What is an Interface?

An interface is like an Apex class in which none of the methods have been implemented. It only contains the method signatures, but the body of each method is empty. To use an interface, another class must implement it by providing a body for all of the methods contained in the interface.

Interfaces are used mainly for providing the abstraction layer for your code. They separate the implementation from declaration of the method.

Let's take an example of our Chemical Company. Suppose that we need to provide the discount to Premium and Ordinary customers and discounts for both will be different.

We will create an Interface called as DiscountProcessor.

```
//Interface
public interface DiscountProcessor{
Double percentageDiscountTobeApplied();//method signature only
}

//Premium Customer Class
public class PremiumCustomer implements DiscountProcessor{
 //Method Call
 public Double percentageDiscountTobeApplied () {
  //For Premium customer, discount should be 30%
  return 0.30;
 }
}

//Normal Customer Class
public class NormalCustomer implements DiscountProcessor{
 //Method Call
 public Double percentageDiscountTobeApplied () {
  //For Premium customer, discount should be 10%
  return 0.10;
 }
}
```

When you implement the Interface then it is mandatory to implement the method of that Interface. If you don't implement the Interface methods, it will throw an error. You should use Interfaces when you want to make the method implementation mandatory for developer.

Standard Salesforce Interface for Batch Apex

SFDC do have standard interfaces like Database.Batchable, Schedulable, etc. For example, if you implement the Database.Batchable Interface, then you must implement the three methods defined in the Interface: Start, Execute and Finish.

Below is the example for standard Salesforce provided Database.Batchable Interface which sends out emails to users with Batch Status. This interface has 3 methods, Start, Execute and Finish. Using this interface, we can implement the Batchable functionality and it does provide the BatchableContext variable which we can use to get more information about the Batch which is executing and to perform other functionalities.

```
global class CustomerProessingBatch implements Database.Batchable<sobject>, Schedulable{
//Add here your email address
global String [] email = new String[] {'test@test.com'};

//Start Method
global Database.Querylocator start (Database.BatchableContext BC) {
    //This is the Query which will determine the scope of Records and fetching the same
```

```
    return Database.getQueryLocator('Select id, Name, APEX_Customer_Status__c,
APEX_Customer_Decscription__c From APEX_Customer__c WHERE createdDate = today &&
APEX_Active__c = true');
}

//Execute method
global void execute (Database.BatchableContext BC, List<sobject> scope) {
    List<apex_customer__c> customerList = new List<apex_customer__c>();
    List<apex_customer__c> updtaedCustomerList = new List<apex_customer__c>();
    for (sObject objScope: scope) {
     //type casting from generic sOject to APEX_Customer__c
        APEX_Customer__c newObjScope = (APEX_Customer__c)objScope ;
        newObjScope.APEX_Customer_Decscription__c = 'Updated Via Batch Job';
        newObjScope.APEX_Customer_Status__c = 'Processed';
     //Add records to the List
        updtaedCustomerList.add(newObjScope);
    }

    //Check if List is empty or not
    if (updtaedCustomerList != null && updtaedCustomerList.size()>0) {
  //Update the Records
        Database.update(updtaedCustomerList); System.debug('List Size
'+updtaedCustomerList.size());
    }
}

//Finish Method
global void finish(Database.BatchableContext BC){
    Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();

    //get the job Id
    AsyncApexJob a = [Select a.TotalJobItems, a.Status, a.NumberOfErrors, a.JobType,
a.JobItemsProcessed, a.ExtendedStatus, a.CreatedById, a.CompletedDate From AsyncApexJob a
WHERE id = :BC.getJobId()];
    System.debug('$$$ Jobid is'+BC.getJobId());

    //below code will send an email to User about the status
    mail.setToAddresses(email);

    //Add here your email address
    mail.setReplyTo('test@test.com');
    mail.setSenderDisplayName('Apex Batch Processing Module');
    mail.setSubject('Batch Processing '+a.Status);
    mail.setPlainTextBody('The Batch Apex job processed  '+a.TotalJobItems+'batches with
'+a.NumberOfErrors+'failures'+'Job Item processed are'+a.JobItemsProcessed);

    Messaging.sendEmail(new Messaging.Singleemailmessage [] {mail});
}

//Scheduler Method to scedule the class
global void execute(SchedulableContext sc){
    CustomerProessingBatch conInstance = new CustomerProessingBatch();
    database.executebatch(conInstance,100);
}
}
```

To execute this class, you have to run the below code in developer console.

```
CustomerProessingBatch objBatch = new CustomerProessingBatch ();
Database.executeBatch(objBatch);
```