In C# and .NET, the Lookup class is part of the System.Linq namespace and is used to create a collection of keys and associated values. It is similar to a dictionary but **allows multiple values for a single key.**

```
using System;
using System.Linq;

class Program
{
    static void Main()
    {
        // Create a sample collection of data
        var data = new[]
        {
            new { Key = "A", Value = 1 },
            new { Key = "B", Value = 2 },
            new { Key = "A", Value = 3 },
            new { Key = "C", Value = 4 },
            new { Key = "B", Value = 5 }
        };

        // Create a Lookup from the data using a lambda expression to
define the key
        var lookup = data.ToLookup(item => item.Key, item => item.Value);

        // Access values by key
        foreach (var key in lookup)
        {
            Console.WriteLine($"Key: {key.Key}");

            foreach (var value in key)
            {
                Console.WriteLine($"  Value: {value}");
            }
        }
    }
}
```

Output:

```
Key: A
  Value: 1
  Value: 3
Key: B
  Value: 2
  Value: 5
Key: C
  Value: 4
```

Note that Lookup is read-only, meaning you cannot modify its contents directly.

if u need mutable version than apply same logic in Dictionary

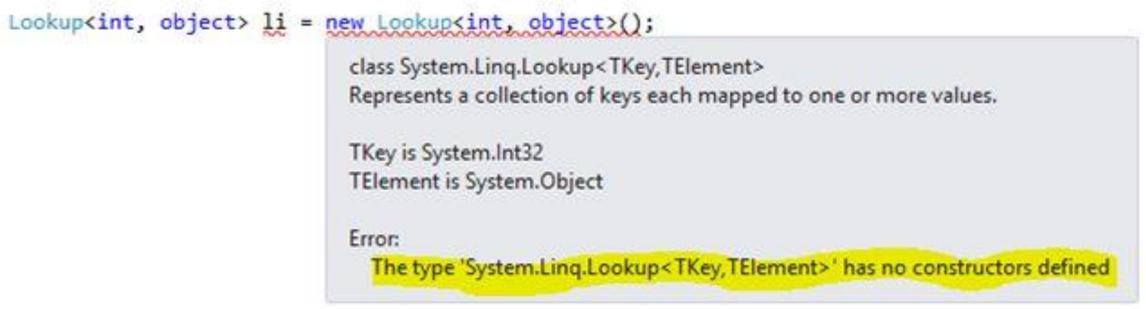
```
using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
    static void Main()
    {
        // Create a sample collection of data
        var data = new[]
        {
            new { Key = "A", Value = 1 },
            new { Key = "B", Value = 2 },
            new { Key = "A", Value = 3 },
            new { Key = "C", Value = 4 },
            new { Key = "B", Value = 5 }
        };

        // Create a mutable version using Dictionary and List
        var mutableLookup = new Dictionary<string, List<int>>();

        foreach (var item in data)
        {
            if (!mutableLookup.ContainsKey(item.Key))
            {
                mutableLookup[item.Key] = new List<int>();
            }

            mutableLookup[item.Key].Add(item.Value);
        }

        // Access values by key
        foreach (var entry in mutableLookup)
        {
            Console.WriteLine($"Key: {entry.Key}");

            foreach (var value in entry.Value)
            {
                Console.WriteLine($"  Value: {value}");
            }
        }
    }
}
```

The Lookup class in C# is immutable, meaning once it's created, you cannot add, remove, or modify its contents directly. However, you can create a new Lookup with additional elements using LINQ or other methods. Here's an example:

```
using System;
using System.Linq;

class Program
{
    static void Main()
    {
        // Create an initial Lookup
        var initialData = new[]
        {
            new { Key = "A", Value = 1 },
            new { Key = "B", Value = 2 },
            new { Key = "A", Value = 3 },
            new { Key = "C", Value = 4 },
            new { Key = "B", Value = 5 }
        };

        var initialLookup = initialData.ToLookup(item => item.Key, item
=> item.Value);

        // Create a new Lookup with additional elements
        var newData = new[]
        {
            new { Key = "A", Value = 6 },
            new { Key = "B", Value = 7 },
            new { Key = "C", Value = 8 },
            new { Key = "D", Value = 9 }
        };

        var newLookup = initialLookup.Concat(newData.ToLookup(item =>
item.Key, item => item.Value));

        // Access values in the new Lookup
        foreach (var key in newLookup)
        {
            Console.WriteLine($"Key: {key.Key}");

            foreach (var value in key)
            {
                Console.WriteLine($"  Value: {value}");
            }
        }
    }
}
```

We can create an object of a dictionary but we cannot create object of a lookup. In real time we only use this lookup to convert from one data type to another. For example, converting a list to Lookup, etc.

