Introduction to Collections

Simon Robinson
http://TechieSimon.com
@TechieSimon

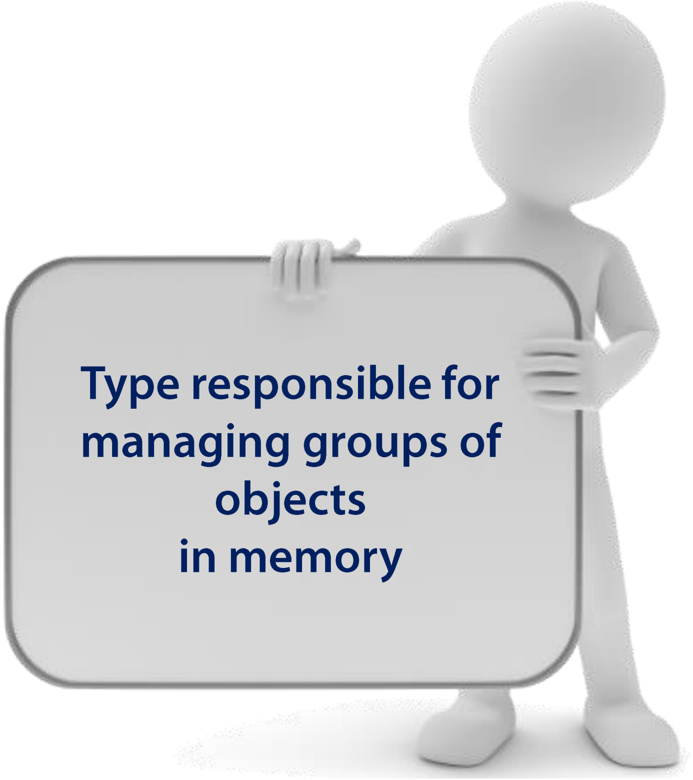
pluralsight
hardcore developer training

Overview

- **What is a collection**
  - Collection operations
- **Types of collection**
  - Lists
  - Dictionaries
  - Sets
- **Collections in .NET**
  - A brief history
  - The current collection landscape

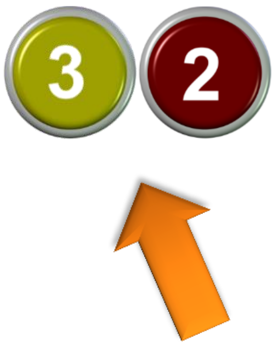
High level
– won't look at individual classes (yet)

What is a Collection?

