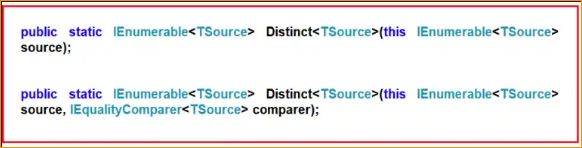
1. What is LINQ Distinct(different) Method in C#?

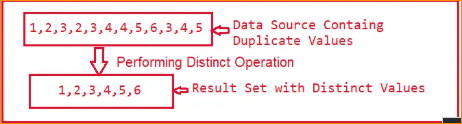
The LINQ Distinct Method in C# is used to return the distinct elements from a single data source. There are two overloaded versions available for the Distinct Method as shown below.



The one and only difference between these two methods is the second overloaded version take an IEqualityComparer as an input parameter which means the Distinct Method can also be used with Comparer also.

Example to Understand LINQ Distinct Method on Value Type using C#

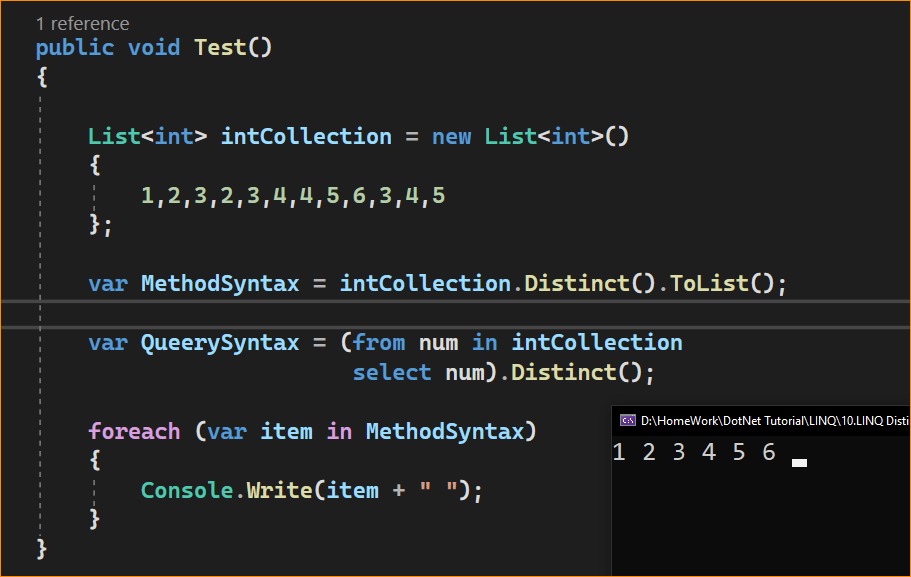
Here we have an integer collection that contains duplicate integer values. Our requirement is to remove the duplicate values and return only the distinct values as shown below.



* The following example shows how to get the distinct integer values from the data source using both Method and Mixed syntax using LINQ Distinct Extension Method. In Query Syntax, there is no such operator call distinct, so we need to use both Query and Method syntax to achieve the same.

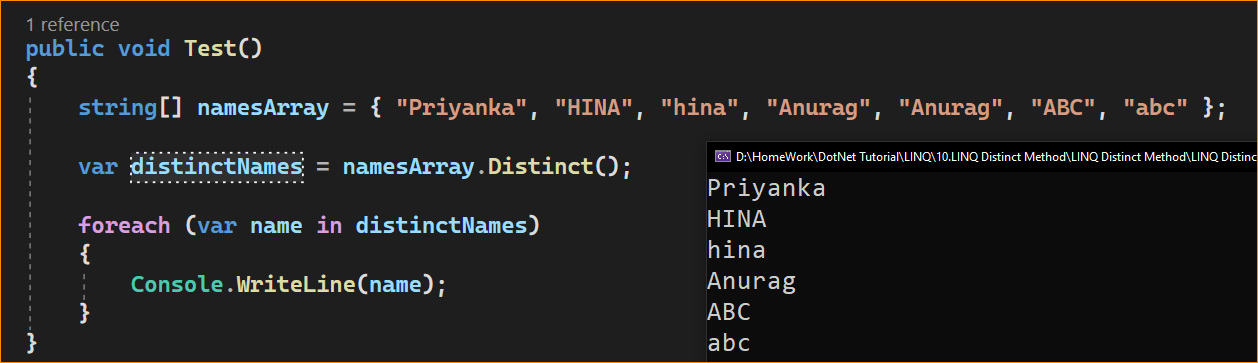
1. Example to Understand LINQ Distinct Method with String Values:

Let us see how we can use the LINQ Distinct Method with string values. In the below example, we have a string array of names and we need to return the distinct names from that array collection. To do so, we are using the LINQ Distinct Method.