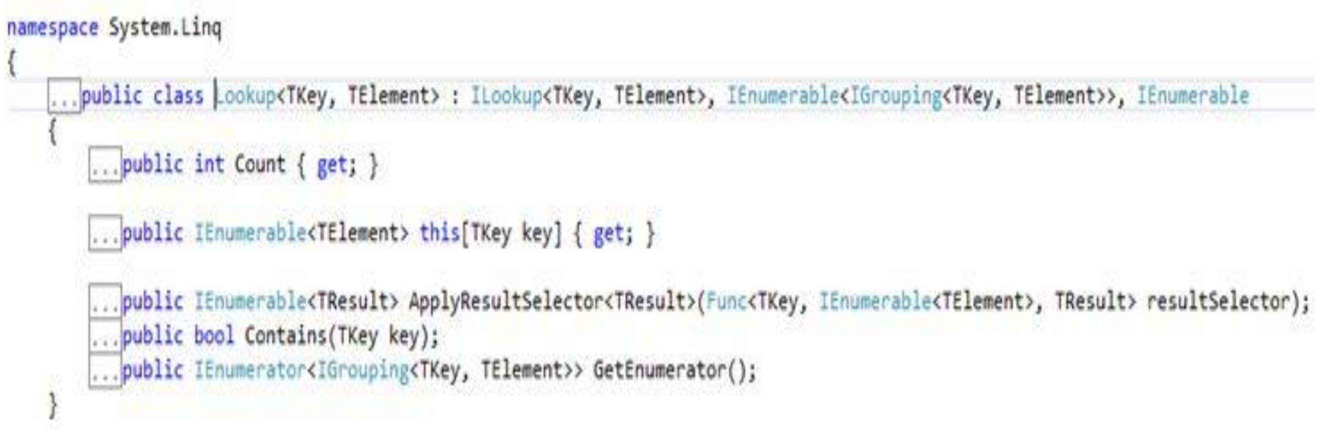
This is what error I got while creating the object of a Lookup.



```
Lookup<int, object> li = new Lookup<int, object>();
```

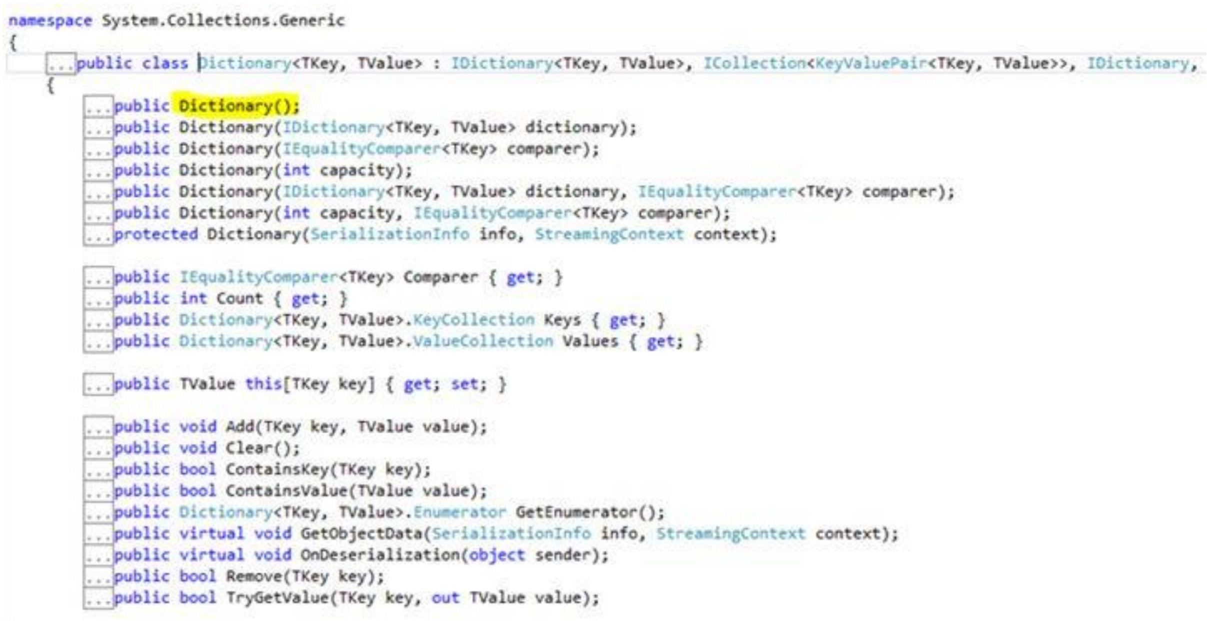
```
class System.Linq.Lookup<TKey,TElement>
Represents a collection of keys each mapped to one or more values.

TKey is System.Int32
TElement is System.Object

Error:
    The type 'System.Linq.Lookup<TKey,TElement>' has no constructors defined
```