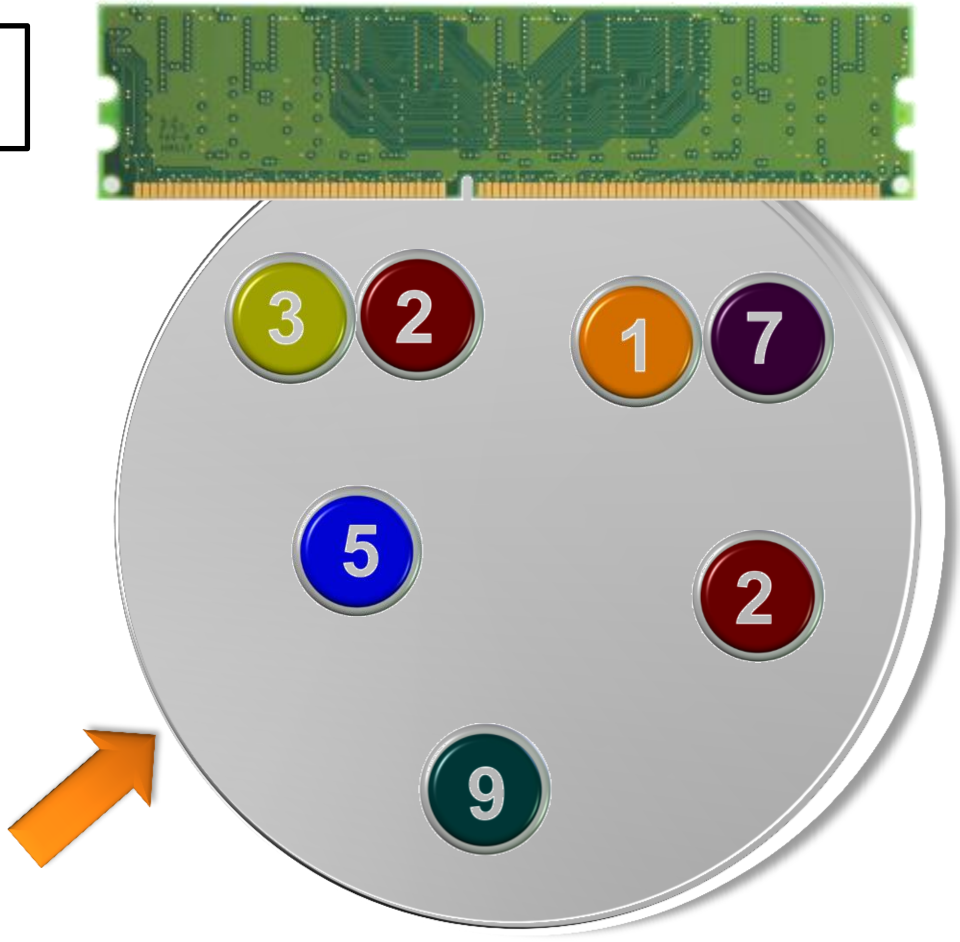


Type responsible for managing groups of objects in memory

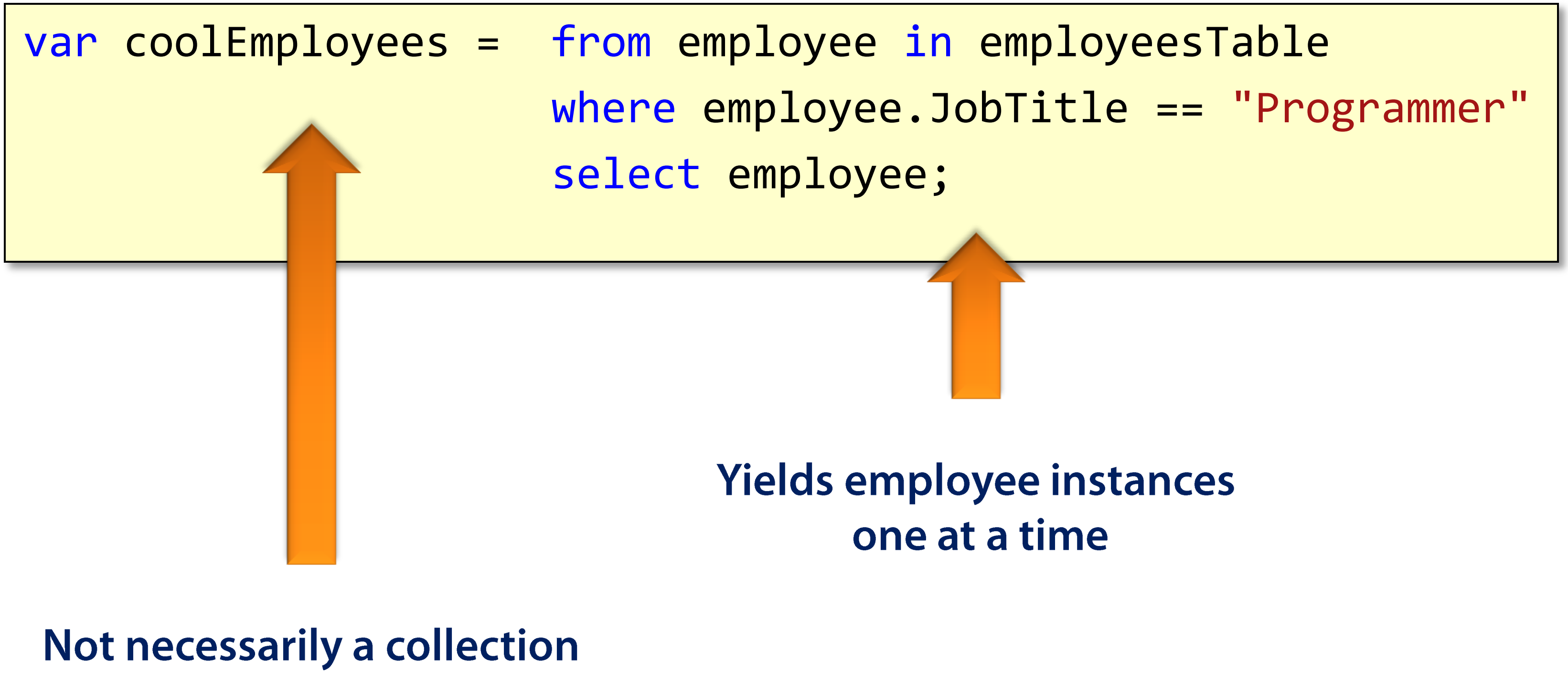
In memory!



An integer!



