```
        /// <summary>
        /// This method implements the UnionAll extension method.
        /// This extension method results in all of the rows in <paramref name="Source1"/>
        /// having all the rows in <paramref name="Source2"/> appended.<br/>
        /// This method makes use of the Linq extension method
        /// <seealso cref=
        /// "System.Linq.Enumerable.Union&lt;TSource&gt;(IEnumerable&lt;TSource&gt; , IEnumerable&lt;TSource&gt; , IEqualityComparer&lt;TSou
rce&gt; )"/>
        ///  with and implementation of the <seealso cref="System.Collections.Generic.IEqualityComparer&lt; T&gt;"/>
        /// </summary>
        /// <typeparam name="TSource">
        /// The type of objects in both <paramref name="Source1"/> and <paramref name="Source2"/>.
        /// </typeparam>
        /// <param name="Source1">
        /// The input sequence which is used as the first set of rows in the output.
        /// These rows are of <typeparamref name="TSource"/> type.
        /// </param>
        /// <param name="Source2">
        /// The input sequence which is used as the second set of rows in the output.
        /// These rows are of <typeparamref name="TSource"/> type.
        /// </param>
        /// <returns>An output IEnumerable of type <typeparamref name="TSource"/>that contains all the rows from
        /// <paramref name="Source1"/> and <paramref name="Source2"/>.
        /// </returns>
        /// <remarks>
        /// This implementation used the Linq extension method <seealso cref=
        /// "System.Linq.Enumerable.Union&lt;TSource&gt;(IEnumerable&lt;TSource&gt; , IEnumerable&lt;TSource&gt; , IEqualityComparer&lt;TSou
rce&gt; )"/>
        ///  with and implementation of the <seealso cref="System.Collections.Generic.IEqualityComparer&lt; T&gt;"/>
        ///  that makes all objects not equal. This forces the distinct phase of the union process to maintain all of the rows in the inputs
.
        /// </remarks>
        public static IEnumerable<TSource> UnionAll<TSource>(
            this IEnumerable<TSource> Source1,
            IEnumerable<TSource> Source2)
        {
            EverythingDifferentEqualityComaperer<TSource> AllDifferent = new EverythingDifferentEqualityComaperer<TSource>( );
            return Source1.Union(Source2, AllDifferent);
        }
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LINQ_Extension_Methods
{
    /// <summary>
    /// A class which implements the IEqualityComparer Interface.
    /// The implementation makes all elements not the same.
    /// Currently, this is class used in the UnionAll extension method to achieve the all rows preserved.
    /// </summary>
    /// <typeparam name="TSource">The type of objects being compared.</typeparam>
    internal class EverythingDifferentEqualityComaperer<TSource> : IEqualityComparer<TSource>
    {
        /// <summary>
        /// Public constructor for the class.
        /// </summary>
        public EverythingDifferentEqualityComaperer( )
        {

        }

        #region IEqualityComparer<TSource> Members
        /// <summary>
        /// This method says 'everything is different'
        /// </summary>
        /// <param name="x">One of the objects of <typeparamref name="TSource"/> to be compared. The implementation ignores the object.</par
am>
        /// <param name="y">One of the objects of <typeparamref name="TSource"/> to be compared. The implementation ignores the object.</par
am>
        /// <returns>false for all values.</returns>
        bool IEqualityComparer<TSource>.Equals(TSource x, TSource y)
        {
            return false;
        }
        /// <summary>
        /// Returns the hash code of the object.
        /// </summary>
        /// <param name="obj">The object of <typeparamref name="TSource"/></param>
        /// <returns>Hash code of the object.</returns>
        int IEqualityComparer<TSource>.GetHashCode(TSource obj)
        {
            if (Object.ReferenceEquals(obj, null))
```

```
                return 0;
            return obj.GetHashCode( );
        }

        #endregion
    }
}
```