Capacity

- It is used to pre-allocate the internal array used by the list. (The size of this internal array is given by List.Capacity.
- List<T>.Capacity Property is used to gets or sets the total number of elements the internal data structure can hold without resizing.

Properties of List:

- It is different from the arrays. A list can be resized dynamically but arrays cannot.
- List class can accept null as a valid value for reference types and it also allows duplicate elements.
- If the Count becomes equals to Capacity then the capacity of the List increases automatically by reallocating the internal array. The existing elements will be copied to the new array before the addition of the new element.

Capacity Vs Count:

- Count is always less than the Capacity. While adding the elements, if Count exceeds Capacity then the Capacity will increase automatically by reallocating the internal array before copying the old elements and adding the new elements.
- Capacity is the number of the elements which the List can store before resizing of List needed. But Count is the number of the elements which are actually present in the List.
- If the Capacity is much larger than the Count the user can decrease capacity by either calling the TrimExcess method or explicitly setting the Capacity to a lower value.
- If the Capacity is settled explicitly then the internal array is also reallocated to accommodate the specified capacity, and all the elements are copied.
- Retrieving the value of Capacity property is an O(1) operation while setting the Capacity is an O(n) operation, where n is the new capacity.