The LINQ ToList and ToArray methods in C# are used to convert collections or sequences to a List<T> or an array T[], respectively. Both methods are part of the System.Linq namespace can be applied to any type implementing IEnumerable<T>. These methods are commonly used for materializing query results into concrete collections.

1. LINQ ToList Method in C#:

The LINQ ToList Method in C# creates a List<T> from an IEnumerable<T>. This is particularly useful when you need a list with the functionality it provides, like the ability to add or remove items. This method create a System.Collections.Generic.List<T> collection from a System.Collections.Generic.IEnumerable<T>. This method causes the query to be executed immediately. The signature of the ToList method is shown below.

* Parameters:

source: The data type of source is System.Collections.Generic.IEnumerable<T>.

* Type parameters:

TSource: The type of elements contained in the source sequence.

* Returns:

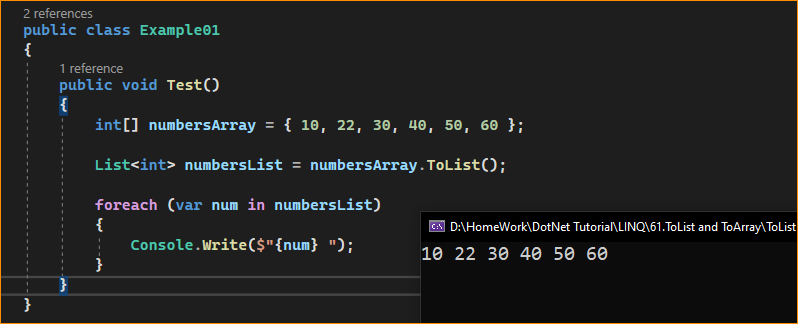
It returns System.Collections.Generic.List<T>, which contains elements from the source sequence.

* Exceptions:

This method throws System.ArgumentNullException when the source sequence is null.

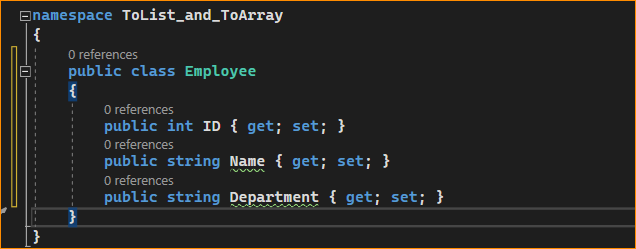
1. Example to Convert int array to List<int> using the LINQ ToList Method in C#

In the following example, we first create an integer array and then convert that integer array into a list (i.e., List<int>) by using the LINQ ToList Method in C#.

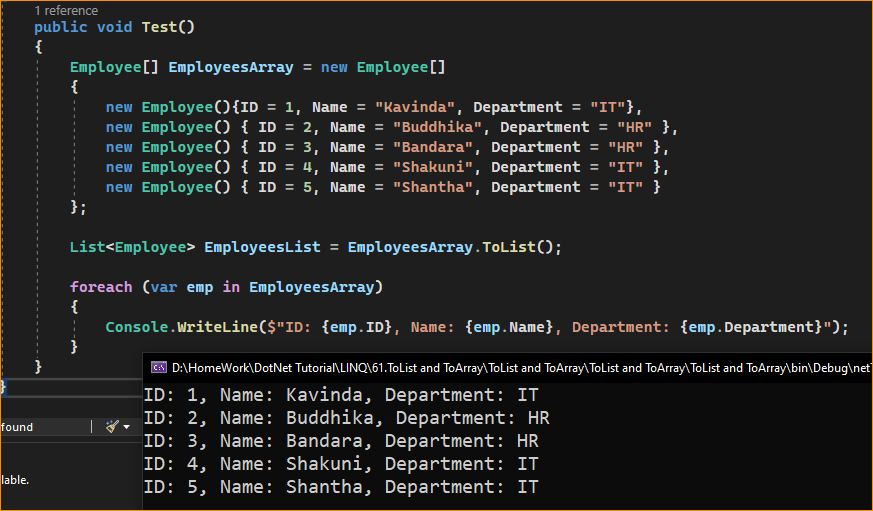


1. Working with Complex Type:

Let us see how we can work with the ToArray method of a collection of Complex Types. For this, we are going to use the following Employee class. So, create a class file named Employee.cs and copy and paste the following code. It’s a very simple class, having only three properties.

