



Comparable Interface

Adds sorting support for Lists that contain non-primitive types, that is, Lists of user-defined types. Your implementation must explicitly handle null inputs in the `compareTo()` method to avoid a null pointer exception.

Namespace

[System](#)

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:

```
public class Employee implements Comparable {
```

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

```
public Integer compareTo(Object compareTo) {  
    // Your code here  
}
```

The implemented method must be declared as `global` or `public`.

- [Comparable Methods](#)
- [Comparable Example Implementation](#)

See Also

- [List Class](#)

Comparable Methods

The following are methods for `Comparable`.

- [compareTo\(objectToCompareTo\)](#)
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



Type: Object

Return Value

Type: [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 example implements 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.

```
public class Employee implements Comparable {

    public Long id;
    public String name;
    public String phone;

    // Constructor
    public Employee(Long i, String n, String p) {
        id = i;
        name = n;
        phone = p;
    }

    // Implement the compareTo() method
    public Integer compareTo(Object compareTo) {
        Employee compareToEmp = (Employee)compareTo;
        if (id == compareToEmp.id) return 0;
        if (id > compareToEmp.id) return 1;
        return -1;
    }
}
```

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

```
@isTest
private class EmployeeSortingTest {
    @isTest
    static void test1() {
        List<Employee> empList = new List<Employee>();
        empList.add(new Employee(101, 'Joe Smith', '4155551212'));
        empList.add(new Employee(101, 'J. Smith', '4155551212'));
        empList.add(new Employee(25, 'Caragh Smith', '4155551000'));
        empList.add(new Employee(105, 'Mario Ruiz', '4155551099'));

        // Sort using the custom compareTo() method
        empList.sort();

        // Write list contents to the debug log
        System.debug(empList);

        // Verify list sort order.
    }
}
```



DID THIS ARTICLE SOLVE YOUR ISSUE?
Let us know so we can improve!

Share your feedback



DEVELOPER CENTERS

- Heroku
- MuleSoft
- Tableau
- Commerce Cloud
- Lightning Design System
- Einstein
- Quip

POPULAR RESOURCES

- Documentation
- Component Library
- APIs
- Trailhead
- Sample Apps
- Podcasts
- AppExchange

COMMUNITY

- Trailblazer Community
- Events and Calendar
- Partner Community
- Blog
- Salesforce Admins
- Salesforce Architects

© Copyright 2025 Salesforce, Inc. [All rights reserved](#). Various trademarks held by their respective owners. Salesforce, Inc.
Salesforce Tower, 415 Mission Street, 3rd Floor, San Francisco, CA 94105, United States

[Privacy Information](#) [Terms of Service](#) [Legal](#) [Use of Cookies](#) [Trust](#) [Cookie Preferences](#)
 [Your Privacy Choices](#) [Responsible Disclosure](#) [Contact](#)