A collection
(of integers)



```
var coolEmployees =  from employee in employeesTable
                     where employee.JobTitle == "Programmer"
                     select employee;
```

Yields employee instances
one at a time

Not necessarily a collection

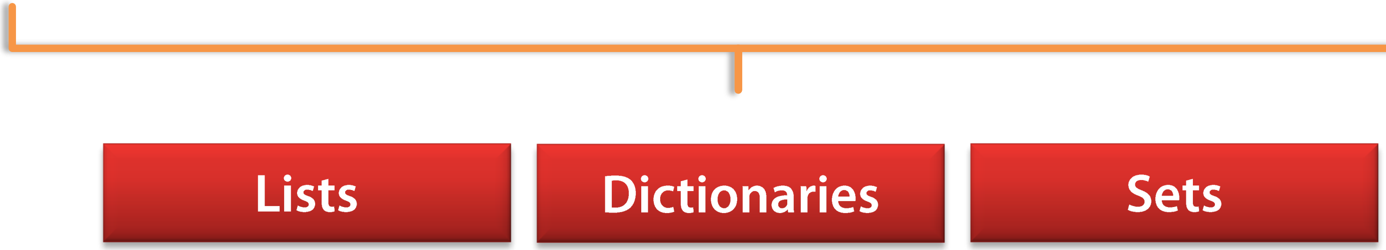
```
var coolEmployees = (from employee in employeesTable
                     where employee.JobTitle == "Programmer"
                     select employee).ToList();
```

`ToList()` **puts all the items in a list**

This IS a collection
(a `List<Employee>`)

Many collection classes…

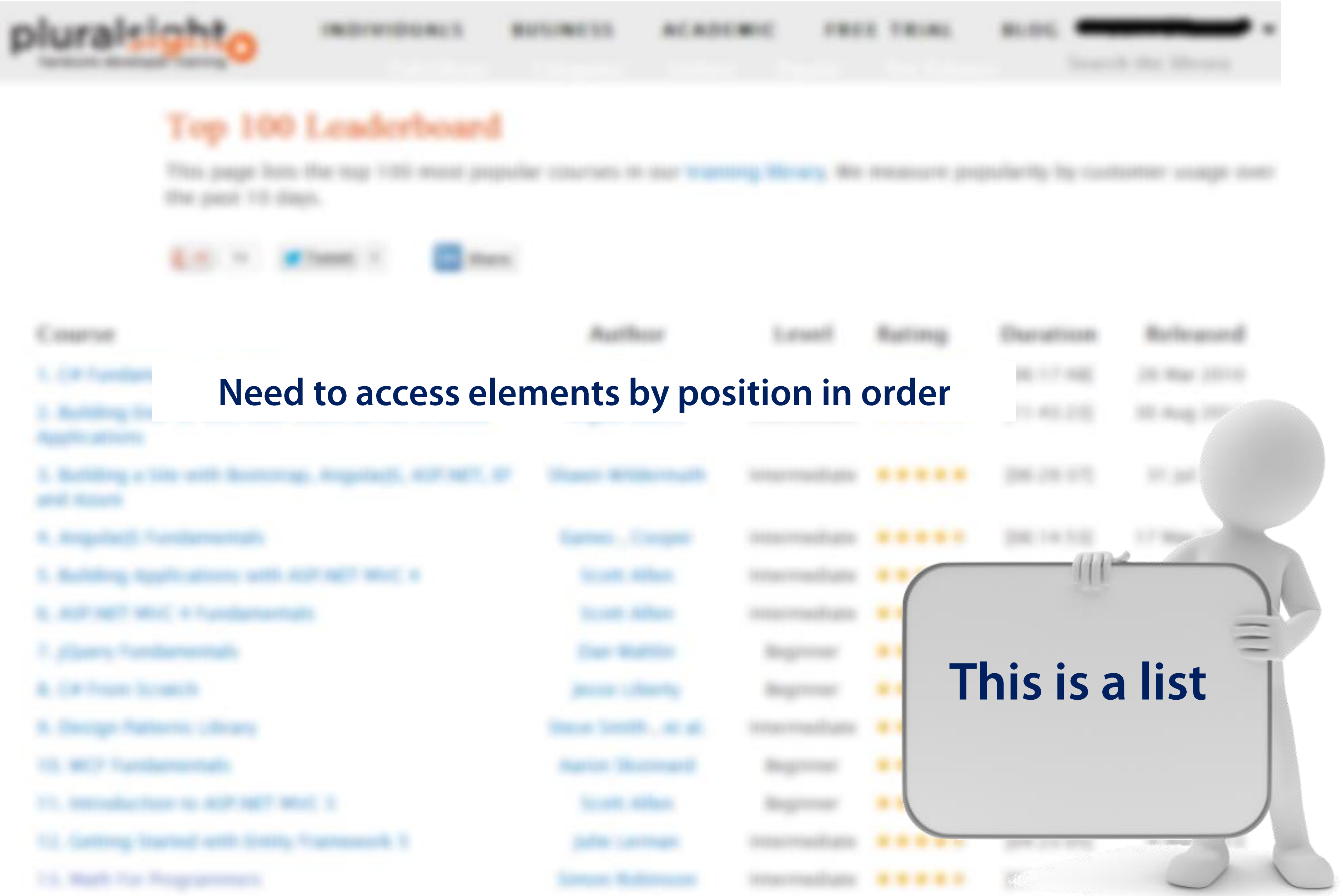
```
ReadOnlyCollection<T>
SortedList<TKey, TValue>
LinkedList<T>
Stack<T>
ObservableCollection<T>
HashSet<T>
Array
List<T>
SortedSet<T>
Collection<T>
Dictionary<TKey, TValue>
KeyedCollection<TKey, TItem>
```