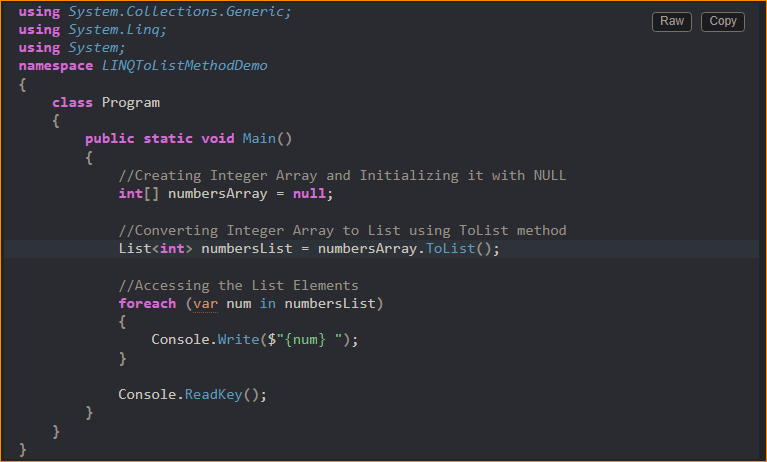


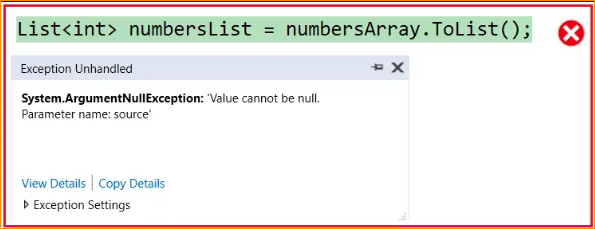
Next, modify the Main method of the Program class as follows. As you can see, first, we are creating an array of Employees and storing 5 employee information. Then, we convert that array of Employees to a list of Employees by calling the ToList Method.



1. What happens when the Source Array is Null?

When the Source Array is Null, and if we try to convert that null array into a list using the ToList method, we will get one runtime exception saying the value cannot be null. For a better understanding, please have a look at the following example. In the example below, we create one null array and then convert that integer null array to a list.





1. LINQ ToArray Method in C#:

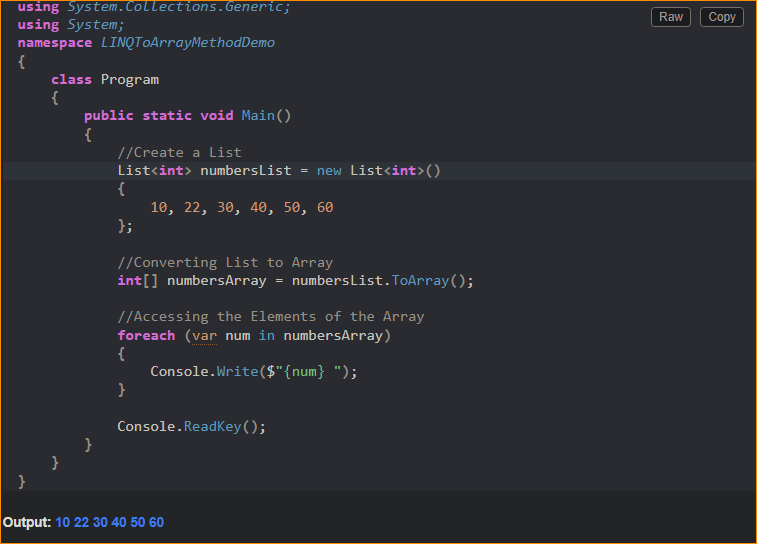
The LINQ ToArray Method converts an IEnumerable<T> to an array T[]. This is useful when you need a fixed-size collection or interfacing with APIs requiring arrays. This method copy the elements of the System.Collections.Generic.List<T> to a new array. This method causes the query to be executed immediately. The signature of this method is shown below.



Here, T is the array type, and this method converts a list into an array and returns that array containing copies of the elements of the System.Collections.Generic.List<T>.

