Read-only and Immutable Collections

Simon Robinson
SOFTWARE DEVELOPER

@techiesimon www.simonrobinson.com

Overview

Preventing modifications

- Read-only collections
- Immutable collections

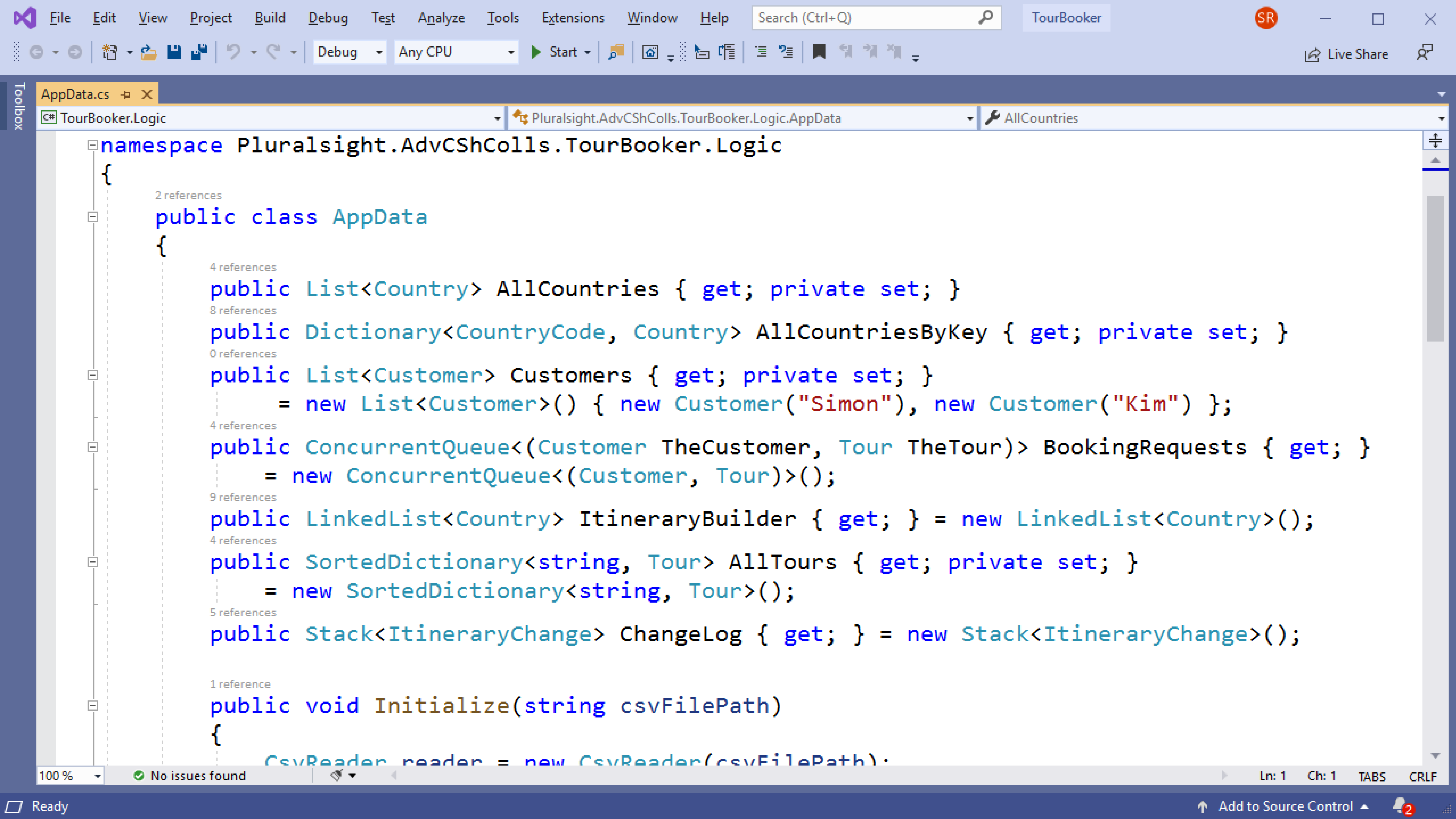
Types of Collections

Lists	Dictionaries	Linked lists
Stacks	Queues	Sets

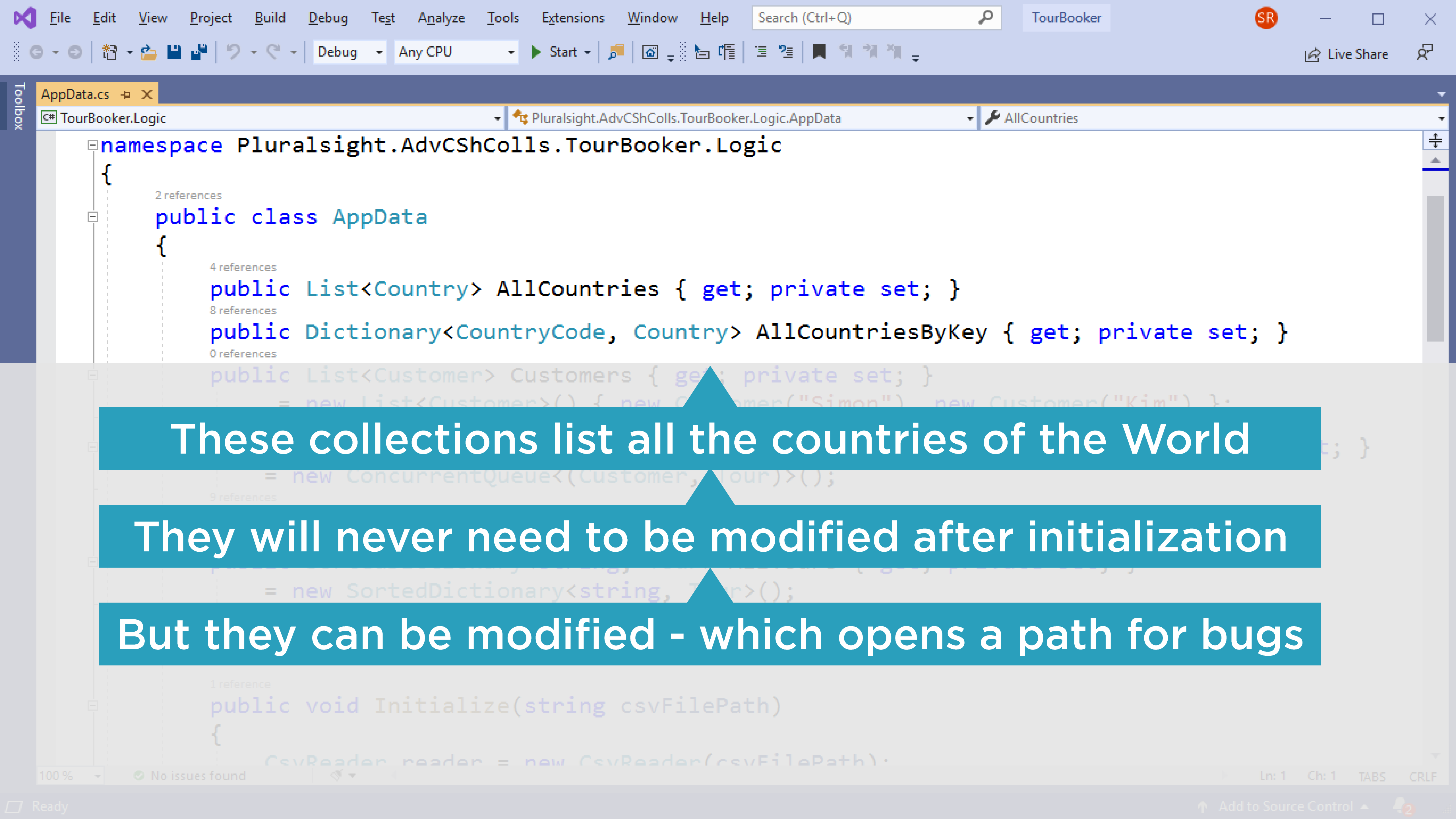
All are freely editable

But freely editable isn't always appropriate

Concurrency



```
namespace Pluralsight.AdvCShColls.TourBooker.Logic
{
        2 references
    public class AppData
    {
            4 references
        public List<Country> AllCountries { get; private set; }
            8 references
        public Dictionary<CountryCode, Country> AllCountriesByKey { get; private set; }
            0 references
        public List<Customer> Customers { get; private set; }
             = new List<Customer>() { new Customer("Simon"), new Customer("Kim") };
            4 references
        public ConcurrentQueue<(Customer TheCustomer, Tour TheTour)> BookingRequests { get; }
            = new ConcurrentQueue<(Customer, Tour)>();
            9 references
        public LinkedList<Country> ItineraryBuilder { get; } = new LinkedList<Country>();
            4 references
        public SortedDictionary<string, Tour> AllTours { get; private set; }
            = new SortedDictionary<string, Tour>();
            5 references
        public Stack<ItineraryChange> ChangeLog { get; } = new Stack<ItineraryChange>();

            1 reference
        public void Initialize(string csvFilePath)
        {
            CsvReader reader = new CsvReader(csvFilePath);
```