Lists

Dictionaries

Sets



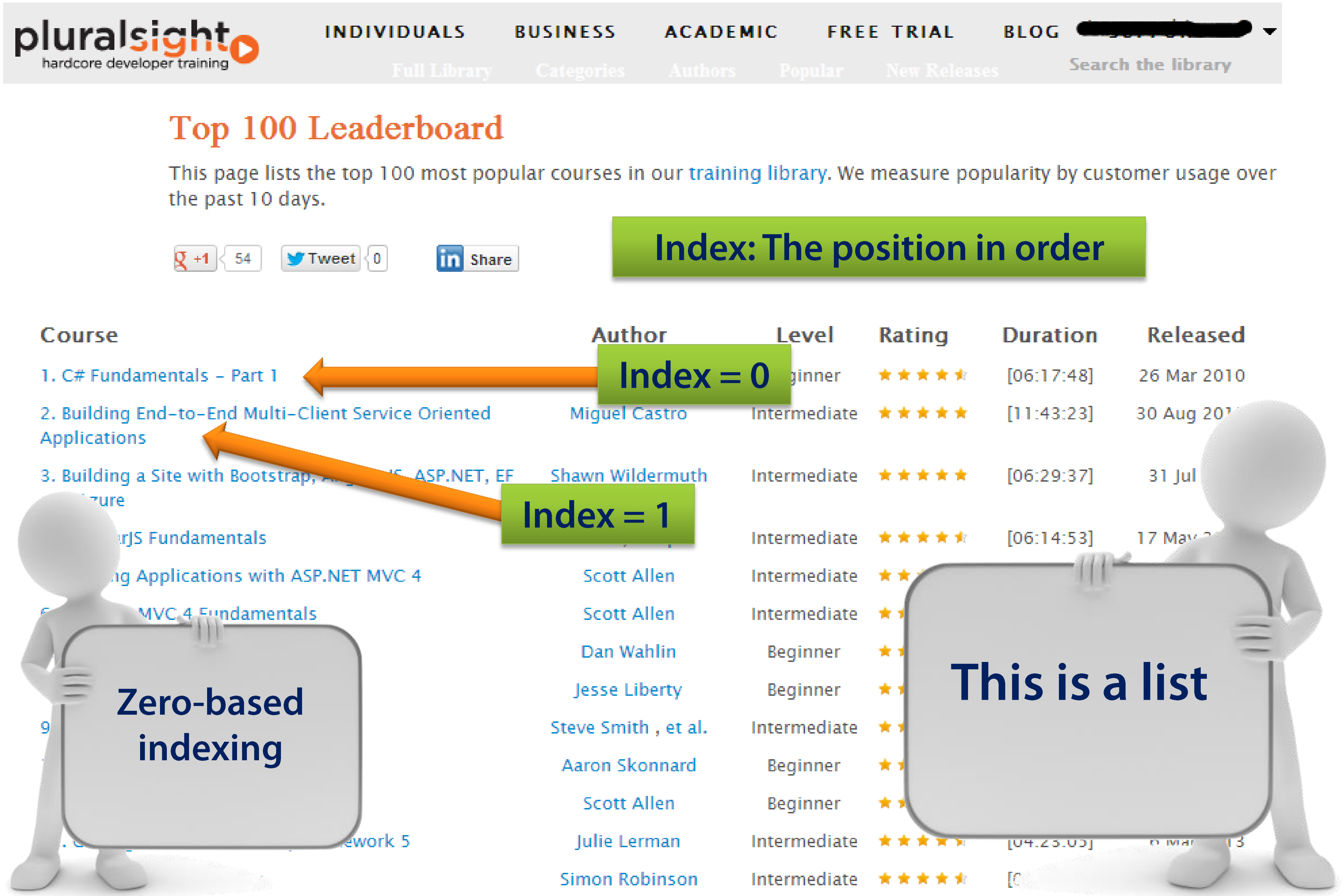
Top 100 Leaderboard

This page lists the top 100 most popular courses in our training library. We measure popularity by customer usage over the past 15 days.

