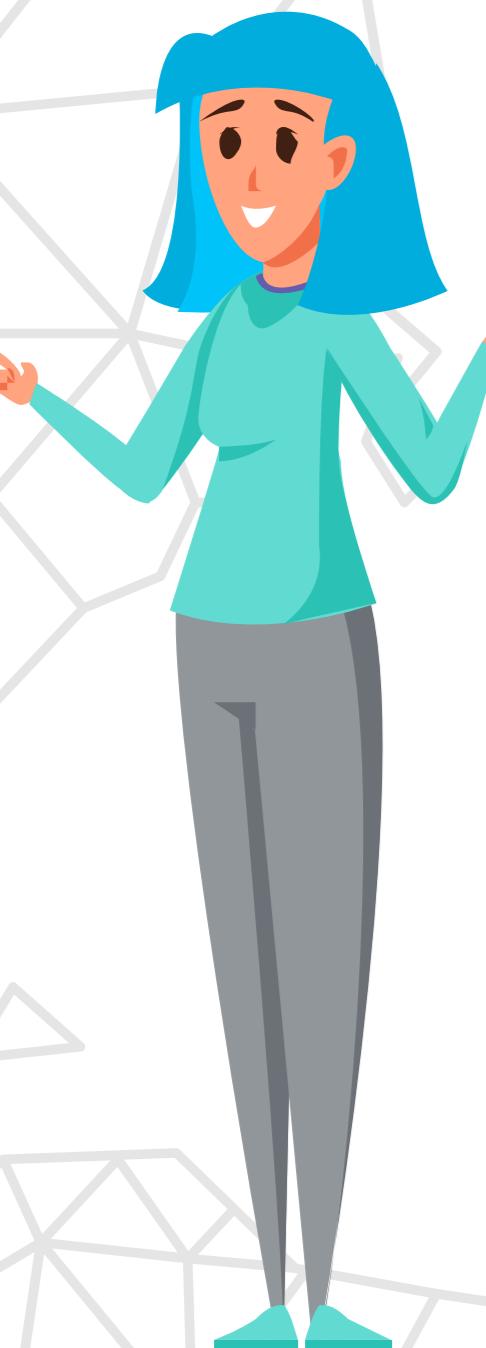
TriggerHandler

A screenshot of Visual Studio Code displaying the `OrderTriggerHandler.cls` file. The code implements a trigger handler for the `Order` object. It defines a class `OrderTriggerHandler` with a constructor and a `run` method. The `run` method retrieves new orders and performs calculations based on whether the trigger is before or after the update.

```
public with sharing class OrderTriggerHandler {
    public OrderTriggerHandler(){}
    public void run() {
        List<Order> newOrders = Trigger.new ;
        if (Trigger.isBefore && Trigger.isUpdate) {
            OrderTriggerHelper.calculateNetAmountOrder(newOrders);
        } else if (Trigger.isAfter && Trigger.isUpdate) {
            OrderTriggerHelper.updateListOfAccountsRevenueWhenOrderStatusActivated(newOrders, (Map<Id, Order>) T)
        }
    }
}
```

Correction du Trigger

Bonnes Pratiques
1 trigger par objet



TriggerHelper

The screenshot shows a Visual Studio Code interface with the following details:

