**Lists of Custom Types and Sorting**

Lists can hold objects of your user-defined types (your Apex classes). Lists of user-defined types can be sorted.

To sort such a list using the List.sort method, your Apex classes must implement the Comparable interface.

The sort criteria and sort order depends on the implementation that you provide for the compareTo method of the Comparable interface. For more information on implementing the Comparable interface for your own classes, see the [Comparable Interface](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_comparable.htm#apex_comparable).

**Comparable Interface**

Adds sorting support for Lists that contain non-primitive types, that is, Lists of user-defined types.

Namespace

[System](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_namespace_System.htm)

Usage

To add List sorting support for your Apex class, you must implement the Comparable interface with its compareTo method in your class.

To implement the Comparable interface, you must first declare a class with the implements keyword as follows:

|  |  |
| --- | --- |
| 1 | global class Employee implements Comparable { |

Next, your class must provide an implementation for the following method:

|  |  |  |
| --- | --- | --- |
| 1 | global Integer compareTo(Object compareTo) { | |
| 2 | // Your code here |

|  |  |
| --- | --- |
| 3 | } |

The implemented method must be declared as global or public.

* [**Comparable Methods**](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_comparable.htm#apex_System_Comparable_methods)
* [**Comparable Example Implementation**](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_comparable.htm#apex_comparable_example)

**See Also**

* [List Class](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_methods_system_list.htm#apex_methods_system_list)

Comparable Methods

The following are methods for Comparable.

* [**compareTo(objectToCompareTo)**](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_comparable.htm#apex_System_Comparable_compareTo)  
  Returns an Integer value that is the result of the comparison.

**compareTo(objectToCompareTo)**

Returns an Integer value that is the result of the comparison.

**Signature**

public Integer compareTo(Object objectToCompareTo)

**Parameters**

*objectToCompareTo*

Type: Object

**Return Value**

Type: [Integer](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_methods_system_integer.htm#apex_methods_system_integer)

**Usage**

The implementation of this method returns the following values:

* 0 if this instance and *objectToCompareTo* are equal
* > 0 if this instance is greater than *objectToCompareTo*
* < 0 if this instance is less than *objectToCompareTo*

If this object instance and *objectToCompareTo* are incompatible, a System.TypeException is thrown.

Comparable Example Implementation

This is an example implementation of the Comparable interface. The compareTo method in this example compares the employee of this class instance with the employee passed in the argument. The method returns an Integer value based on the comparison of the employee IDs.

|  |  |  |
| --- | --- | --- |
| 01 | global class Employee implements Comparable { | |
| 02 |  |

|  |  |
| --- | --- |
| 03 | public Long id; |
| 04 | public String name; | |

|  |  |  |
| --- | --- | --- |
| 05 | public String phone; | |
| 06 |  |

|  |  |
| --- | --- |
| 07 | // Constructor |
| 08 | public Employee(Long i, String n, String p) { | |

|  |  |
| --- | --- |
| 09 | id = i; |
| 10 | name = n; | |

|  |  |  |
| --- | --- | --- |
| 11 | phone = p; | |
| 12 | } |

|  |  |
| --- | --- |
| 13 |  |
| 14 | // Implement the compareTo() method | |

|  |  |
| --- | --- |
| 15 | global Integer compareTo(Object compareTo) { |
| 16 | Employee compareToEmp = (Employee)compareTo; | |

|  |  |  |
| --- | --- | --- |
| 17 | if (id == compareToEmp.id) return 0; | |
| 18 | if (id > compareToEmp.id) return 1; |

|  |  |  |
| --- | --- | --- |
| 19 | return -1; | |
| 20 | } |

|  |  |
| --- | --- |
| 21 | } |

This example tests the sort order of a list of Employee objects.

[view source](https://developer.salesforce.com/docs/#viewSource)

[print](https://developer.salesforce.com/docs/#printSource)[?](https://developer.salesforce.com/docs/#about)

|  |  |
| --- | --- |
| 01 | @isTest |
| 02 | private class EmployeeSortingTest { | |

|  |  |
| --- | --- |
| 03 | static testmethod void test1() { |
| 04 | List<Employee> empList = new List<Employee>(); | |

|  |  |  |
| --- | --- | --- |
| 05 | empList.add(new Employee(101,'Joe Smith', '4155551212')); | |
| 06 | empList.add(new Employee(101,'J. Smith', '4155551212')); |

|  |  |  |
| --- | --- | --- |
| 07 | empList.add(new Employee(25,'Caragh Smith', '4155551000')); | |
| 08 | empList.add(new Employee(105,'Mario Ruiz', '4155551099')); |

|  |  |
| --- | --- |
| 09 |  |
| 10 | // Sort using the custom compareTo() method | |

|  |  |  |
| --- | --- | --- |
| 11 | empList.sort(); | |
| 12 |  |

|  |  |  |
| --- | --- | --- |
| 13 | // Write list contents to the debug log | |
| 14 | System.debug(empList); |

|  |  |
| --- | --- |
| 15 |  |
| 16 | // Verify list sort order. | |

|  |  |  |
| --- | --- | --- |
| 17 | System.assertEquals('Caragh Smith', empList[0].Name); | |
| 18 | System.assertEquals('Joe Smith', empList[1].Name); |

|  |  |
| --- | --- |
| 19 | System.assertEquals('J. Smith', empList[2].Name); |
| 20 | System.assertEquals('Mario Ruiz', empList[3].Name); | |

|  |  |  |
| --- | --- | --- |
| 21 | } | |
| 22 | } |