Now if we check the Lookup by right clicking and going to the definition we didn't find any constructor defined in the Lookup class.



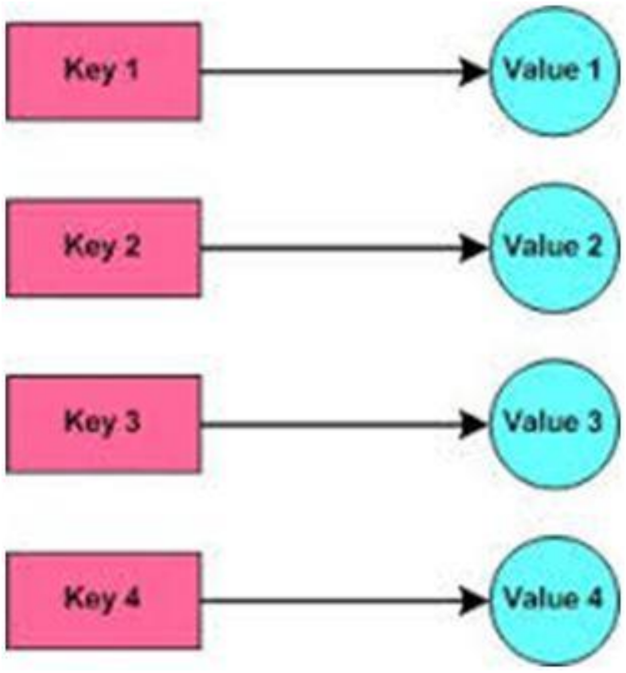
```
namespace System.Linq
{
    ...public class Lookup<TKey, TElement> : ILookup<TKey, TElement>, IEnumerable<IGrouping<TKey, TElement>>, IEnumerable
    {
        ...public int Count { get; }

        ...public IEnumerable<TElement> this[TKey key] { get; }

        ...public IEnumerable<TResult> ApplyResultSelector<TResult>(Func<TKey, IEnumerable<TElement>, TResult> resultSelector);
        ...public bool Contains(TKey key);
        ...public IEnumerator<IGrouping<TKey, TElement>> GetEnumerator();
    }
```

Now if we check the dictionary we can create the object of it and a constructor is defined in it as follows



```
namespace System.Collections.Generic
{
    ...public class Dictionary<TKey, TValue> : IDictionary<TKey, TValue>, ICollection<KeyValuePair<TKey, TValue>>, IDictionary,
    {
        ...public Dictionary();
        ...public Dictionary(IDictionary<TKey, TValue> dictionary);
        ...public Dictionary(IEqualityComparer<TKey> comparer);
        ...public Dictionary(int capacity);
        ...public Dictionary(IDictionary<TKey, TValue> dictionary, IEqualityComparer<TKey> comparer);
        ...public Dictionary(int capacity, IEqualityComparer<TKey> comparer);
        ...protected Dictionary(SerializationInfo info, StreamingContext context);

        ...public IEqualityComparer<TKey> Comparer { get; }
        ...public int Count { get; }
        ...public Dictionary<TKey, TValue>.KeyCollection Keys { get; }
        ...public Dictionary<TKey, TValue>.ValueCollection Values { get; }

        ...public TValue this[TKey key] { get; set; }

        ...public void Add(TKey key, TValue value);
        ...public void Clear();
        ...public bool ContainsKey(TKey key);
        ...public bool ContainsValue(TValue value);
        ...public Dictionary<TKey, TValue>.Enumerator GetEnumerator();
        ...public virtual void GetObjectData(SerializationInfo info, StreamingContext context);
        ...public virtual void OnDeserialization(object sender);
        ...public bool Remove(TKey key);
        ...public bool TryGetValue(TKey key, out TValue value);
```

In a dictionary we have a key value pair and the key in a dictionary cannot be duplicated. But in a Lookup we can have multiple values with a single key.



In LookUP