1. LINQ Cast Method in C#:

The LINQ Cast method in C# is used to cast elements of a non-generic collection to a specified type. It’s a part of the System.Linq namespace can be applied to any type that implements IEnumerable, converting it into an IEnumerable<T> where T is the target type. The Cast method is particularly useful when working with not strongly typed collections, allowing you to apply LINQ queries that require a specific type.

The LINQ Cast Method in C# is used to cast all the elements of a collection (System.Collections.IEnumerable) to a specified type and then return a new System.Collections.Generic.IEnumerable<T> collection containing all the source sequence elements cast to the specified type. This method uses deferred execution. The signature of this method is given below.



Parameters:

* source: The System.Collections.IEnumerable that contains the elements to be cast to type TResult.

Type Parameters:

* TResult: The type to cast the elements of the source.

Exceptions:

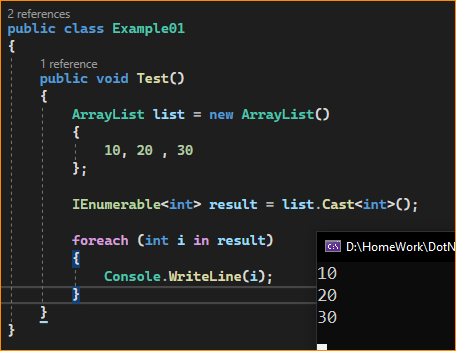
* It will throw the System.ArgumentNullException when the source is null.
* It will throw the System.InvalidCastException when an element in the source sequence cannot be cast to the specified type TResult.

Returns:

* It returns a System.Collections.Generic.IEnumerable contains each element of the source sequence cast to the specified type.

1. Example to Understand LINQ Cast Method in C#:

In the following example, the source sequence contains 3 elements, all of which are type integers. So, all these elements are cast into integers. So, while casting using the Cast method, we need to specify the type we want to cast.



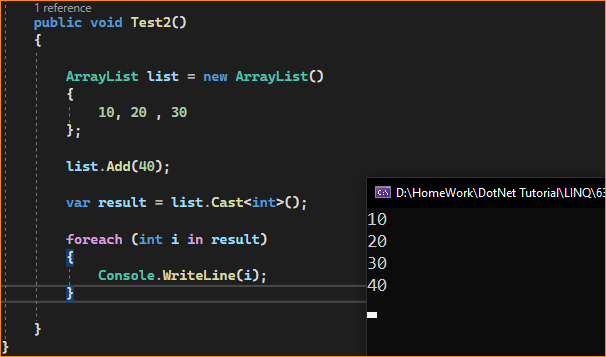
1. LINQ Cast Method Key Points

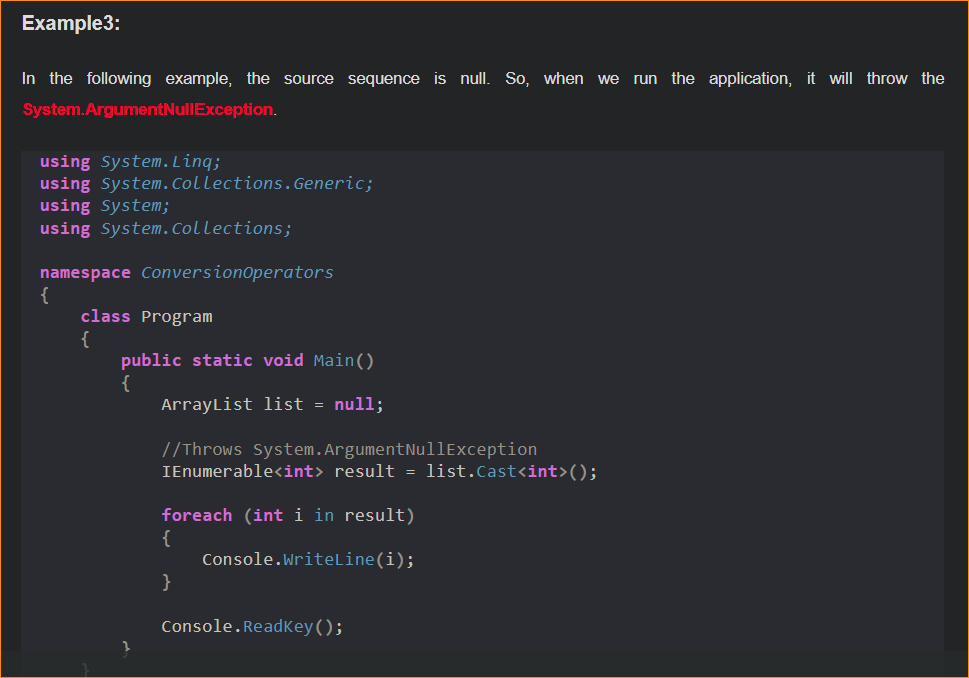
* Type Casting: Cast attempts to cast every element in the collection to the specified type. If any element cannot be cast to the specified type, an InvalidCastException is thrown at runtime.
* Use with Non-Generic Collections: Cast is mainly used with non-generic collections like ArrayList or non-generic IEnumerable. It’s unnecessary for generic collections like List<T>, as they are already strongly typed.
* Deferred Execution: The casting operation is performed when you iterate over the resulting IEnumerable<T>, not when the Cast method is called.
* Combining with LINQ Queries: After casting, you can apply standard LINQ query operators (like Where, Select, OrderBy) on the collection.

.

1. Example2:

In the following example, we have created a non-generic (ArrayList) collection that contains integer and string values. Then, we cast that collection to IEnumerable<int> using the Cast<int> method. At runtime, while casting the values,





1. When to Use the LINQ Cast Method in C#?

The LINQ Cast method in C# is particularly useful in scenarios where you have a non-generic collection and need to perform operations that require the elements to be of a specific type. Here are some typical use cases:

* Working with Non-Generic Collections: When dealing with older non-generic collections like ArrayList or Hashtable, where elements are stored as objects, Cast can be used to convert these elements into a specific type for further processing.
* Applying LINQ Queries to Non-Generic Collections: To use LINQ query operators (like Where, Select, OrderBy) on non-generic collections, you must first cast the elements to a specific type. Cast enables this transition.
* Data Transformation: In scenarios where data retrieved from sources like databases, file systems, or external services comes in a non-generic format, Cast can be used to transform this data into a strongly typed format for ease of manipulation.
* Interoperating with Older .NET Code: If you’re working with code predating generics in .NET and need to integrate it with newer, strongly typed code, Cast can serve as a bridge by converting collections to the required generic types.
* Type-Safe Enumeration: When you need to enumerate through a non-generic collection while performing type-specific operations on each element, Cast ensures that the elements are of the expected type, thus enabling type-safe operations.
* Combining Collections of Different Types: If you have multiple non-generic collections that you need to combine or process uniformly, Cast can be used to ensure all collections are of the same type before combining or querying them.

It’s important to note that Cast will throw an InvalidCastException at runtime if any element in the collection cannot be cast to the specified type. Therefore, its use assumes that you know the types of elements in the collection and are confident that all elements can be safely cast to the target type.