Course	Author	Level	Rating	Duration	Released
1. C# Fundam[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2. Building [illegible] Applications	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
3. Building a Site with Bootstrap, AngularJS, ASP.NET, EF and Azure	Shawn Wildermuth	Intermediate	★★★★★	[illegible]	[illegible]
4. AngularJS Fundamentals	[illegible]	Intermediate	★★★★★	[illegible]	[illegible]
5. Building Applications with ASP.NET MVC 4	Scott Allen	Intermediate	[illegible]	[illegible]	[illegible]
6. ASP.NET MVC 4 Fundamentals	Scott Allen	Intermediate	[illegible]	[illegible]	[illegible]
7. jQuery Fundamentals	Dan Wahlin	Beginner	[illegible]	[illegible]	[illegible]
8. C# From Scratch	Jesse Liberty	Beginner	[illegible]	[illegible]	[illegible]
9. Design Patterns Library	Steve Smith, et al	Intermediate	[illegible]	[illegible]	[illegible]
10. WCF Fundamentals	Aaron Skonnard	Beginner	[illegible]	[illegible]	[illegible]
11. Introduction to ASP.NET MVC 3	Scott Allen	Beginner	[illegible]	[illegible]	[illegible]
12. Getting Started with Entity Framework 5	Julie Lerman	Intermediate	★★★★★	[illegible]	[illegible]
13. Math For Programmers	Simon Robinson	Intermediate	★★★★★	[illegible]	[illegible]

Need to access elements by position in order

This is a list



Top 100 Leaderboard

This page lists the top 100 most popular courses in our training library. We measure popularity by customer usage over the past 10 days.

Course	Author	Level	Rating	Duration	Released
1. C# Fundamentals - Part 1		ginner	★★★★½	[06:17:48]	26 Mar 2010
2. Building End-to-End Multi-Client Service Oriented Applications	Miguel Castro	Intermediate	★★★★★	[11:43:23]	30 Aug 201
3. Building a Site with Bootstrap, A JS, ASP.NET, EF zure	Shawn Wildermuth	Intermediate	★★★★★	[06:29:37]	31 Jul
rJS Fundamentals		Intermediate	★★★★½	[06:14:53]	17 May
ng Applications with ASP.NET MVC 4	Scott Allen	Intermediate	★★		
6 MVC 4 Fundamentals	Scott Allen	Intermediate	★		
	Dan Wahlin	Beginner	★		
	Jesse Liberty	Beginner	★		
9	Steve Smith , et al.	Intermediate	★		
	Aaron Skonnard	Beginner	★		
	Scott Allen	Beginner	★		
. ework 5	Julie Lerman	Intermediate	★★★★	[04:23:05]	6 Mar 13
	Simon Robinson	Intermediate	★★★★½	[0	

Index: The position in order

Index = 0

Index = 1

Zero-based indexing

This is a list

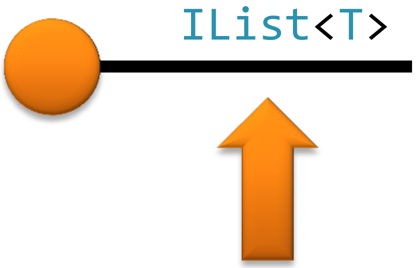
List Types in .NET:

`T[]`   `List<T>`

`Collection<T>`

`ReadOnlyCollection<T>`

`ObservableCollection<T>`



`IList<T>`

Contract for index-based lists

Good for memory use

Efficient for accessing elements

Dictionaries

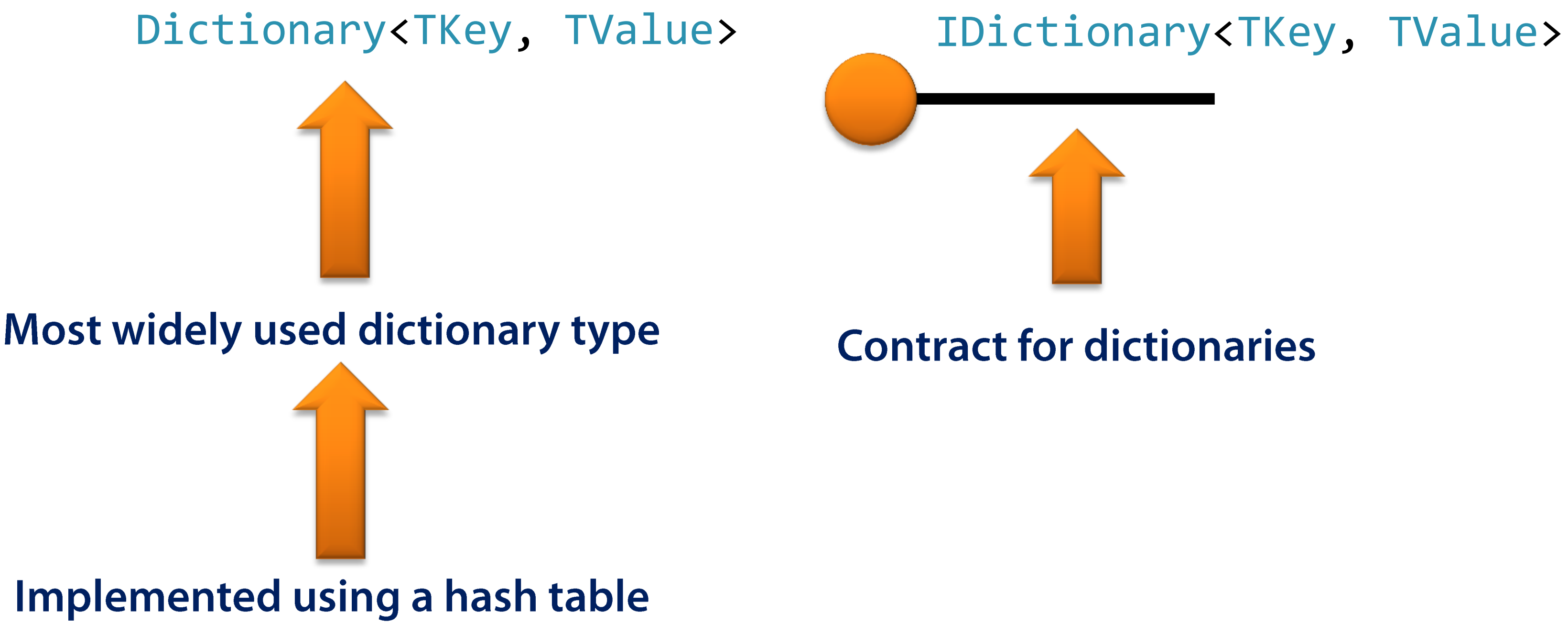
Collection of employee instances

```
class Employee {…
```