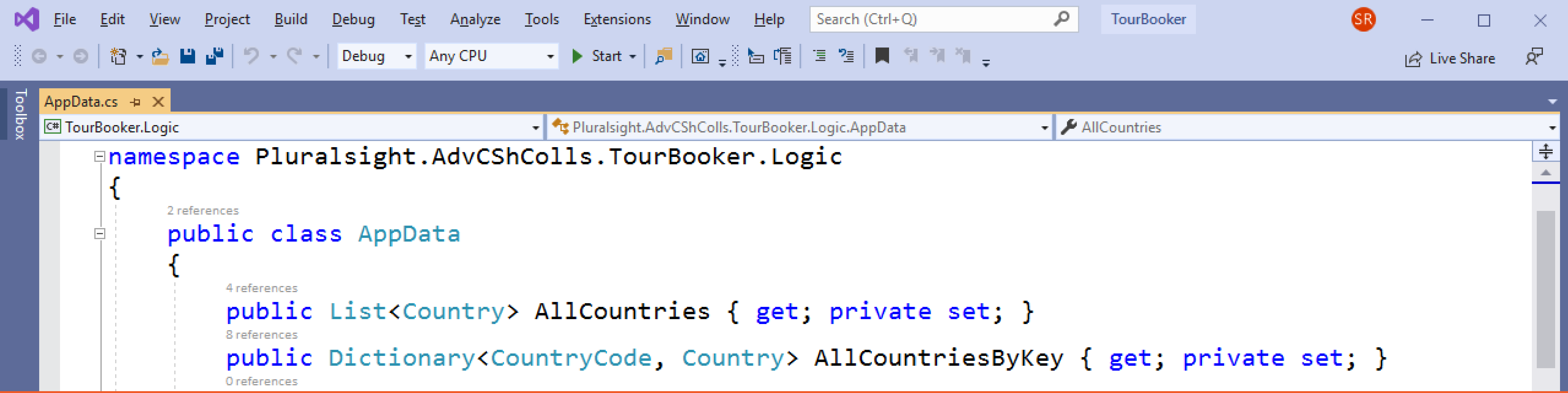

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
TourBooker
Debug
Any CPU
Start
Live Share
AppData.cs
TourBooker.Logic
Pluralsight.AdvCShColls.TourBooker.Logic.AppData
AllCountries

```
namespace Pluralsight.AdvCShColls.TourBooker.Logic
{
    2 references
    public class AppData
    {
        4 references
        public List<Country> AllCountries { get; private set; }
        8 references
        public Dictionary<CountryCode, Country> AllCountriesByKey { get; private set; }
        0 references
        public List<Customer> Customers { get; private set; }
            = new List<Customer>() { new Customer("Simon"), new Customer("Kim") };

            = new ConcurrentQueue<(Customer, Tour)>();

            = new SortedDictionary<string, Tour>();

        1 reference
        public void Initialize(string csvFilePath)
        {
            CsvReader reader = new CsvReader(csvFilePath);
```

These collections list all the countries of the World

They will never need to be modified after initialization

But they can be modified - which opens a path for bugs



```
namespace Pluralsight.AdvCShColls.TourBooker.Logic
{
    2 references
    public class AppData
    {
        4 references
        public List<Country> AllCountries { get; private set; }
        8 references
        public Dictionary<CountryCode, Country> AllCountriesByKey { get; private set; }
        0 references
```

