C# | HashSet Class

geeksforgeeks.org/c-sharp-hashset-class

October 30, 2018

A **HashSet<T>** is an unordered collection of the **unique elements**. It comes under ***System.Collections.Generic*** namespace. It is used in a situation where we want to prevent duplicates from being inserted in the collection. As far as performance is concerned, it is better in comparison to the list.

Characteristics of HashSet Class:

- The HashSet<**T**> class provides high-performance set operations. A set is a collection that contains no duplicate elements, and whose elements are in no particular order.
- The capacity of a HashSet<**T**> object is the number of elements that the object can hold.
- A HashSet<**T**> object's capacity automatically increases as elements are added to the object.
- A HashSet<**T**> collection is not sorted and cannot contain duplicate elements.
- HashSet<**T**> provides many mathematical set operations, such as set addition (unions) and set subtraction.

Constructors

Constructor	Description
HashSet()	Initializes a new instance of the HashSet class that is empty and uses the default equality comparer for the set type.
HashSet(IEnumerable)	Initializes a new instance of the HashSet class that uses the default equality comparer for the set type, contains elements copied from the specified collection, and has sufficient capacity to accommodate the number of elements copied.
HashSet(IEnumerable, IEqualityComparer)	Initializes a new instance of the HashSet class that uses the specified equality comparer for the set type, contains elements copied from the specified collection, and has sufficient capacity to accommodate the number of elements copied.

HashSet(IEqualityComparer)	Initializes a new instance of the HashSet class that is empty and uses the specified equality comparer for the set type.
HashSet(Int32)	Initializes a new instance of the HashSet class that is empty, but has reserved space for capacity items and uses the default equality comparer for the set type.
HashSet(Int32, IEqualityComparer)	Initializes a new instance of the HashSet class that uses the specified equality comparer for the set type, and has sufficient capacity to accommodate capacity elements.
HashSet(SerializationInfo, StreamingContext)	Initializes a new instance of the HashSet class with serialized data.

Example:

filter_none
edit

play_arrow

brightness_4

```
using System;
using System.Collections.Generic;
class GFG {
public static void Main()
{
HashSet< int > odd = new HashSet< int >();
for ( int i = 0; i < 5; i++) {
odd.Add(2 * i + 1);
}
foreach ( int i in odd)
{
Console.WriteLine(i);
}
}
}
```

Output:

```
1
3
5
7
9
```

Properties

Property	Description
Comparer	Gets the IEqualityComparer object that is used to determine equality for the values in the set.
Count	Gets the number of elements that are contained in a set.

Example:

filter_none
edit

play_arrow

brightness_4

```
using System;
using System.Collections.Generic;
class GFG {
public static void Main()
{
HashSet< int > mySet = new HashSet< int >();
for ( int i = 0; i < 5; i++) {
mySet.Add(i * 2);
}
Console.WriteLine(mySet.Count);
}
}
```

Output:

```
5
```

Methods

Method	Description
Add(T)	Adds the specified element to a set.

Clear()	Removes all elements from a HashSet object.
Contains(T)	Determines whether a HashSet object contains the specified element.
CopyTo()	Copies the elements of a HashSet collection to an array.
CreateSetComparer()	Returns an IEqualityComparer object that can be used for equality testing of a HashSet object.
Equals(Object)	Determines whether the specified object is equal to the current object.
ExceptWith(IEnumerable)	Removes all elements in the specified collection from the current HashSet object.
GetEnumerator()	Returns an enumerator that iterates through a HashSet object.
GetHashCode()	Serves as the default hash function.
GetObjectData(SerializationInfo, StreamingContext)	Implements the ISerializable interface and returns the data needed to serialize a HashSet object.
GetType()	Gets the Type of the current instance.
IntersectWith(IEnumerable)	Modifies the current HashSet object to contain only elements that are present in that object and in the specified collection.
IsProperSubsetOf(IEnumerable)	Determines whether a HashSet object is a proper subset of the specified collection.
IsProperSupersetOf(IEnumerable)	Determines whether a HashSet object is a proper superset of the specified collection.
IsSubsetOf(IEnumerable)	Determines whether a HashSet object is a subset of the specified collection.

IsSupersetOf(IEnumerable)	Determines whether a HashSet object is a superset of the specified collection.
MemberwiseClone()	Creates a shallow copy of the current Object.
OnDeserialization(Object)	Implements the ISerializable interface and raises the deserialization event when the deserialization is complete.
Overlaps(IEnumerable)	Determines whether the current HashSet object and a specified collection share common elements.
Remove(T)	Removes the specified element from a HashSet object.
RemoveWhere(Predicate)	Removes all elements that match the conditions defined by the specified predicate from a HashSet collection.
SetEquals(IEnumerable)	Determines whether a HashSet object and the specified collection contain the same elements.
SymmetricExceptWith(IEnumerable)	Modifies the current HashSet object to contain only elements that are present either in that object or in the specified collection, but not both.
ToString()	Returns a string that represents the current object.
TrimExcess()	Sets the capacity of a HashSet object to the actual number of elements it contains, rounded up to a nearby, implementation-specific value.
TryGetValue(T, T)	Searches the set for a given value and returns the equal value it finds, if any.
UnionWith(IEnumerable)	Modifies the current HashSet object to contain all elements that are present in itself, the specified collection, or both.