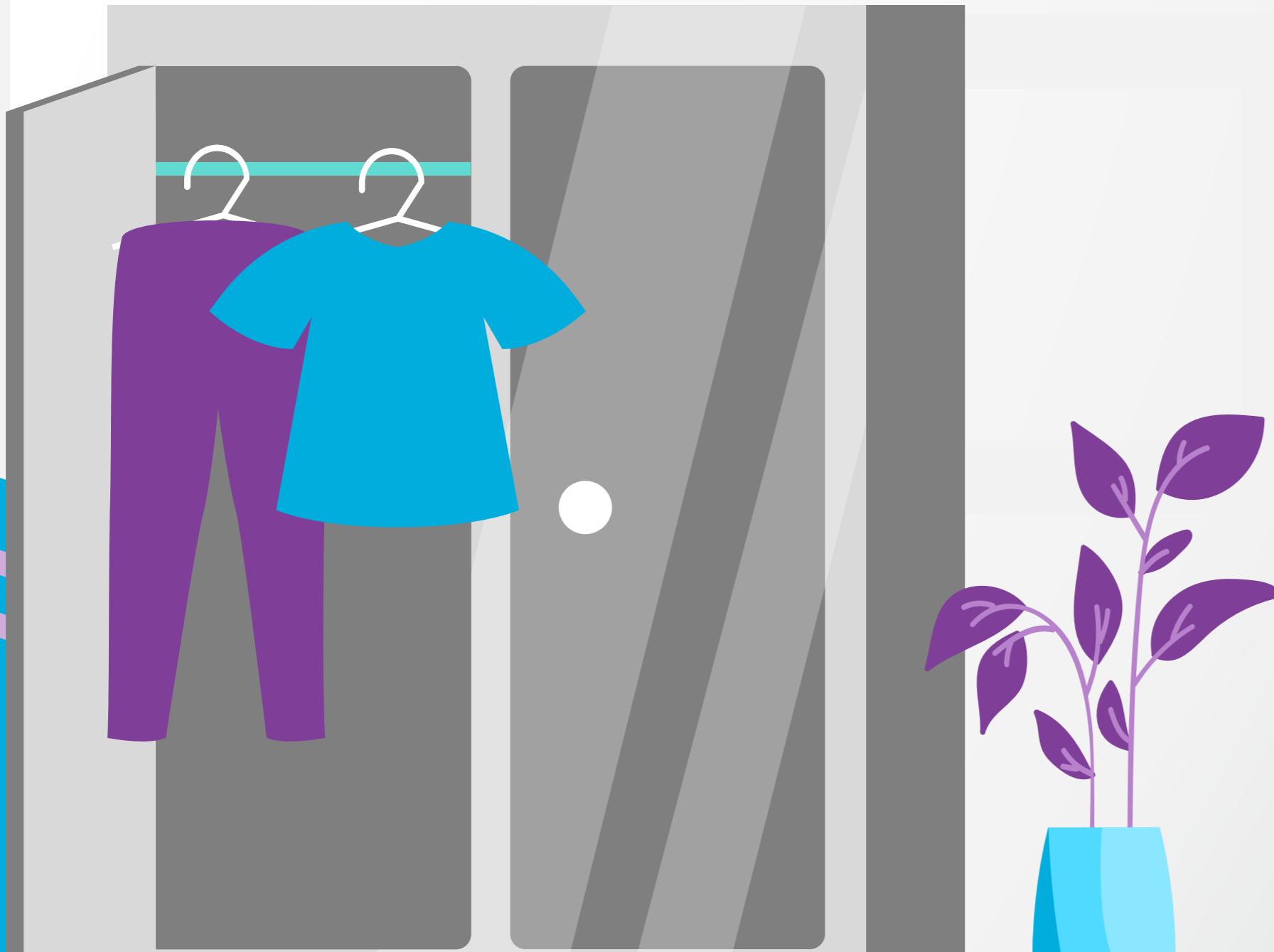
- Title Bar:** OrderTriggerHelper.cls - Fasha_OC_P9 - Visual Studio Code
- File Explorer:** Shows a project structure under "FASHA_OC_P9":
 - OrderTriggerHelper.cls
 - OrderTriggerHelper.cls-meta.xml
 - OrderTriggerHelperTest.cls
 - OrderTriggerHelperTest.cls-meta.xml
 - OrderTriggerTest.cls
 - OrderTriggerTest.cls-meta.xml
 - PricebookSelector.cls
 - PricebookSelector.cls-meta.xml
 - PricebookServices.cls
 - PricebookServices.cls-meta.xml
 - PricebookServicesTest.cls
 - PricebookServicesTest.cls-meta.xml
 - ProductSelector.cls
 - ProductSelector.cls-meta.xml
 - ProductServicesTest.cls
 - ProductServicesTest.cls-meta.xml
 - TeamOrderSumControllerTest.cls
 - TeamOrderSumControllerTest.cls-meta.xml
 - TestDataFactory.cls
 - TestDataFactory.cls-meta.xml
- Code Editor:** The "OrderTriggerHelper.cls" tab is open, displaying Apex code:

```
public static void updateListOfAccountsRevenueWhenOrderStatusActivated(List<Order> orderListNew, Map<String, Order> oldMap) {
    List<Order> updateOrderWithStatusActivated = new List<Order>();
    Set<Id> accountIds = new Set<Id>();
    for (Order orderNew : orderListNew) {
        if (oldMap.get(orderNew.Id).Status != 'Activated' && orderNew.Status == 'Activated') {
            updateOrderWithStatusActivated.add(orderNew);
        }
    }
    for (Order order : updateOrderWithStatusActivated) {
        accountIds.add(order.AccountId);
    }
    List<Account> accounts = AccountSelector.getAccountByIds(accountIds);
    accounts = calculateChiffreAffaireAccount(updateOrderWithStatusActivated, accounts);
    update accounts;
}

public static void calculateNetAmountOrder(List<Order> orders){
    for (Order order : orders) {
        if (order.ShippingCost__c != null){
            order.NetAmount__c = order.TotalAmount - order.ShippingCost__c;
        } else {
            order.ShippingCost__c = 0;
            order.NetAmount__c = order.TotalAmount - order.ShippingCost__c;
        }
    }
}
```
- Bottom Status Bar:** main* 0 0 △ 0 vscodeOrg Git Graph
- Bottom Right:** Salesforce CLI, Ln 35, Col 14, Spaces: 4, UTF-8, LF, Apex, Go



2



Projet mené pour la Société Fasha



Le Batch

Création de la mise à jour en Batch

Start method
Database.QueryLocator

```
global Database.QueryLocator start(Database.BatchableContext bc) {  
    return Database.getQueryLocator(  
        [ SELECT Id, Chiffre_affaire__c FROM Account WHERE Id IN (SELECT AccountId FROM Order WHERE Status =  
        'Activated'))];  
}
```

1
SELECT



Création de la mise à jour en Batch

Execute method
Database.BatchableContext

```
global void execute(Database.BatchableContext bc, List<Account> scope) {  
    Set<Id> sAccountTurnoverToUpdate = (new Map<Id, SObject>(scope)).keySet();  
    NetAmountCalculator.calculateAmount(sAccountTurnoverToUpdate);  
}
```

2

Database.BatchableContext
→ List<Account> scope



Création de la mise à jour en Batch

Finish method
`finish()`

```
global void finish(Database.BatchableContext bc) {  
}
```

3

Finish

`(Database.BatchableContext bc)`



Création de la mise à jour en Batch

AccountProcessorBatch
Traitement des données en masse



Finish

(**Database.BatchableContext bc**)