Accessing items by index
is not required…

This is a dictionary

Access elements using a **key**



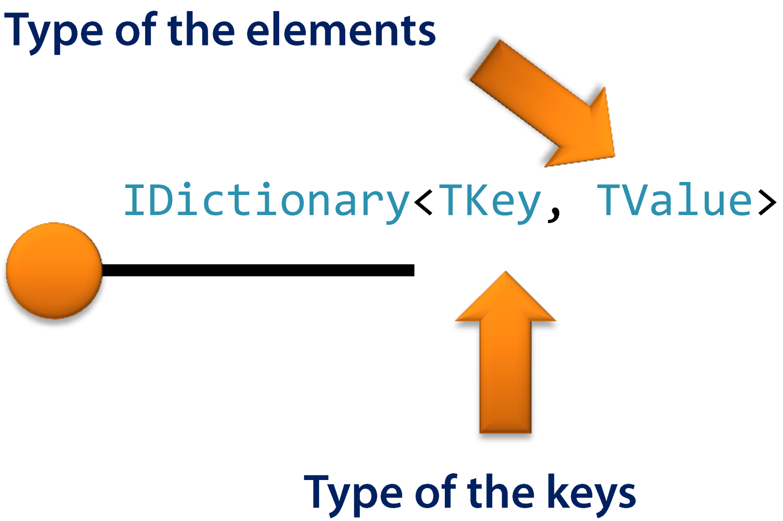
`Dictionary<TKey, TValue>`

Most widely used dictionary type

Implemented using a hash table

`IDictionary<TKey, TValue>`

Contract for dictionaries



Type of the elements

```
IDictionary<TKey, TValue>
```

Type of the keys

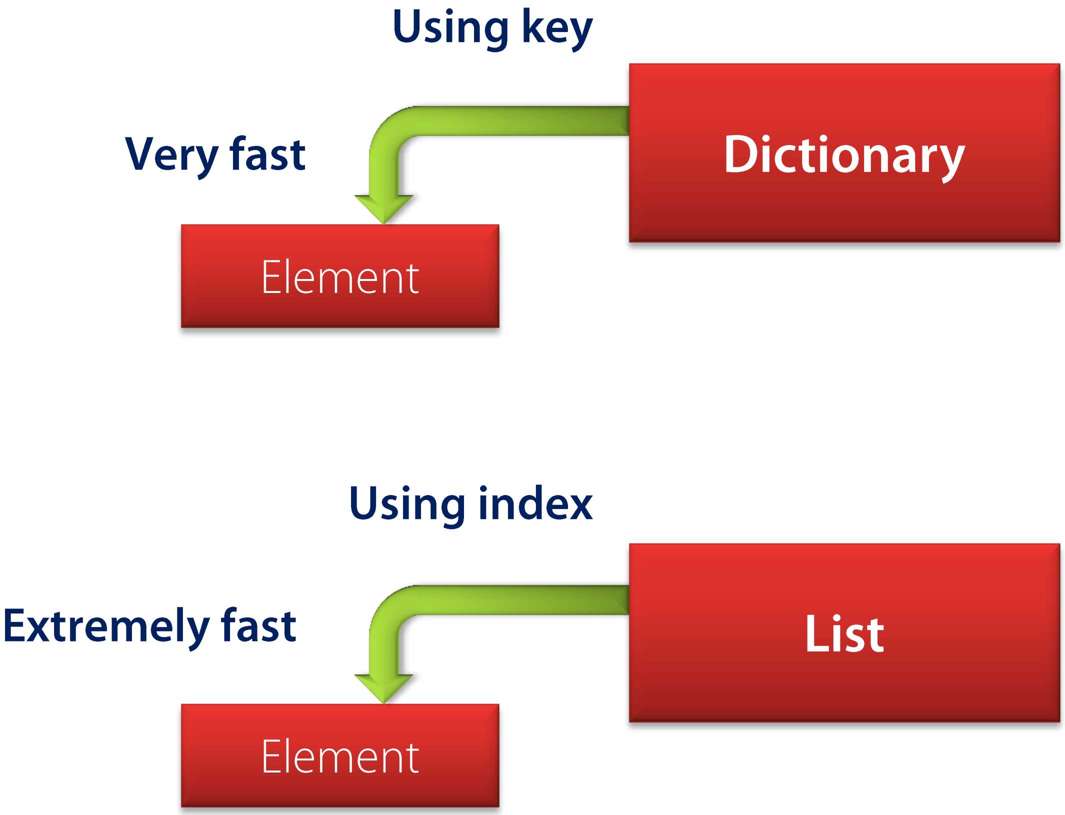
Declare a dictionary that allows looking up by name:

```
var employees =
   new Dictionary<string, Employee>();
```

For looking up by social security no., declare:

```
var employees = new Dictionary<SocialSecNo, Employee>();
```