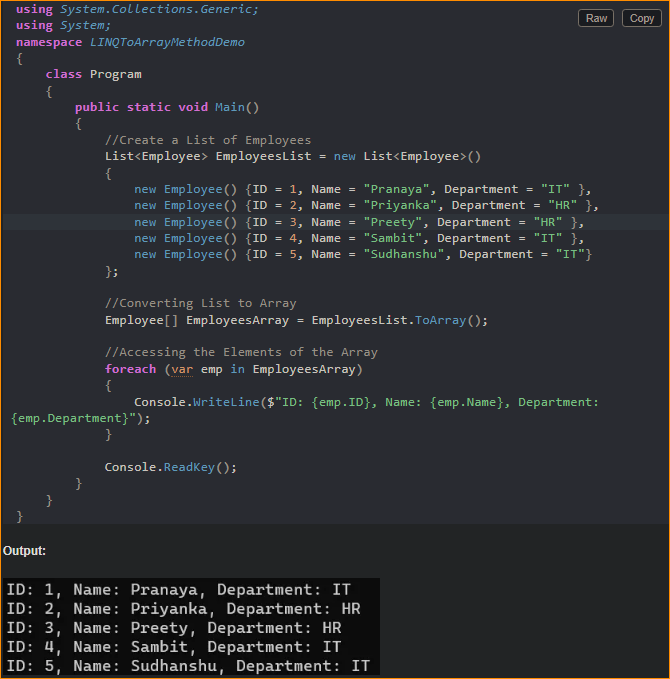
Example: Convert List<int> to Integer Array in C# using LINQ ToArray Method.

In the following example, first, we create a list of integers and then convert that list of integers into an integer array (i.e., int[]) by using the LINQ ToArray Method in C#. The following code is self-explained, so please go through the comment lines for a better understanding.



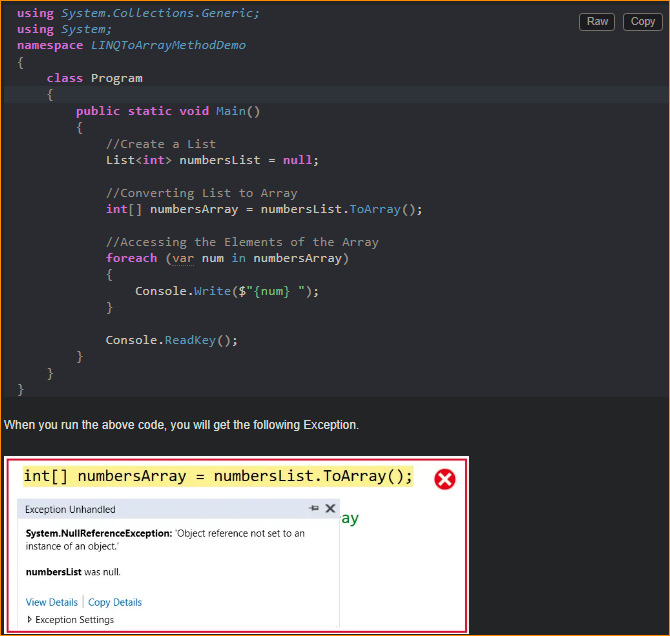
1. Working with Complex Type:

We are going to work with the same Employee class. So, modify the Main method of the Program class as follows. As you can see, first, we are creating a list collection of type Employees and storing 5 employee information. Then, we convert that list of Employees to an array of Employees by calling the ToArray Method.



1. What Happens When We Call the ToArray Method on a Null List?

We will get a Null Reference Exception if we call the ToArray Method on a Null List. For a better understanding, please have a look at the following example. In the below example, we are creating one list, initializing that list with NULL, and then calling the ToArrya method on that list instance.



Key Points

* Materialization: ToList and ToArray are used for materializing the results of a LINQ query. This means they execute the query and create a concrete collection with the results.
* Immutability of Results: The resulting List<T> or T[] is a snapshot of the data during the call. Subsequent changes to the source collection do not affect the list’s contents or array.
* Memory Allocation: Both methods allocate memory for the new collection. This can be a consideration for very large datasets.
* Choice of Method: Use ToList when you need the dynamic features of a list (like adding or removing items), and use ToArray when you need a fixed-size, immutable collection or are interfacing with APIs that require arrays.

Both ToList and ToArray are commonly used in scenarios where you need to work with the results of a LINQ query as a standard collection, like in data-binding scenarios or when passing data to methods that require specific collection types.