To prevent changes to collections

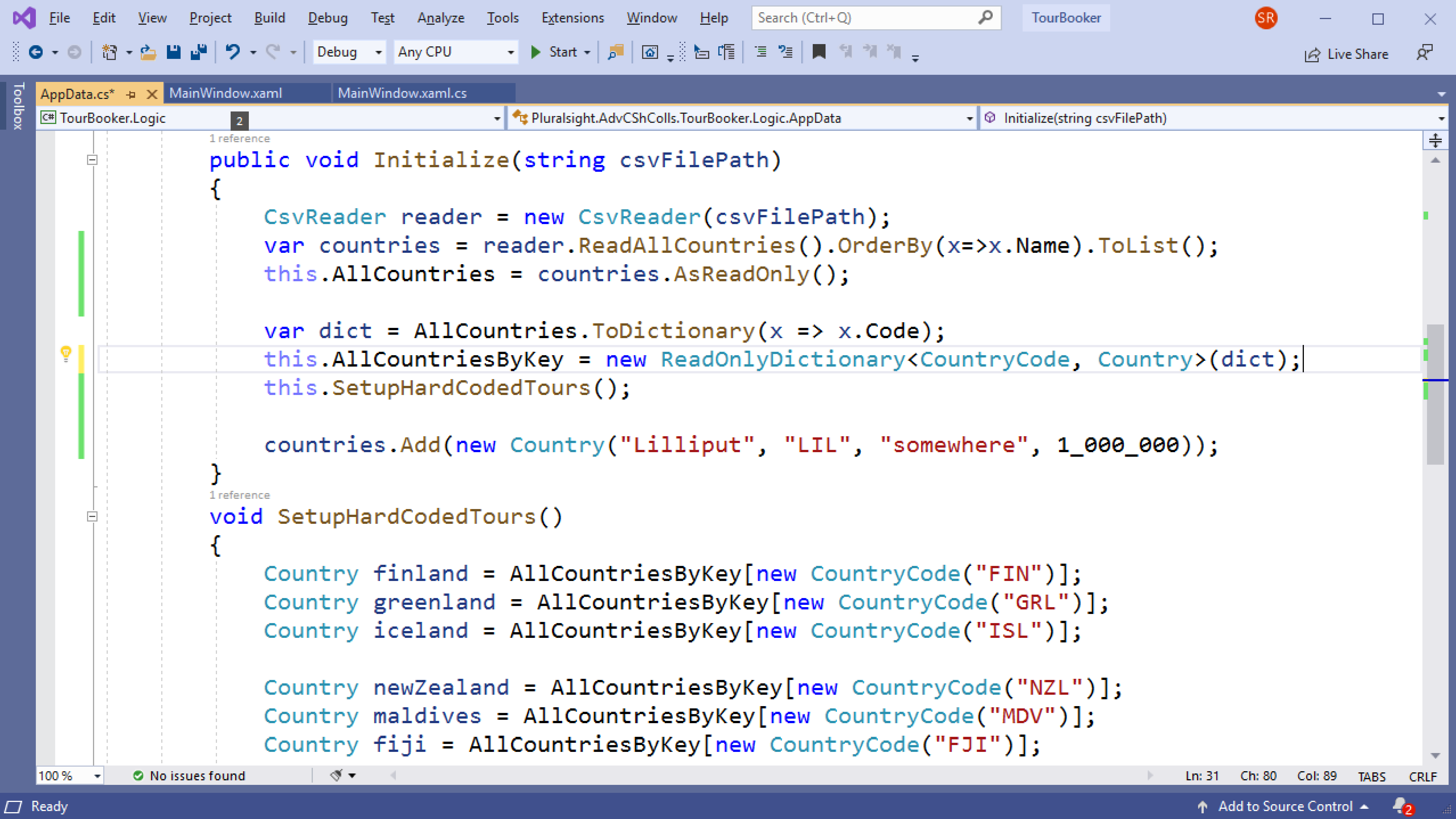
Read-only collections

Immutable collections

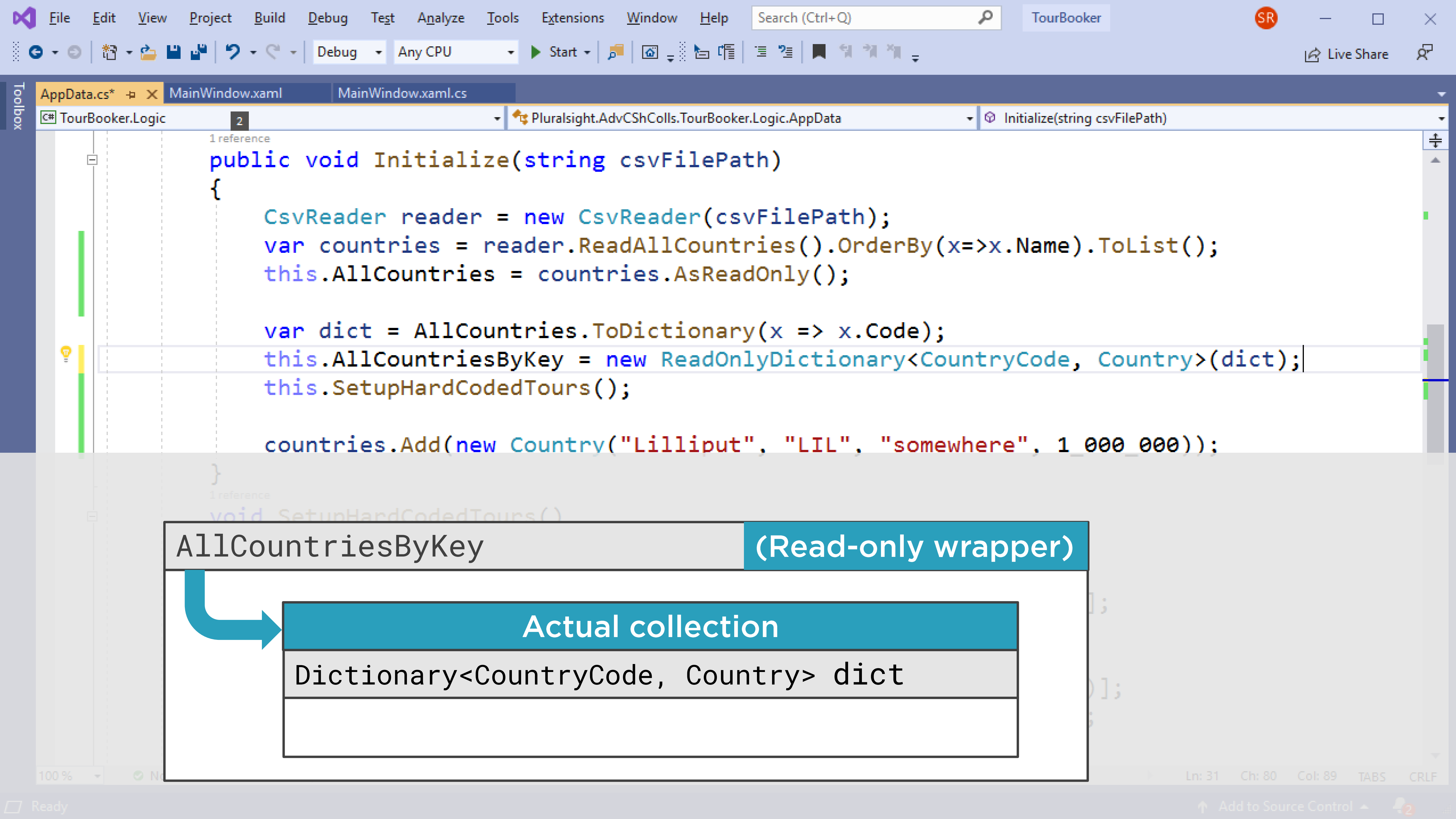
Demo

Convert collections to read-only

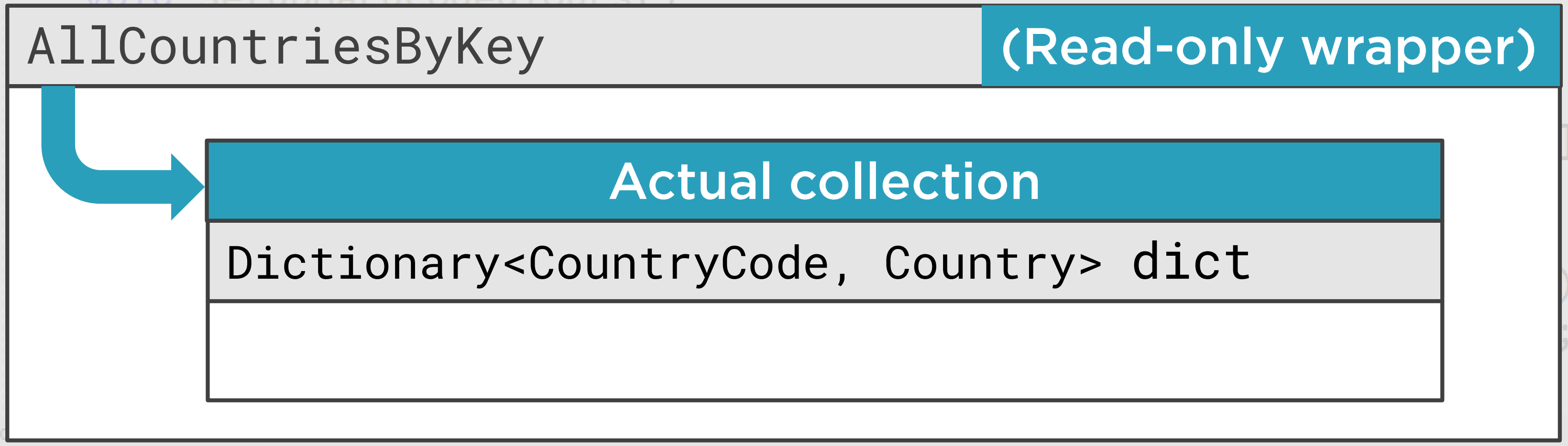
- The `AllCountries` list
- The `AllCountriesByKey` dictionary

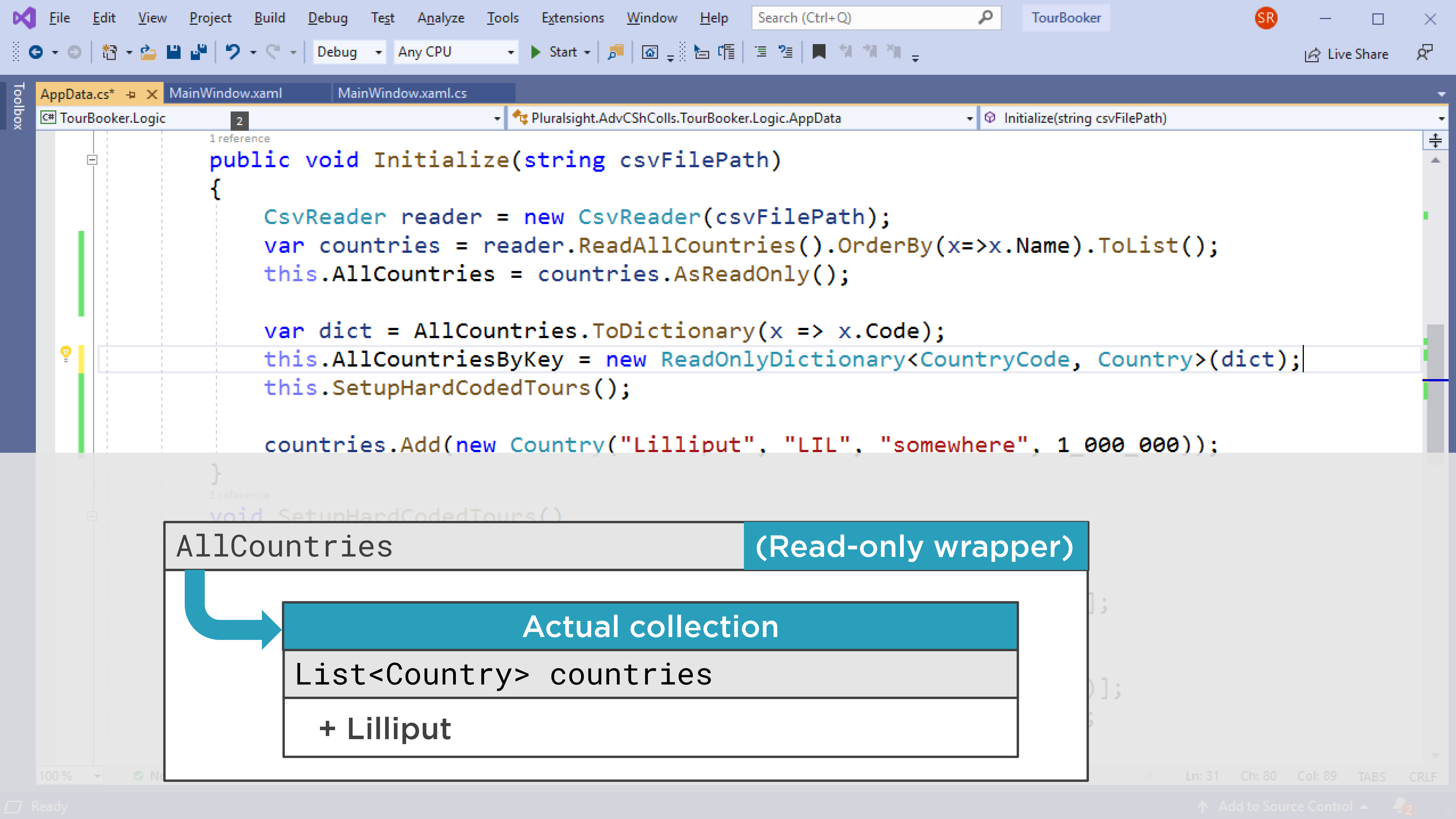


```
        1 reference
        public void Initialize(string csvFilePath)
        {
            CsvReader reader = new CsvReader(csvFilePath);
            var countries = reader.ReadAllCountries().OrderBy(x=>x.Name).ToList();
            this.AllCountries = countries.AsReadOnly();

            var dict = AllCountries.ToDictionary(x => x.Code);
            this.AllCountriesByKey = new ReadOnlyDictionary<CountryCode, Country>(dict);
            this.SetupHardCodedTours();

            countries.Add(new Country("Lilliput", "LIL", "somewhere", 1_000_000));
        }
        1 reference
        void SetupHardCodedTours()
        {
            Country finland = AllCountriesByKey[new CountryCode("FIN")];
            Country greenland = AllCountriesByKey[new CountryCode("GRL")];
            Country iceland = AllCountriesByKey[new CountryCode("ISL")];

            Country newZealand = AllCountriesByKey[new CountryCode("NZL")];
            Country maldives = AllCountriesByKey[new CountryCode("MDV")];
            Country fiji = AllCountriesByKey[new CountryCode("FJI")];
```