The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** AccountProcessorBatch.cls - Fasha_OC_P9 - Visual Studio Code
- File Explorer:** Shows the project structure under "FASHA_OC_P9": .vscode, config, force-app\main\default\applications, aura, classes (AccountProcessorBatch.cls is selected), and various test and meta files.
- Code Editor:** Displays the Apex code for AccountProcessorBatch.cls:

```
global class AccountProcessorBatch implements Database.Batchable<sObject> {
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator(
            [ SELECT Id, Chiffre_affaire__c FROM Account WHERE Id IN (SELECT AccountId FROM Order WHERE Status = 'Activated')]);
    }

    global void execute(Database.BatchableContext bc, List<Account> scope) {
        Set<Id> sAccountTurnoverToUpdate = (new Map<Id, SObject>(scope)).keySet();
        NetAmountCalculator.calculateAmount(sAccountTurnoverToUpdate);
    }

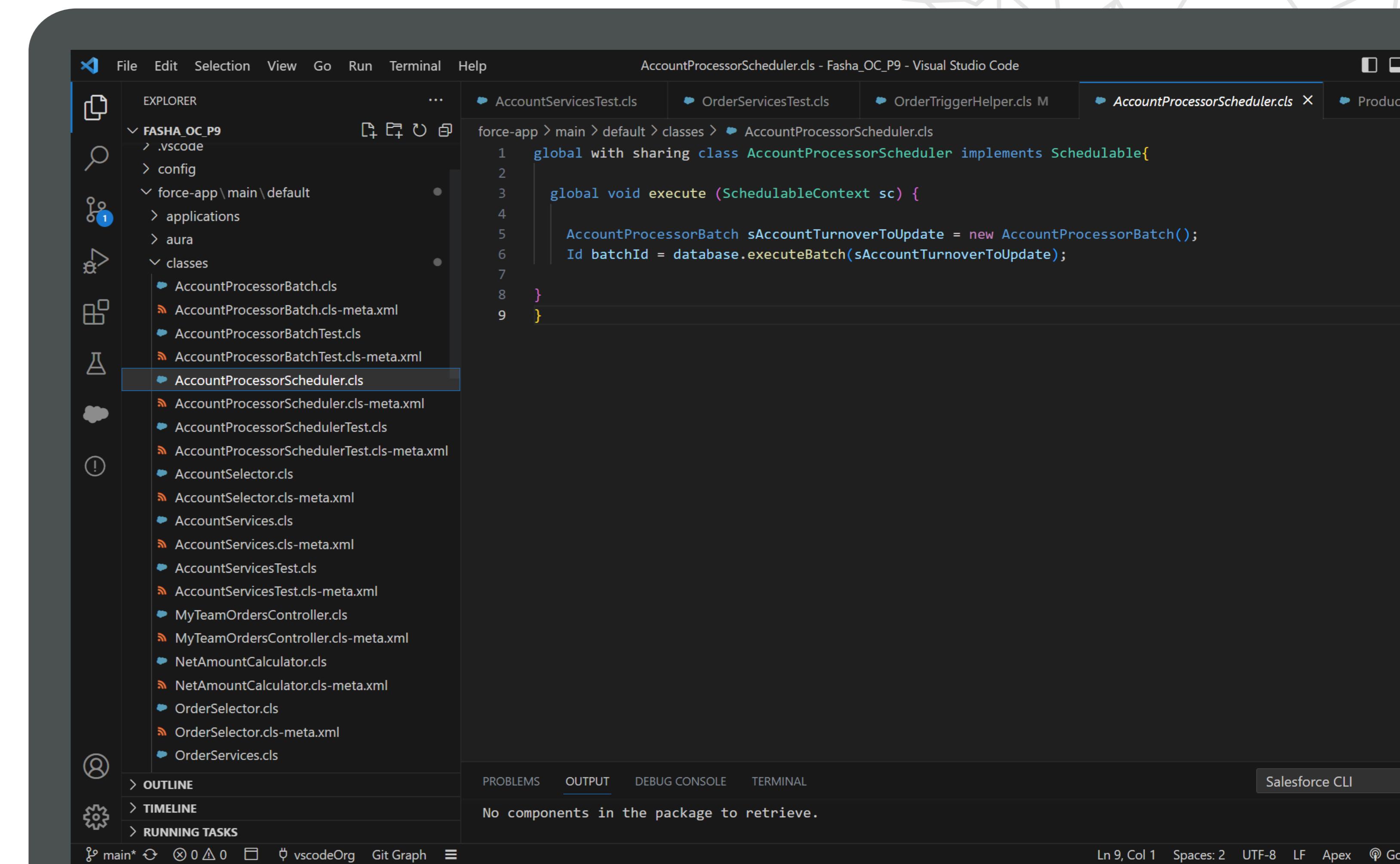
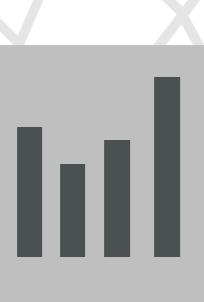
    global void finish(Database.BatchableContext bc) {
    }
}
```
- Bottom Status Bar:** Salesforce CLI, Ln 17, Col 2, Spaces: 4, UTF-8, LF, Apex, Go Live

Création du Scheduler

Account Processor Scheduler
Execute



database.executeBatch
(sAccountTurnoverToUpdate)



