Set Class

Represents a collection of unique elements with no duplicate values.

Namespace

System

Usage

The Set methods work on a set, that is, an unordered collection of elements that was initialized using the `set` keyword. Set elements can be of any data type–primitive types, collections, sObjects, user-defined types, and built-in Apex types. Set methods are all instance methods, that is, they all operate on a particular instance of a Set. The following are the instance methods for sets.

> **Note**
>
> - Uniqueness of set elements of user-defined types is determined by the `equals` and `hashCode` methods, which you provide in your classes. Uniqueness of all other non-primitive types is determined by comparing the objects' fields.
> - If the set contains String elements, the elements are case-sensitive. Two set elements that differ only by case are considered distinct.

For more information on sets, see Sets.

- **Set Constructors**
- **Set Methods**

Set Constructors

The following are constructors for `Set`.

- **Set<T>()**
  Creates a new instance of the `Set` class. A set can hold elements of any data type T.
- **Set<T>(setToCopy)**
  Creates a new instance of the `Set` class by copying the elements of the specified set. T is the data type of the elements in both sets and can be any data type.
- **Set<T>(listToCopy)**
  Creates a new instance of the `Set` class by copying the list elements. T is the data type of the elements in the set and list and can be any data type.

Set<T>()

Creates a new instance of the `Set` class. A set can hold elements of any data type T.

Signature

```
public Set<T>()
```

Example

```
s1.add('item1');
s1.add('item2');
```

Set<T>(setToCopy)

Creates a new instance of the `Set` class by copying the elements of the specified set. T is the data type of the elements in both sets and can be any data type.

Signature

```
public Set<T>(Set<T> setToCopy)
```

Parameters

setToCopy

Type: Set<T>

The set to initialize this set with.

Example

```
Set<String> s1 = new Set<String>();
s1.add('item1');
s1.add('item2');
Set<String> s2 = new Set<String>(s1);
// The set elements in s2 are copied from s1
System.debug(s2);
```

Set<T>(listToCopy)

Creates a new instance of the `Set` class by copying the list elements. T is the data type of the elements in the set and list and can be any data type.

Signature

```
public Set<T>(List<T> listToCopy)
```

Parameters

listToCopy

Type: Integer

The list to copy the elements of into this set.

Example

```
List<Integer> ls = new List<Integer>();
ls.add(1);
ls.add(2);
// Create a set based on a list
Set<Integer> s1 = new Set<Integer>(ls);
// Elements are copied from the list to this set
System.debug(s1);// DEBUG|{1, 2}
```

Set Methods

The following are methods for `Set`. All are instance methods.

- addAll(fromSet)
  Adds all of the elements in the specified set to the set that calls the method if they are not already present.
- clear()
  Removes all of the elements from the set.
- clone()
  Makes a duplicate copy of the set.
- contains(setElement)
  Returns `true` if the set contains the specified element.
- containsAll(listToCompare)
  Returns `true` if the set contains all of the elements in the specified list. The list must be of the same type as the set that calls the method.
- containsAll(setToCompare)
  Returns `true` if the set contains all of the elements in the specified set. The specified set must be of the same type as the original set that calls the method.
- equals(set2)
  Compares this set with the specified set and returns `true` if both sets are equal; otherwise, returns `false`.
- hashCode()
  Returns the hashcode corresponding to this set and its contents.
- isEmpty()
  Returns `true` if the set has zero elements.
- remove(setElement)
  Removes the specified element from the set if it is present.
- removeAll(listOfElementsToRemove)
  Removes the elements in the specified list from the set if they are present.
- removeAll(setOfElementsToRemove)
  Removes the elements in the specified set from the original set if they are present.
- retainAll(listOfElementsToRetain)
  Retains only the elements in this set that are contained in the specified list.
- retainAll(setOfElementsToRetain)
  Retains only the elements in the original set that are contained in the specified set.
- size()
  Returns the number of elements in the set (its cardinality).
- toString()
  Returns the string representation of the set.

add(setElement)

Adds an element to the set if it is not already present.

Signature

```
public Boolean add(Object setElement)
```

Parameters

setElement

Type: Object

Return Value

Type: Boolean

Usage

```
Boolean result = myString.add('d');
System.assertEquals(true, result);
```

addAll(fromList)

Adds all of the elements in the specified list to the set if they are not already present.

Signature

```
public Boolean addAll(List<Object> fromList)
```

Parameters

fromList

Type: List

Return Value

Type: Boolean

Returns `true` if the original set changed as a result of the call.

Usage

This method results in the *union* of the list and the set. The list must be of the same type as the set that calls the method.

addAll(fromSet)

Adds all of the elements in the specified set to the set that calls the method if they are not already present.

Signature

```
public Boolean addAll(Set<Object> fromSet)
```

Parameters

fromSet

Type: Set<Object>

Return Value

Type: Boolean

This method returns `true` if the original set changed as a result of the call.

Usage

This method results in the *union* of the two sets. The specified set must be of the same type as the original set that calls the method.

Example

```
Set<String> myString = new Set<String>{'a', 'b'};
Set<String> sString = new Set<String>{'c'};

Boolean result1 = myString.addAll(sString);
System.assertEquals(true, result1);
```

```
public Void clear()
```

Return Value

Type: Void

clone()

Makes a duplicate copy of the set.

