

### Fields & Object

Object - Without underscore (\_) like - ProductFeature\_\_c

Fields - without underscore (\_) like - isVisible\_\_c

Fields - in Master-detail or lookup relationship append "Ref" - AccountRef\_\_c

Fields - in roll-up summary field append "RS" - NumberOfContactsRS\_\_c

Fields - in formula fields append "F" - ActiveStatusF\_\_c

### Flow

Name should be enough descriptive and tell its purpose and type of flow - ProductSelectionScreenFL

Scheduled flow - SendBirthdayEmailsBatchFL

Record triggers process one record at a time, name should be - AccountCreateFL, AccountUpdateFL

Auto Launch flow - OpportunitySyncAutolaunchFL

Platform event flow - PaymentReceivedEventFL

### Lightning web components

Name should be descriptive and follow camelCase, eg: productSelectionTable.html, activityTimeline.html

Variables should also follow camelCase. eg: leadStatus, caseOrigin

Java script and css file should also follow camelCase. eg: productSelectionTable.js, activityTimeline.js, productSelectionTable.css, activityTimeline.css

### Custom Metadata

Add description and define its purpose of use.

Use Custom Metadata and apply checks in trigger, flow etc to activate and deactivate any automation

### Variables and Methods

Method name should be verb and in camelCase eg: getAccountsByName, applyDiscount

Apex test method name eg: unitTest\_getAccountsByName

Variables - local instance and class variables should follow camelCase. eg: isErrorLoggingEnabled

Constants should be written in UPPER CASE and words separated by underscore(\_). eg: MAXIMUM\_PARTICIPANTS

### Apex Test Execution

Do not use @SeeData = true in test classes

Always write asserts and cover positive and negative scenarios.

Test single and bulk records(200)

Try to cover 100% code coverage including exception scenarios as well.

Use mock class for callout scenarios.

### Custom Labels

Constants should be defined in CAPS letters for ex: INVALID\_EMAIL\_ERROR\_MESSAGE



### Apex & Trigger

Class name should be Descriptive Noun or Noun Phrase using PascalCase

Controller - NewAccountWizardCtrl

Extension - NewAccountWizardCtrlExt

Domain - Account Domain (object specific logic)

Trigger - AccountTrigger (Object specific triggers)

Trigger Handler - AccountTriggerHandler (Object specific trigger handler)

Service - AccountService (Functionality specific service)

Batch - AccountArchiveBatch

Schedule - AccountArchiveScheduler

Custom REST - AccountRestResource

SOAP Resource - AccountSOAPResource

Selector - AccountSelector (Object Specific Query class)

Test class - Main Class name + "Test", AccountServiceTest

### Lightning Components (Aura)

Descriptive name using PascalCase. eg: ProductSelectionTable, ActivityTimeline

Js file, helper, Event and Application name should also use PascalCase. eg: ConfigureQuoteApp, ProductSelectionEvent

Variable name should be in camelCase. eg: recordId, contactName, isModalOpen,

### Version control and VS Code

Always use VS Code to commit code in repository

Keep branch name short and descriptive. eg: feature/JIRA-TICKET-ID.

Commits- message should define the functionality and only commit your changes in branch.

Install VS Code extension like Salesforce CLI, SFDX and Salesforce extension pack

### Apex Structure and readability

Max 100 characters in single line

max 1000 lines per apex class

max 4 input parameters per method

remove unwanted space and white spaces

Every class must have its test class

Tab width should be 4 white spaces long

Remove all debug statements after completing the functionality

Always include curly braces { } to indicate the start and end of looping and branching constructs

keep method definition short up-to 80 lines

Use Controller, Service, Domain, Selector pattern to implement Separation of concern(SOC).

Use Javadoc style of code commenting