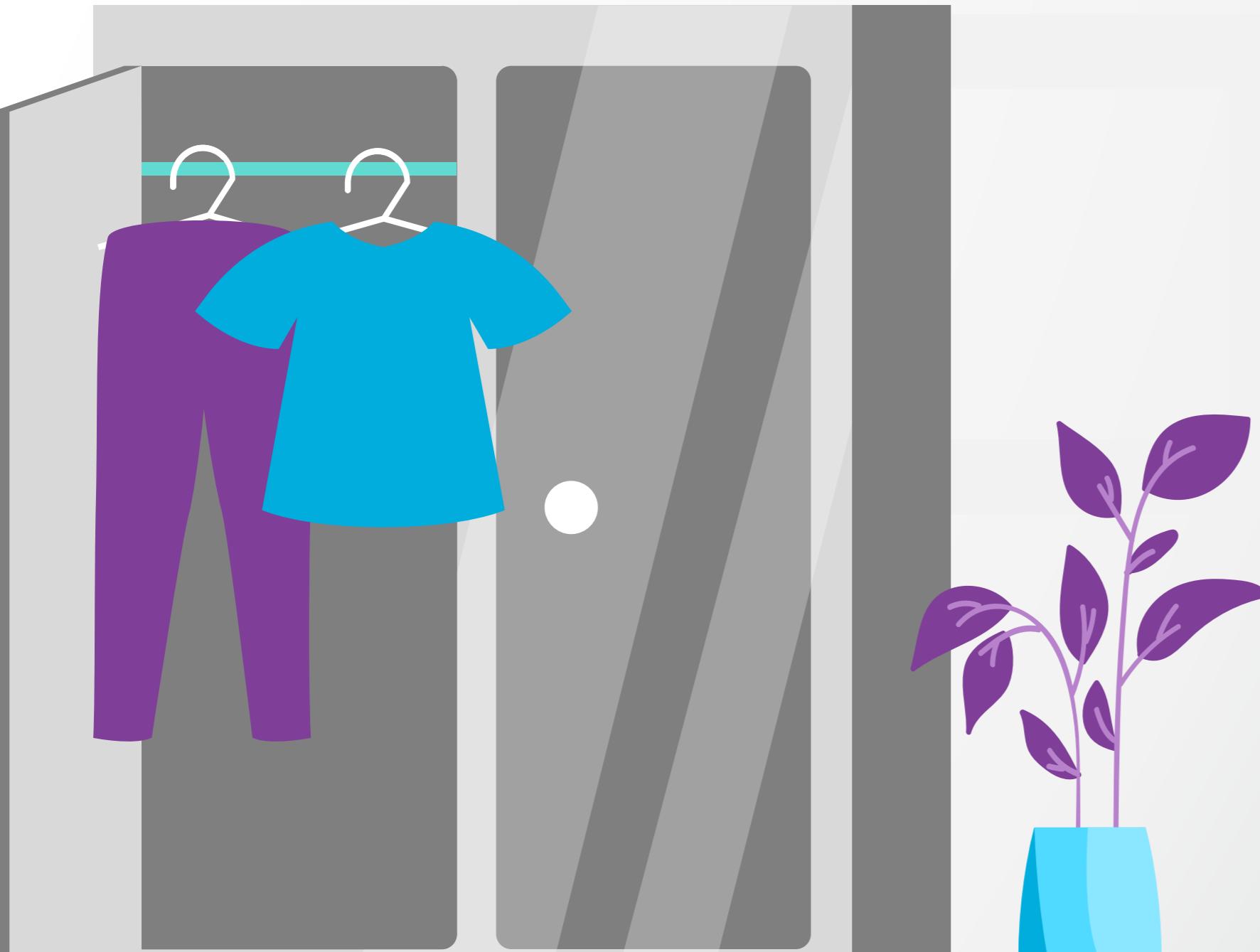
The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** AccountProcessorScheduler.cls - Fasha_OC_P9 - Visual Studio Code
- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Explorer:** Shows the project structure under "FASHA_OC_P9":
 - force-app > main > default > classes > AccountProcessorScheduler.cls
 - AccountServicesTest.cls
 - OrderServicesTest.cls
 - OrderTriggerHelper.cls M
 - AccountProcessorScheduler.cls (selected)
 - AccountProcessorBatch.cls
 - AccountProcessorBatch.cls-meta.xml
 - AccountProcessorBatchTest.cls
 - AccountProcessorBatchTest.cls-meta.xml
 - AccountProcessorScheduler.cls
 - AccountProcessorScheduler.cls-meta.xml
 - AccountProcessorSchedulerTest.cls
 - AccountProcessorSchedulerTest.cls-meta.xml
 - AccountSelector.cls
 - AccountSelector.cls-meta.xml
 - AccountServices.cls
 - AccountServices.cls-meta.xml
 - AccountServicesTest.cls
 - AccountServicesTest.cls-meta.xml
 - MyTeamOrdersController.cls
 - MyTeamOrdersController.cls-meta.xml
 - NetAmountCalculator.cls
 - NetAmountCalculator.cls-meta.xml
 - OrderSelector.cls
 - OrderSelector.cls-meta.xml
 - OrderServices.cls
- Editor:** Displays the code for AccountProcessorScheduler.cls:

```
global with sharing class AccountProcessorScheduler implements Schedulable{
    global void execute (SchedulableContext sc) {
        AccountProcessorBatch sAccountTurnoverToUpdate = new AccountProcessorBatch();
        Id batchId = database.executeBatch(sAccountTurnoverToUpdate);
    }
}
```
- Bottom Status Bar:** Salesforce CLI, Ln 9, Col 1, Spaces: 2, UTF-8, LF, Apex, Git Graph