Signature

```
public Set<Object> clone()
```

Return Value

Type: Set (of same type)

contains(setElement)

Returns `true` if the set contains the specified element.

Signature

```
public Boolean contains(Object setElement)
```

Parameters

setElement

Type: Object

Return Value

Type: Boolean

Example

```
Set<String> myString = new Set<String>{'a', 'b'};
Boolean result = myString.contains('z');
System.assertEquals(false, result);
```

containsAll(listToCompare)

Returns `true` if the set contains all of the elements in the specified list. The list must be of the same type as the set that calls the method.

Signature

```
public Boolean containsAll(List<Object> listToCompare)
```

Parameters

listToCompare

Type: List<Object>

Return Value

Type: Boolean

Signature

```
public Boolean containsAll(Set<Object> setToCompare)
```

Parameters

setToCompare

Type: Set<Object>

Return Value

Type: Boolean

Example

```
Set<String> myString = new Set<String>{'a', 'b'};
Set<String> sString = new Set<String>{'c'};
Set<String> rString = new Set<String>{'a', 'b', 'c'};

Boolean result1, result2;
result1 = myString.addAll(sString);
system.assertEquals(true, result1);

result2 = myString.containsAll(rString);
System.assertEquals(true, result2);
```

equals(set2)

Compares this set with the specified set and returns `true` if both sets are equal; otherwise, returns `false`.

Signature

```
public Boolean equals(Set<Object> set2)
```

Parameters

set2

Type: Set<Object>

The *set2* argument is the set to compare this set with.

Return Value

Type: Boolean

Usage

Two sets are equal if their elements are equal, regardless of their order. The `==` operator is used to compare the elements of the sets.

The `==` operator is equivalent to calling the `equals` method, so you can call `set1.equals(set2);` instead of `set1 == set2;`.

hashCode()

Returns the hashcode corresponding to this set and its contents.

Signature

```
public Integer hashCode()
```

isEmpty()

Returns `true` if the set has zero elements.

Signature

```
public Boolean isEmpty()
```

Return Value

Type: Boolean

Example

```
Set<Integer> mySet = new Set<Integer>();
Boolean result = mySet.isEmpty();
System.assertEquals(true, result);
```

remove(setElement)

Removes the specified element from the set if it is present.

Signature

```
public Boolean remove(Object setElement)
```

Parameters

setElement

Type: Object

Return Value

Type: Boolean

Returns `true` if the original set changed as a result of the call.

removeAll(listOfElementsToRemove)

Removes the elements in the specified list from the set if they are present.

Signature

```
public Boolean removeAll(List<Object> listOfElementsToRemove)
```

Parameters

listOfElementsToRemove

Type: List<Object>

Return Value

Type: Boolean

Returns `true` if the original set changed as a result of the call.

Usage

This method results in the *relative complement* of the two sets. The list must be of the same type as the set that calls the method.

Example

```
System.assertEquals(true, result);
Integer result2 = mySet.size();
System.assertEquals(1, result2);
```

removeAll(setOfElementsToRemove)

Removes the elements in the specified set from the original set if they are present.

Signature

```
public Boolean removeAll(Set<Object> setOfElementsToRemove)
```

Parameters

setOfElementsToRemove

Type: Set<Object>

Return Value

Type: Boolean

This method returns `true` if the original set changed as a result of the call.

Usage

This method results in the *relative complement* of the two sets. The specified set must be of the same type as the original set that calls the method.

retainAll(listOfElementsToRetain)

Retains only the elements in this set that are contained in the specified list.

Signature

```
public Boolean retainAll(List<Object> listOfElementsToRetain)
```

Parameters

listOfElementsToRetain

Type: List<Object>

Return Value

Type: Boolean

This method returns `true` if the original set changed as a result of the call.

Usage

This method results in the *intersection* of the list and the set. The list must be of the same type as the set that calls the method.

Example

```
Set<integer> mySet = new Set<integer>{1, 2, 3};
List<integer> myList = new List<integer>{1, 3};
Boolean result = mySet.retainAll(myList);
System.assertEquals(true, result);
```

retainAll(setOfElementsToRetain)

Parameters

setOfElementsToRetain

Type: Set

Return Value

Type: Boolean

Returns `true` if the original set changed as a result of the call.

Usage

This method results in the *intersection* of the two sets. The specified set must be of the same type as the original set that calls the method.

size()

Returns the number of elements in the set (its cardinality).

Signature

```
public Integer size()
```

Return Value

Type: Integer

Example

```
Set<Integer> mySet = new Set<Integer>{1, 2, 3};
Set<Integer> retainSet = new Set<Integer>{1, 3};
Boolean result = mySet.retainAll(retainSet);

Assert.isTrue(result, 'Expected to have changed mySet');

Integer retainedSetSize = mySet.size();
Assert.areEqual(2, retainedSetSize);
```

toString()

Returns the string representation of the set.

Signature

```
public String toString()
```

Return Value

Type: String

Usage

When used in cyclic references, the output is truncated to prevent infinite recursion. When used with large collections, the output is truncated to avoid exceeding total heap size and maximum CPU time.

- Up to 10 items per collection are included in the output, followed by an ellipsis (...).
- If the same object is included multiple times in a collection, it's shown in the output only once; subsequent references are shown as `(already output)`.

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