Understanding the Data Types

As we have studied, the Apex language is strongly typed so every variable in Apex will be declared with the specific data type. All apex variables are initialized to null initially. As a best practice developer has to make sure it should get assigned proper value otherwise such variables when used, will throw null pointer exceptions or any unhandled expections.

Apex supports the following data-types:

- Primitive *Integer, Double, Long, Date, Datetime, String, ID, orBoolean*
- Collections *Lists, SetsandMaps TobecoveredinChapter*6
- sObject
- Enums
- Classes, Objects and Interfaces *TobecoveredinChapter*11, 12*and*13

In this chapter, we will look at all the Primitive Data Types, sObjects and Enums. We will be looking at Collections, Classes, Objects and Interfaces in upcoming chapters since they are key topics to be learnt individually.

Primitive Data Type

Integer

Any 32 bit number which does not include any decimal point. The value range is -2,147,483,648 and a maximum value of 2,147,483,647.

Example: We want to declare a variable which would store the quantity of barrels which needs to be shipped to buyer of chemical processing plant.

```
Integer barrelNumbers = 1000;
system.debug(' value of barrelNumbers variable: '+barrelNumbers);
```

System.debug is function which prints the value of variable so that we could use this to debug or to get to know what value the variable holds currently.

Paste the above code to Developer console and the click on execute. Once logs are generated then it will show the value of variable "barrelNumbers" as 1000.

Boolean

This variable can either be true, false or null. Many times, this type of variables can be used as flag in programming to identify the particular condition set or not set.

Example: If we would like to set shipmentDispatched as true, then it can be declared as:

```
Boolean shipmentDispatched;
shipmentDispatched = true;
System.debug('Value of shipmentDispatched '+shipmentDispatched);
```

Date

This is the variable of type date. This can only store the date not the time. For saving date along with time we would need to store it in variable of DateTime.

Example:

```
//ShipmentDate can be stored when shipment is dispatched.
Date ShipmentDate = date.today();
```

```
System.debug('ShipmentDate '+ShipmentDate);
```

Long

This is a 64-bit number without a decimal point. Use this data type when you need a range of values wider than those provided by Integer.

Example: If we would like to store the company revenue, then we would use data type as Long.

```
Long companyRevenue  = 21474838973344648L;
system.debug('companyRevenue'+companyRevenue);
```

Object

We can refer this as any data type which is supported in Apex. For example, Class variable can be object of that class, and the sObject generic type is also an object and similarly specific object type like Account is also an Object.

Example:

```
Account objAccount = new Account (Name = 'Test Chemical');
system.debug('Account value'+objAccount);
```

You can create an object of predefined class as well, as given below:

```
//Class Name: MyApexClass
MyApexClass  classObj = new MyApexClass();
```

This is the class object which will be used as class variable. No need to execute this code, this is just for reference.

String

String is any set of characters within single quotes. It does not have limit of number of characters, but the heap size would be used to determine so that Apex program should not monopolize the resources and does not grow too large.

Example:

```
String companyName = 'Abc International';
System.debug('Value companyName variable'+companyName);
```

Time

This variable is used to store the particular time. This variable should always be declared with system static method.

Blob

The Blob is collection of binary data which is stored as object. This will be used when we want to store the attachment in salesforce into a variable. This data type converts the attachments in a single object. When we need to convert the blob into string then we could use toString and valueOf methods to convert it to string when required.

sObject

This is a special data type in Salesforce. It is similar to a table in SQL and contains fields which are similar to columns in SQL. There are two types of sObjects: Standard and Custom.

For example, Account is a standard sObject and any other user defined object *likeCustomerobjectthatwecreated* is Custom sObject.

Example:

```
//Declaring an sObject variable of type Account
```

```
Account objAccount = new Account();

//Assignment of values to fields of sObjects
objAccount.Name = 'ABC Customer';
objAccount.Description = 'Test Account';
System.debug('objAccount variable value'+objAccount);

//Declaring an sObject for custom object APEX_Invoice_c
APEX_Customer_c objCustomer = new APEX_Customer_c();

//Assigning value to fields
objCustomer.APEX_Customer_Decscription_c = 'Test Customer';
System.debug('value objCustomer'+objCustomer);
```

Enum

Enum is an abstract data type that stores one value of a finite set of specified identifiers. You could use the keyword Enum to define an Enum. Enum can be used as any other data type in Salesforce.

Example:

Suppose, you would like to declare the possible names of Chemical Compound, then you could do something like this:

```
//Declaring enum for Chemical Compounds
public enum Compounds {HCL, H2SO4, NACL, HG}
Compounds objC = Compounds.HCL;
System.debug('objC value: '+objC);
```