3



Projet mené pour la Société Fasha



Les Classes

Nomenclature des Classes

Des Classes Spécialisées

Les Classes Selector

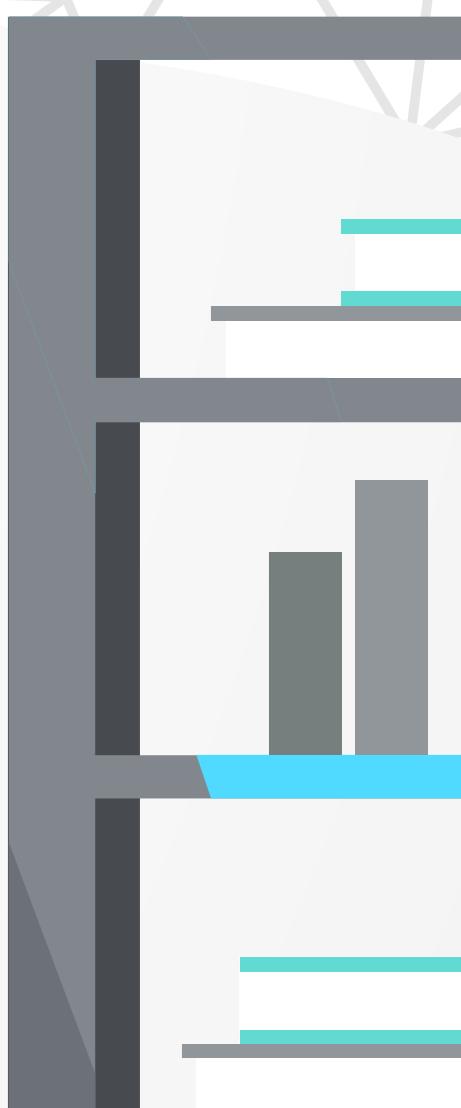
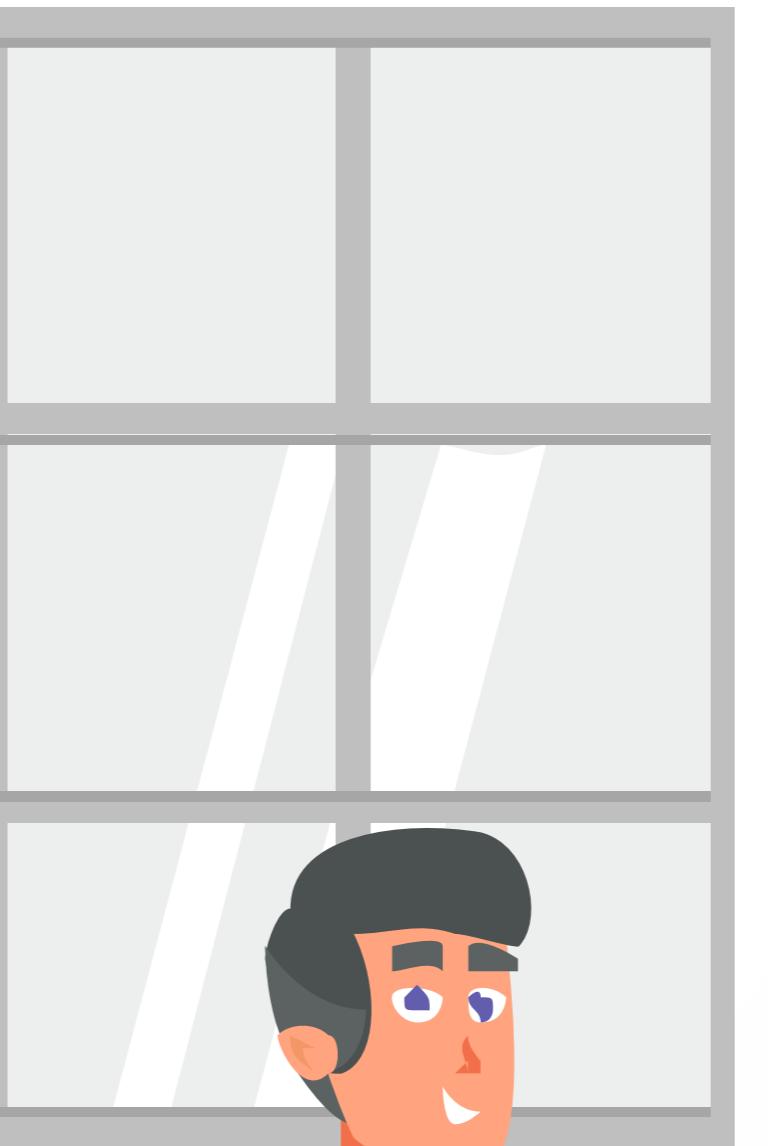
AccountSelector

OrderSelector

PricebookSelector

Sélection de données (GET)

- getAccount()
- getOrder()
- getPricebook()