1. Example to Understand LINQ Distinct Method with String Values:

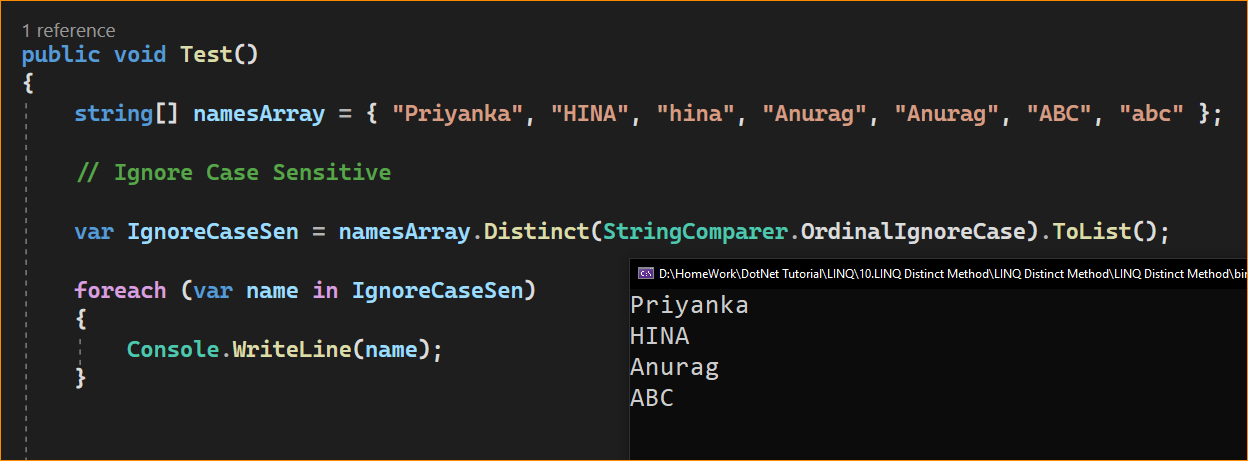
Let us see how we can use the LINQ Distinct Method with string values. In the below example, we have a string array of names and we need to return the distinct names from that array collection. To do so, we are using the LINQ Distinct Method.

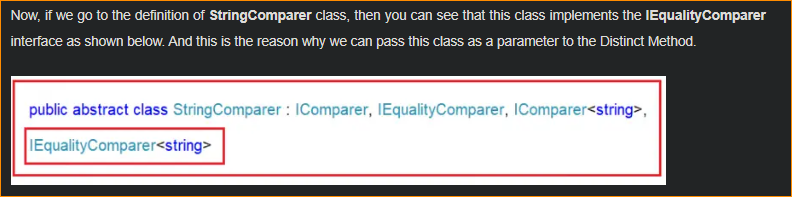


Here C# is Case Sensitive- Name meaning same but Simple and capital. So deferent words

As you can see the name Hina and Abc have appeared twice. This is because the default comparer, which is used by the LINQ Distinct method to filter the duplicate values is case-sensitive. So, if you want to make the comparison to be case-insensitive then you need to use the other overloaded version of the Distinct Method which takes IEqualityComparer as an argument. So here we need to pass a class that must implement the IEqualityComparer interface.

So let’s modify the Program class as follows. **Here, you can see, we are passing StringComparer as an argument to the LINQ Distinct method and saying OrdinalIgnoreCase which means please ignore the case sensitive while checking the duplicity.**