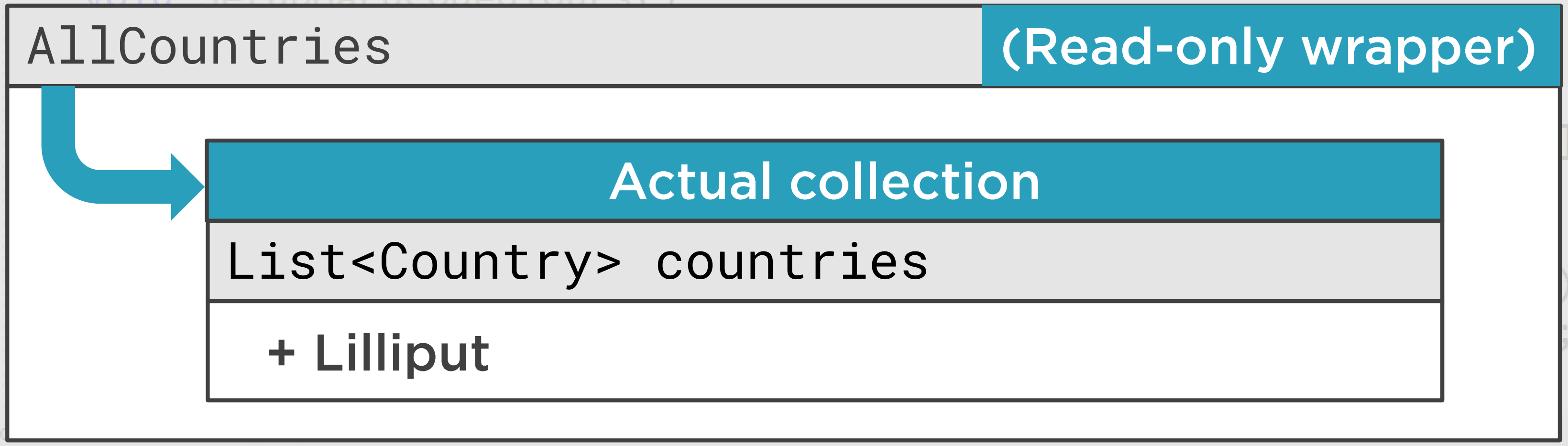


```
1 reference
public void Initialize(string csvFilePath)
{
    CsvReader reader = new CsvReader(csvFilePath);
    var countries = reader.ReadAllCountries().OrderBy(x=>x.Name).ToList();
    this.AllCountries = countries.AsReadOnly();

    var dict = AllCountries.ToDictionary(x => x.Code);
    this.AllCountriesByKey = new ReadOnlyDictionary<CountryCode, Country>(dict);
    this.SetupHardCodedTours();

    countries.Add(new Country("Lilliput", "LIL", "somewhere", 1_000_000));
}
```