Nomenclature des Classes

Des Classes Spécialisées

Les Classes Services

AccountServices

OrderServices

PricebookServices

AccountProcessorBatch

NetAmountCalculator

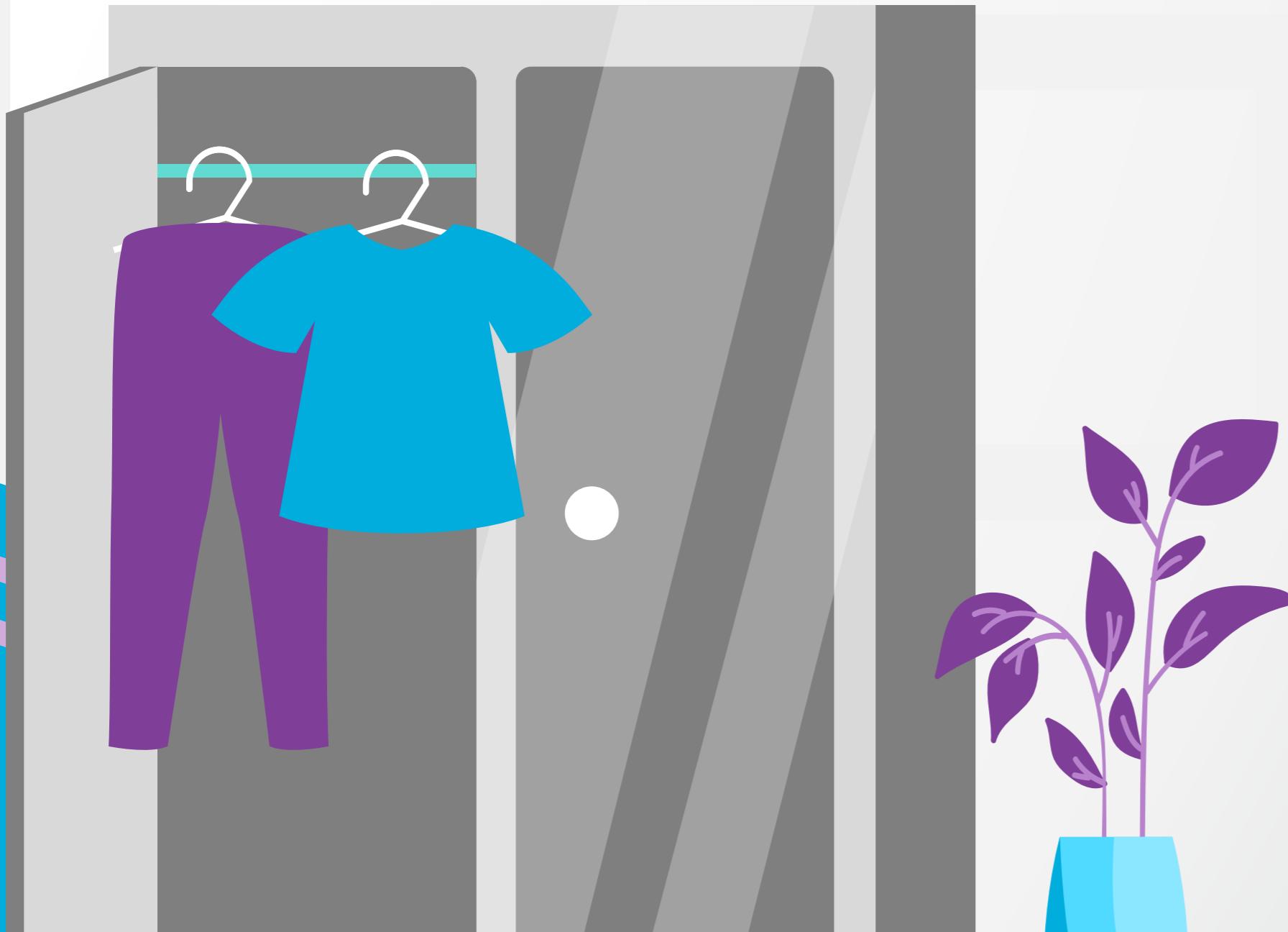
Traitement sur les données

- updateOrderStatus()
- updateAccountNetAmount()





4



Projet mené pour la Société Fasha



Les Tests

Les tests & Best Practices

La création des données de Test
Test data Factory

- createAccount()
- createOrder()
- createPricebook()
- createProduct()
- ...



Les tests & Best Practices

La nomenclature des classes

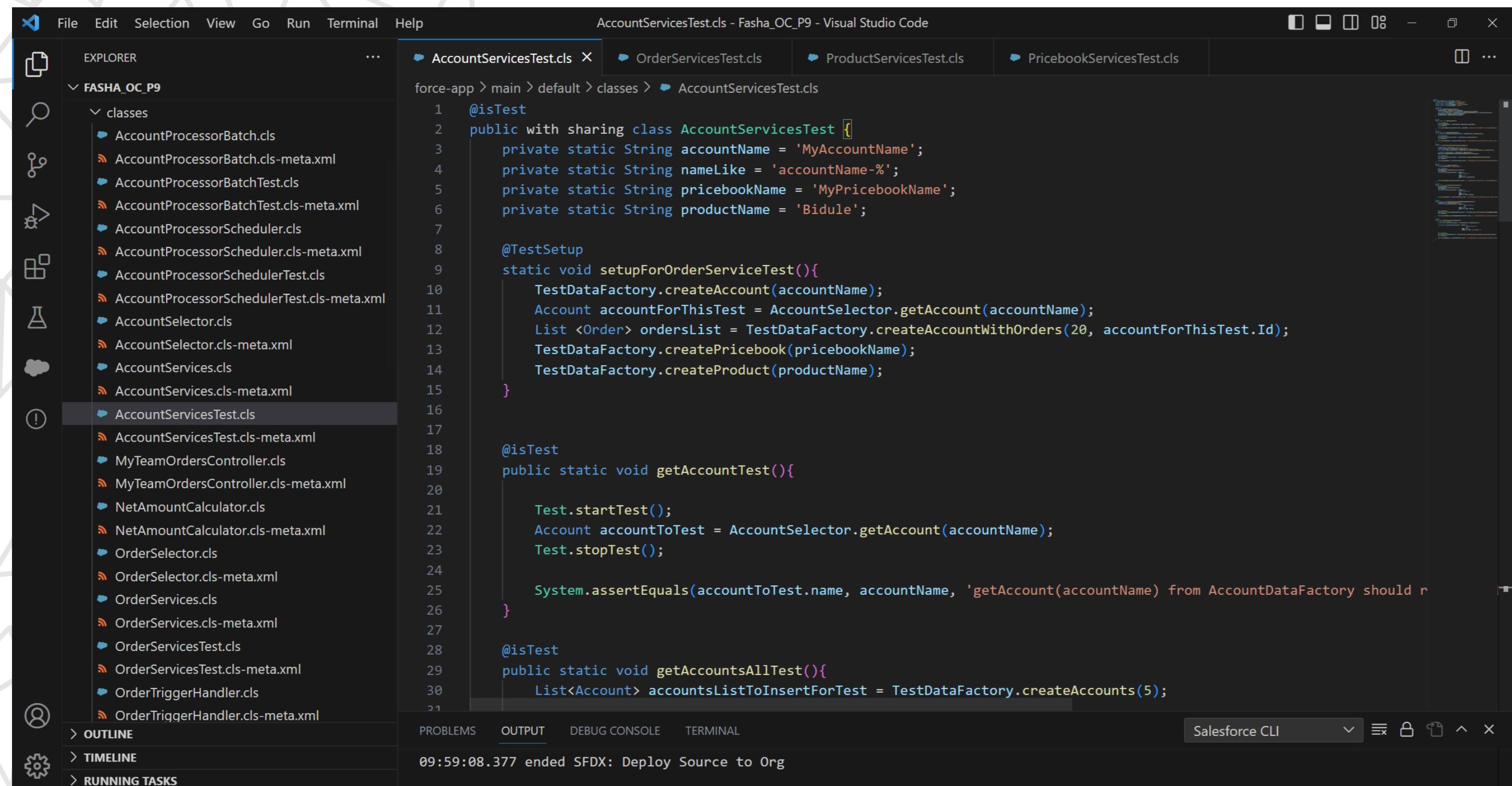
Une classe Test pour une classe Services