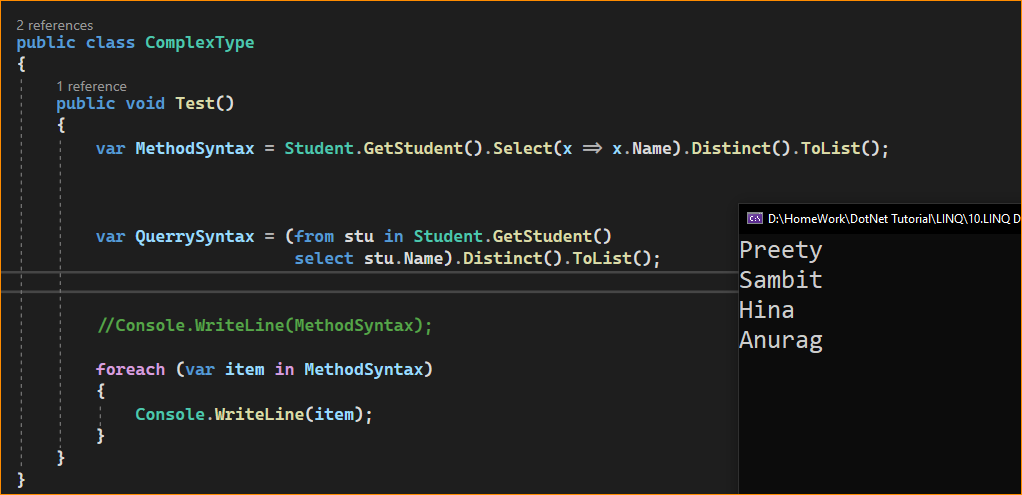


LINQ Distinct Operation with Complex Data Type using C#:

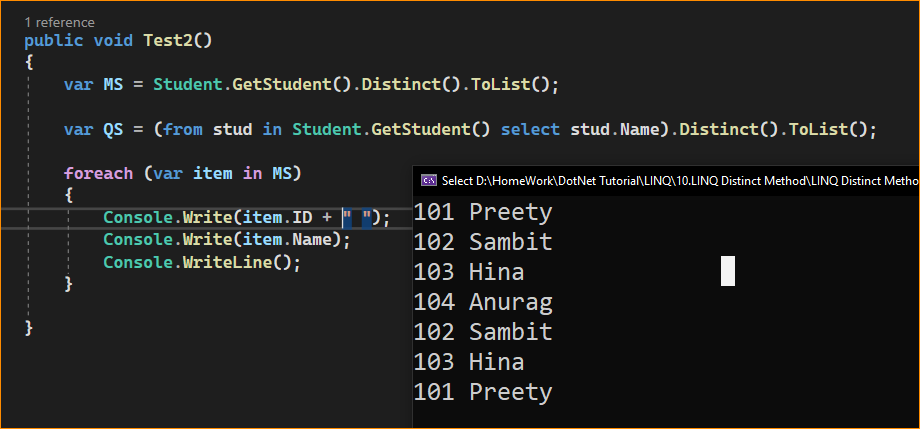
The LINQ Distinct Method in C# will work in a different manner with complex data types like Employee, Product, Student, etc. Let us understand this with an example. Create a class file with the name Student.cs and then copy and paste the following code into it.



Here we created the student class with the two properties i.e. ID and Name. Along the same way, we have also created the GetStudents() method which will return a hard-coded collection of students. So, basically, it is returning the following Student data.



In our previous example, we try to retrieve the distinct student names and it works as expected. Now, our requirement is to select distinct students (both ID and Name) from the collection. As you can see in our collection three students are identical and in our result set, they should appear only once. Let us modify the program class as shown below to fetch the distinct student using the LINQ Distinct Method.



As you can see, it will not select distinct students rather it select all the students. This is because the default comparer which is used for comparison by LINQ Distinct Method is only checked whether two object references are equal or not and not the individual property values of the complex object.

* How to Solve the Above Problem?

We can solve the above problem in four different ways. They are as follows

1. We need to use the other overloaded version of the Distinct() method which takes the IEqualityComparer interface as an argument. So, here we need to create a class that implements the IEqualityComparer interface and then we need to pass that compare instance to the Distinct() method.
2. In the second approach, we need to override the Equals() and GetHashCode() methods within the Student class itself.
3. In the third approach, we need to project the required properties into a new anonymous type, which already overrides the Equals() and GetHashCode() methods
4. By Implementing IEquatable<T> interface.

**Approach 3: Using Anonymous Type (This is Easy Way)**

In this approach, we need to project the properties of the Student class into a new anonymous type and it will work as expected. The reason is the Annonymous Type already overrides the Equals() and GetHashCode() methods of the Object Class. So, modify the Main Method of the Program class as follows. Here, you can see, using the Select Projection Operator and Select Method, we are projecting the output to an anonymous type.