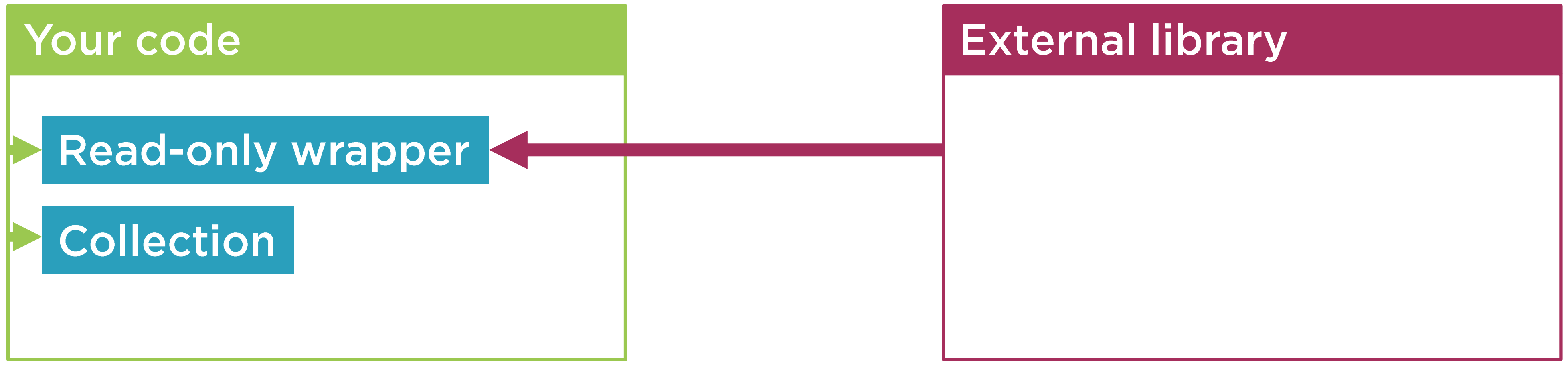


```
1 reference
public void Initialize(string csvFilePath)
{
    CsvReader reader = new CsvReader(csvFilePath);
    var countries = reader.ReadAllCountries().OrderBy(x=>x.Name).ToList();
    this.AllCountries = countries.AsReadOnly();

    var dict = AllCountries.ToDictionary(x => x.Code);
    this.AllCountriesByKey = new ReadOnlyDictionary<CountryCode, Country>(dict);
    this.SetupHardCodedTours();

    countries.Add(new Country("Lilliput", "LIL", "somewhere", 1_000_000));
}
```



Read-only collections can be modified – if you have a reference to the underlying collection

Read-only Collections



Your code can modify the collection

The external library can't

But if you want your collection to be completely fixed:

Immutable Collections

Demo

Convert collections to immutable

- The `AllCountries` list
- The `AllCountriesByKey` dictionary

Immutable collections are immutable

- Immutable against normal C# code
- Can circumvent with reflection or unmanaged code
- Protect against **accidental** modifications (not malicious code)

Immutable vs. Read-only

Immutable collections

Collections in their own right

`countries.ToImmutableArray()`

Copies items in `countries` to a brand new collection

The immutable collection doesn't see changes to `countries`

Read-only collections

Wrappers that guard other collections

`countries.AsReadOnly()`

Creates thin wrapper around `countries`

The read-only collection sees changes to `countries`

Concurrency

Immutable collections →	Immutable →	Thread-safe	✓
Standard generic collections →	No internal synchronization →	Not thread-safe	✗
Concurrent collections →	Internal synchronization →	Thread-safe	✓

Concurrency

If your collections must be accessed from multiple threads:

For reading:	For writing:	Do not use:
Immutable collections ✓	Concurrent collections ✓	Standard generic collections ✗

Summary

Write protection

- Read-only collections
  - Simple wrappers
  - Can modify with access to underlying collection
- Immutable collections
  - Immutable once constructed
  - Thread-safe

Next up: Interfaces