- OrderServicesTest
- PricebookServicesTest
- AccountServicesTest
- AccountProcessorBatchTest
- AccountProcessorSchedulerTest
- ...



Les tests & Best Practices

Les Best Practices des Tests Apex La syntaxe



```
File Edit Selection View Go Run Terminal Help
AccountServicesTest.cls - Fasha_OC_P9 - Visual Studio Code
force-app > main > default > classes > AccountServicesTest.cls
AccountServicesTest.cls X OrderServicesTest.cls ProductServicesTest.cls PricebookServicesTest.cls
EXPLORER
FASHA_OC_P9
  classes
    AccountProcessorBatch.cls
    AccountProcessorBatch.cls-meta.xml
    AccountProcessorBatchTest.cls
    AccountProcessorBatchTest.cls-meta.xml
    AccountProcessorScheduler.cls
    AccountProcessorScheduler.cls-meta.xml
    AccountProcessorSchedulerTest.cls
    AccountProcessorSchedulerTest.cls-meta.xml
    AccountSelector.cls
    AccountSelector.cls-meta.xml
    AccountServices.cls
    AccountServices.cls-meta.xml
    AccountServicesTest.cls
    AccountServicesTest.cls-meta.xml
    MyTeamOrdersController.cls
    MyTeamOrdersController.cls-meta.xml
    NetAmountCalculator.cls
    NetAmountCalculator.cls-meta.xml
    OrderSelector.cls
    OrderSelector.cls-meta.xml
    OrderServices.cls
    OrderServices.cls-meta.xml
    OrderServicesTest.cls
    OrderServicesTest.cls-meta.xml
    OrderTriggerHandler.cls
    OrderTriggerHandler.cls-meta.xml
    OUTLINE
    TIMELINE
    RUNNING TASKS
@isTest
public with sharing class AccountServicesTest {
    private static String accountName = 'MyAccountName';
    private static String nameLike = 'accountName-%';
    private static String pricebookName = 'MyPricebookName';
    private static String productName = 'Bidule';

    @TestSetup
    static void setupForOrderServiceTest(){
        TestDataFactory.createAccount(accountName);
        Account accountForThisTest = AccountSelector.getAccount(accountName);
        List<Order> ordersList = TestDataFactory.createAccountWithOrders(20, accountForThisTest.Id);
        TestDataFactory.createPricebook(pricebookName);
        TestDataFactory.createProduct(productName);
    }

    @isTest
    public static void getAccountTest(){

        Test.startTest();
        Account accountToTest = AccountSelector.getAccount(accountName);
        Test.stopTest();

        System.assertEquals(accountToTest.name, accountName, 'getAccount(accountName) from AccountDataFactory should return the correct account');
    }

    @isTest
    public static void getAccountsAllTest(){
        List<Account> accountsListToInsertForTest = TestDataFactory.createAccounts(5);
    }
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Salesforce CLI
09:59:08.377 ended SFDX: Deploy Source to Org
```

