UNDERSTANDING COLLECTIONS AND COLLECTIONS INTERFACES

A collection is a set of related objects. Unlike arrays, a collection can grow and shrink dynamically as the number of objects added or deleted. A collection is a class, so you must declare a new collection before you can add elements to that collection.

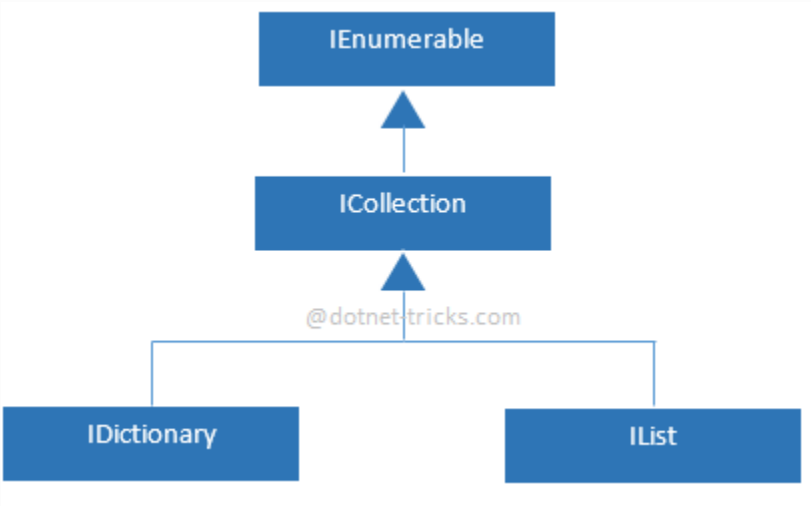
The .NET Framework provides various collections like ArrayList, HashTable , SortedList, Stack and Queue etc. All these collections are exists in System.Collections namespace.

Class	Description
ArrayList	Represents an array of objects whose size is dynamically increased or decreased as required. It is an alternative to an array. It supports add and remove methods for adding and removing objects from the collection.
Hashtable	Represents a collection of objects which are stored in key/value pair's fashion, where key is a hash code and value is an object. The key is used to access or manipulates the objects in the collection. It supports add and remove methods for adding and removing objects from the collection.
SortedList	Represents a collection of objects which are stored in key/value pairs fashion like HashTable and can be sorted by the keys. It can accessible by key or by index number. Typically, it a combination of ArrayList and HashTable. It supports add and remove methods for adding and removing objects from the collection.
Stack	Represents a last in, first out (LIFO) collection of objects. It supports push and pop methods for adding and removing objects from the collection.
Queue	

Represents a first in, first out (FIFO) collection of objects. It supports Enqueue and Dequeue methods for adding and removing objects from the collection.

Collection Interfaces

All of the collection types use some common interfaces. These common interfaces define the basic functionality for each collection class. The key collections interfaces are – IEnumerable, ICollection, IDictionary and IList.



IEnumerable acts as a base interface for all the collection types that is extended by ICollection. ICollection is further extended by IDictionary and IList.

Interface	Description
IEnumerable	Provides an enumerator which supports a simple iteration over a non-generic collection.
ICollection	Defines size, enumerators and synchronization methods for all nongeneric collections.
IDictionary	Represents a nongeneric collection of key/value pairs.
IList	Represents a non-generic collection of objects that can be individually accessed by index.

All collections interfaces are not implemented by all the collections. It depends on collection nature. For example, IDictionary interface would be implemented by only those collection classes which support key/value pairs, like HasTable and SortedList etc.

Source : http://www.dotnet-tricks.com/Tutorial/csharp/bDRH230314-Understanding-Collections-and-Collections-Interfaces.html