Example:

filter_none
edit

play_arrow

brightness_4

```
using System;
using System.Collections.Generic;
class GFG {
public static void Main()
{
HashSet< int > mySet1 = new HashSet< int >();
for ( int i = 1; i <= 5; i++)
mySet1.Add(2 * i);
HashSet< int > mySet2 = new HashSet< int >();
for ( int i = 1; i <= 10; i++)
mySet2.Add(i);
Console.WriteLine(mySet1.IsSubsetOf(mySet2));
}
}
```

Output:

```
True
```

Example:

filter_none
edit

play_arrow

brightness_4

```
using  System;
using  System.Collections.Generic;
class  GFG {
public  static  void  Main()
{
HashSet<  string  > mySet =  new  HashSet<  string  >();
mvSet.Add(  "DS"  );
mvSet.Add(  "C++"  ):
mvSet.Add(  "Iava"  ):
mySet.Add(  "JavaScript"  );
if  (mySet.Contains(  "Java"  ))
Console.WriteLine(  "Required Element is present"  );
else
Console.WriteLine(  "Required Element is not present"  );
}
}
```

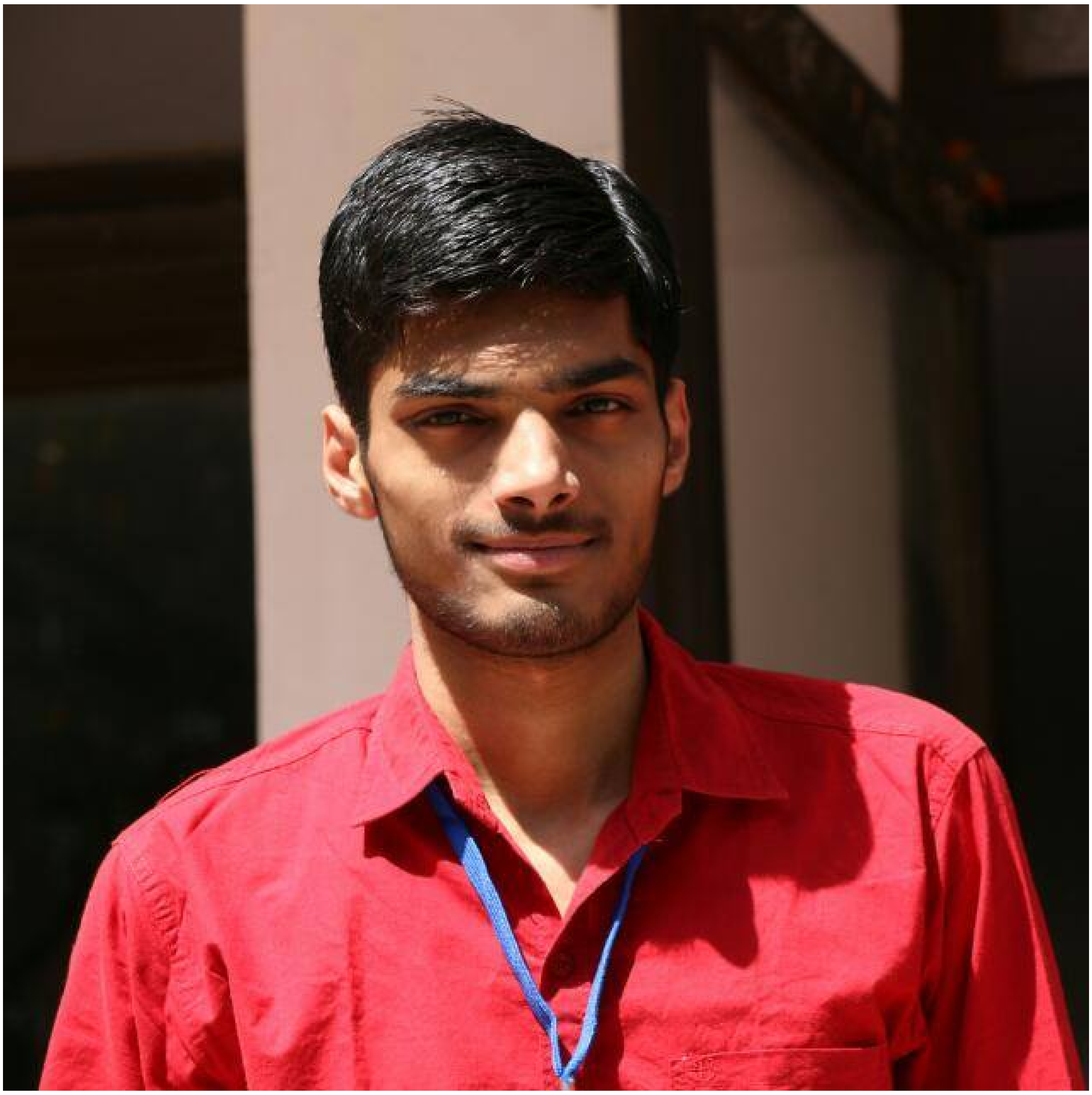
Output:

```
Required Element is present
```

Reference:

> https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.hashset-1?view=netframework-4.7.2

Sahil_Bansall
In love with a semicolon because sometimes i miss it so badly)



If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.