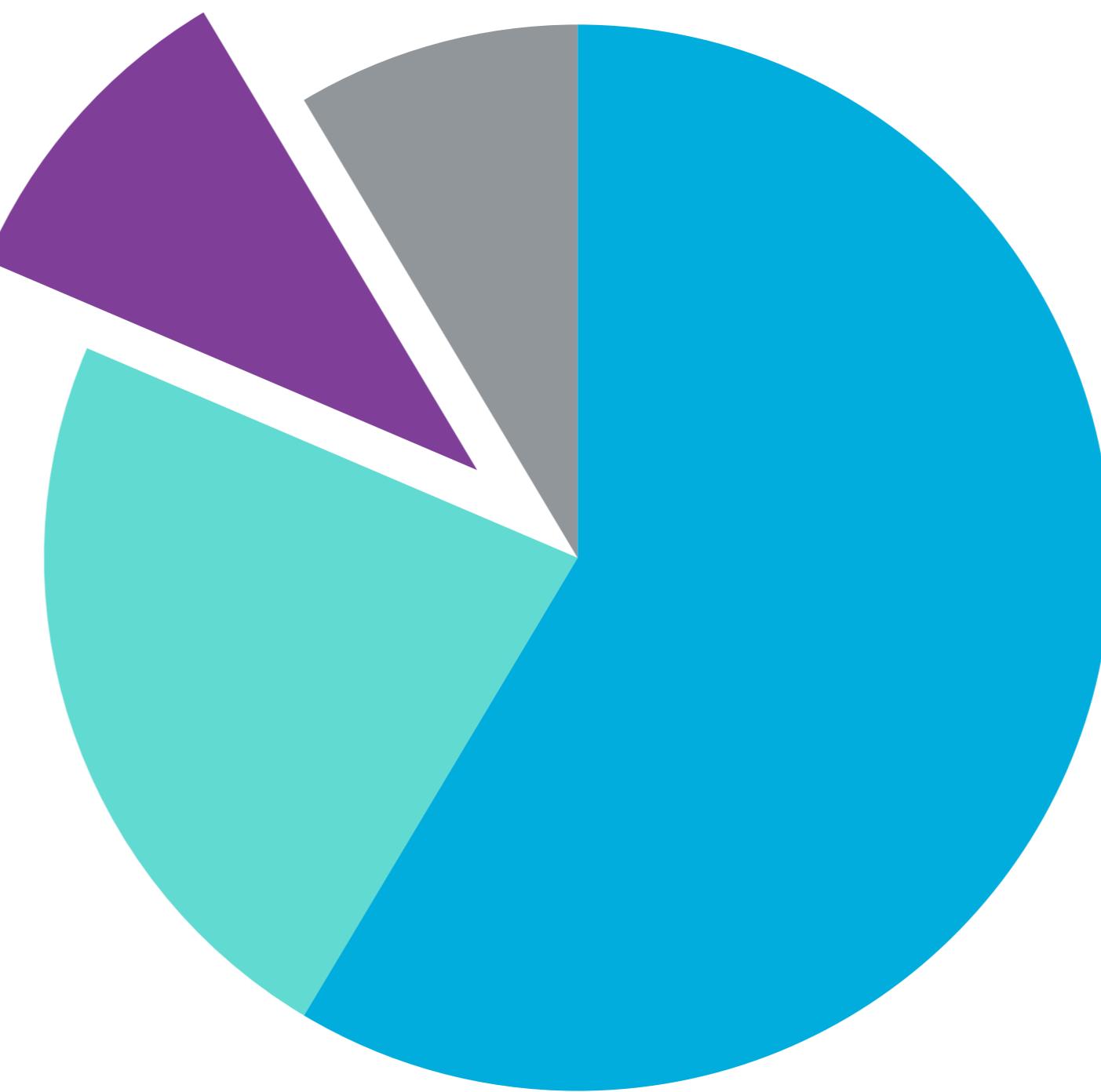
1. Utiliser **@testSetup** pour générer les données nécessaires pour chaque classe de test
3. Démarrer chaque test par **Test.startTest()**
4. Terminer chaque test par **Test.stopTest()**
5. Utiliser les instructions **System.assert()** après **TestStopTest** pour vérifier les données attendues

Les tests & Best Practices

Les Tests Fonctionnels

Couverture du code au moins à 75%

Overall Code Coverage		
Class	Percent	Lines
Overall	87%	
AccountProcessorBatch	100%	7/7
AccountProcessorScheduler	100%	3/3
AccountSelector	100%	12/12
AccountServices	100%	10/10
MyTeamOrdersController	100%	5/5
NetAmountCalculator	100%	9/9
OrderSelector	61%	13/21
OrderServices	100%	15/15
OrderTrigger	100%	1/1
OrderTriggerHandler	100%	7/7
OrderTriggerHelper	75%	34/45
PricebookSelector	100%	11/11
PricebookServices	100%	6/6
ProductSelector	100%	4/4



Les tests & Best Practices

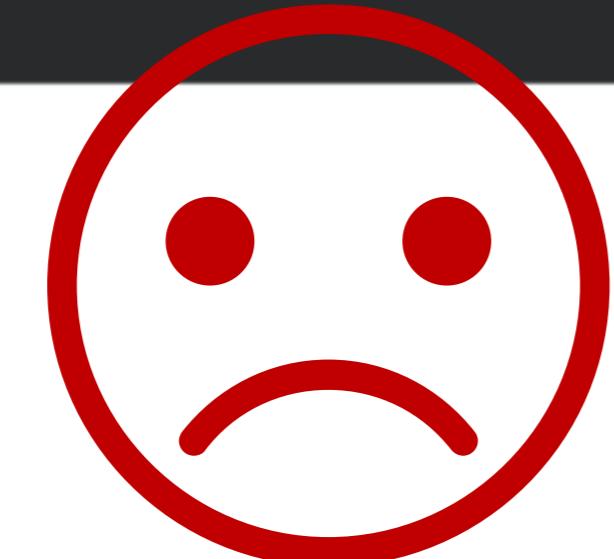
Les Bests Practices des Classes et des Tests

Ecriture de code optimisé pour le traitement de données de masse



Tous les tests et le code doivent marcher pour des données en bulk (avec un nb infini d'enregistrement) : pour se faire on crée un liste on stocke l'objet dans la liste et on récupère la liste d'objets)

```
trigger OpportunityTrigger on Opportunity(before insert) {  
    Opportunity opp = Trigger.new[0];  
  
    if(opp.Amount > 10000){  
        assignSalesLeaderOwner(opp);  
    }  
}
```



```
trigger OpportunityTrigger on Opportunity(before insert) {  
    for (Opportunity opp : Trigger.new) {  
        if (opp.Amount > 10000) {  
            assignSalesLeaderOwner(opp);  
        }  
    }  
}
```





Magdalena Larmet
Salesforce
Developer

