Accessing List

Use a foreach or for loop to iterate a List<T> collection.

Example Accessing List

```
List<int> intList = new List<int>() { 10, 20, 30 };
intList.ForEach(el => Console.WriteLine(el));
```

If you have initialized the List<T> with an IList<T> interface then use seperate foreach statement with implicitly typed variable:

Example: Accessing List

```
IList<int> intList = new List<int>() { 10, 20, 30, 40 }; foreach
(var el in intList)
Console.WriteLine(el);
```

Access individual items by using an indexer (i.e., passing an index in square brackets):

Example: Accessing List

```
IList<int> intList = new List<int>() { 10, 20, 30,
40 }; int elem = intList[1]; // returns 20
```

Use the Count property to get the total number of elements in the List.

Example: Access List elements

```
IList<int> intList = new List<int>() { 10, 20, 30, 40 };
Console.Write("Total elements: {0}", intList.Count);
```

```
Total elements: 4
```

Insert Elements into List

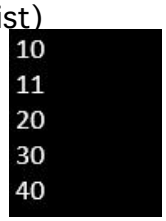
Use the IList.Insert() method inserts an element into a List<T> collection at the specified index.

Insert() signature:void Insert(int index, T item);

Example: Insert elements into List

```
IList<int> intList = new List<int>(){ 10, 20, 30, 40 };
intList.Insert(1, 11);// inserts 11 at 1st index: after
```

10. foreach (var el in intList)
Console.Write(el);



Remove Elements from List

The Remove() and RemoveAt() methods to remove items from a List<T> collection.

Remove() signature: bool Remove(T item) RemoveAt()
signature: void RemoveAt(int index)

Example: Remove elements from List

IList<int> intList = new List<int>(){ 10, 20, 30, 40 }; intList.Remove(10); //
removes the 10 from a list intList.RemoveAt(2); //removes the 3rd element
(index starts from 0) foreach (var el in intList)
Console.Write(el);

Points to Remember:

1. List<T> stores elements of the specified type and it grows automatically.
2. List<T> can store multiple null and duplicate elements.
3. List<T> can be assigned to IList<T> or List<T> type of variable. It provides more helper method When assigned to List<T> variable
4. List<T> can be access using indexer, for loop or foreach statement.
5. LINQ can be use to query List<T> collection.
6. List<T> is ideal for storing and retrieving large number of elements.