C# Collections (Data Structures)

INSTRUCTOR: ADEEN BIN WAHEED

LECTURE 13

Collections Important Functions

Count:

It is used to get the total number of elements in any collection.

AddRange:

It is used to combine two same collections.

Reverse:

It is used to reverse our collections.

Clear:

It is used to clear our collections.

Sort:

It is used to sort collection elements.

Non-Generic vs Generic

Non-Generic	Similar Generic Type
ArrayList	List<T>
Hashtable	Dictionary<TKey,TValue>
SortedList	SortedList<TKey,TValue>
Queue	Queue<T>
Stack	Stack<T>
IEnumerable	IEnumerable<T>
ICollection	N/A (use IEnumerable<T> anything that extends it)
N/A	ICollection<T>

ArrayList

This collection dynamically resizes. It grows in capacity as elements are added (if space is needed). It is most often used in older C# programs.

It appends a new element object to the end of the ArrayList. We can keep adding elements to the collection until memory runs out. The objects are stored in the managed heap.

ArrayList

Running Demo

List vs ArrayList

List<T> is a generic class.

ArrayList is a non-generic class.

Note: ArrayList is an old type not try to use it as compared List Generic Type.

Hashtable

This optimizes lookups. It computes a hash of each key you add. It then uses this hash code to look up the element very quickly.

After creating a hashtable, we directly assign values with the indexer, which uses the square brackets.

Hashtable

Running Demo

Dictionary vs Hashtable

Dictionary is a generic class.

Hastable is a non generic class.

Note: Don't use Hastable**.** It is an older .NET Framework type. It is slower than the generic Dictionary type. But if an old program uses Hashtable, it is helpful to know how to use this type.Dictionary.

Generic Collections vs Non-Generic Collections

For generic classes, use following header.

using System.Collections;

And, for using non-generic classes, use following head.

using System.Collections.Generic;

Generic Collections vs Non-Generic Collections

Running Demos