Looking up an element



Using key

Dictionary

Very fast

Element

Using index

List

Extremely fast

Element

Dictionaries

Very useful

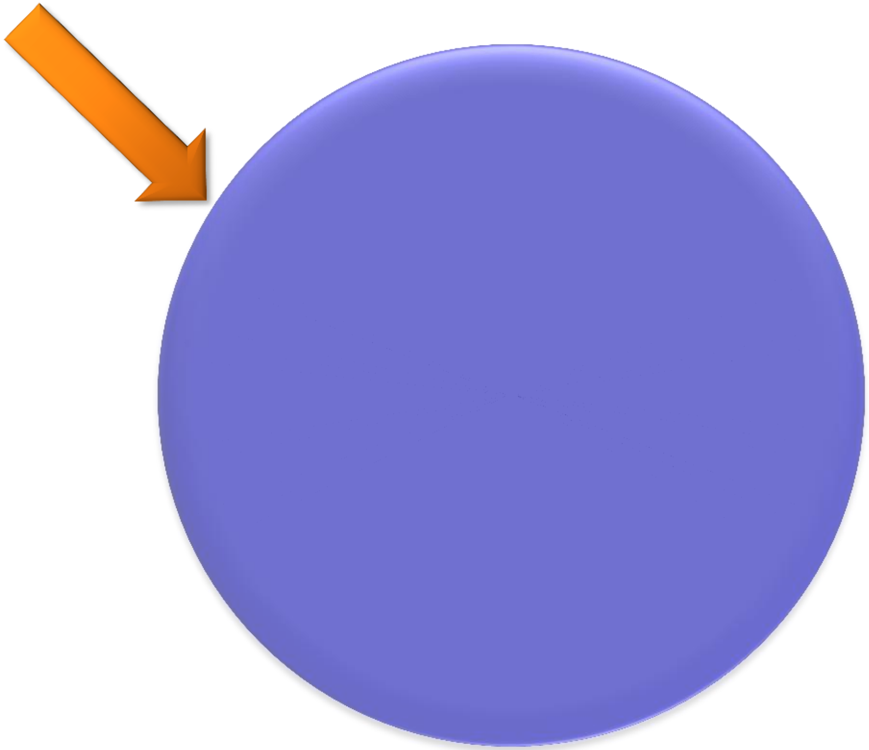
Easy to misuse

Expect keys to behave a certain way

(Cover later in this course)

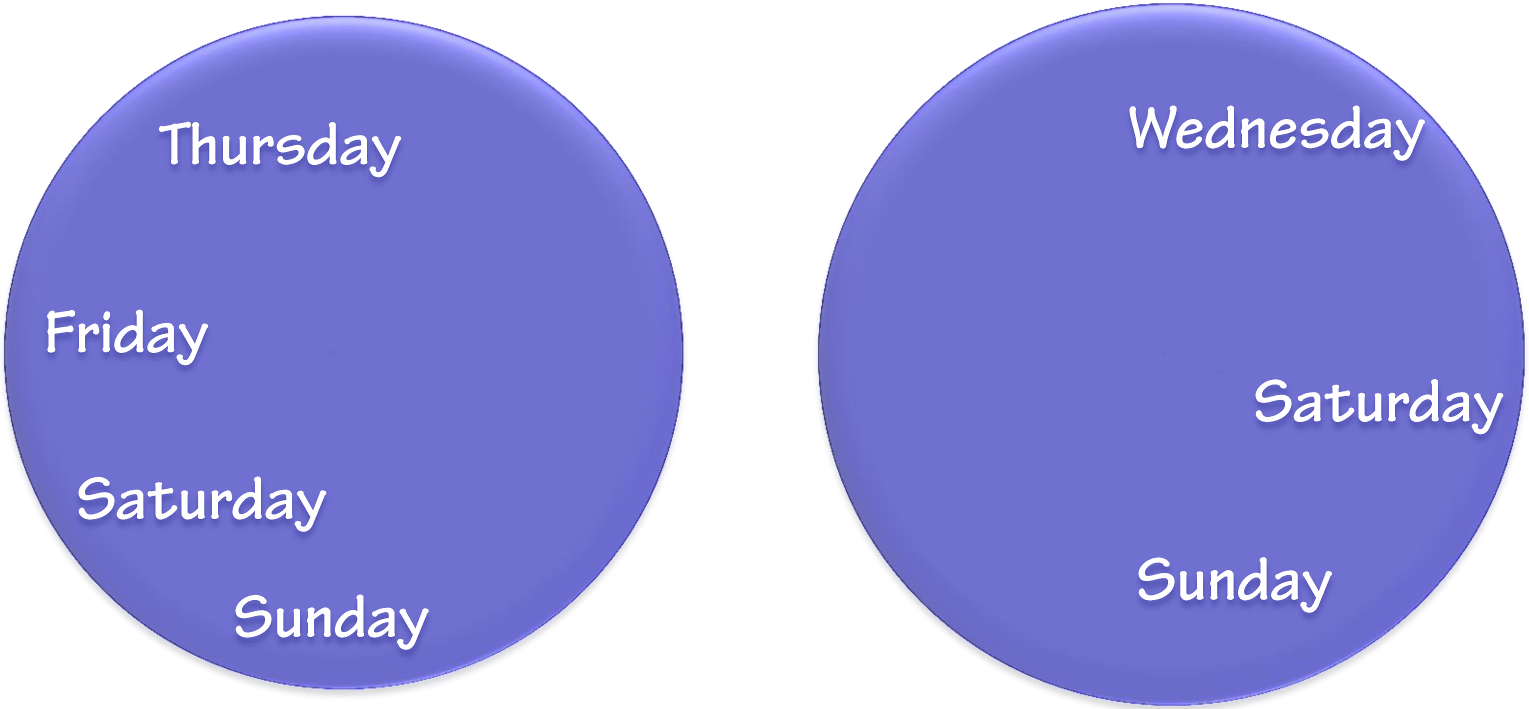
Sets

Sets treat the collection as a whole



Sets

What days are in both collections?



Thursday

Friday

Saturday

Sunday

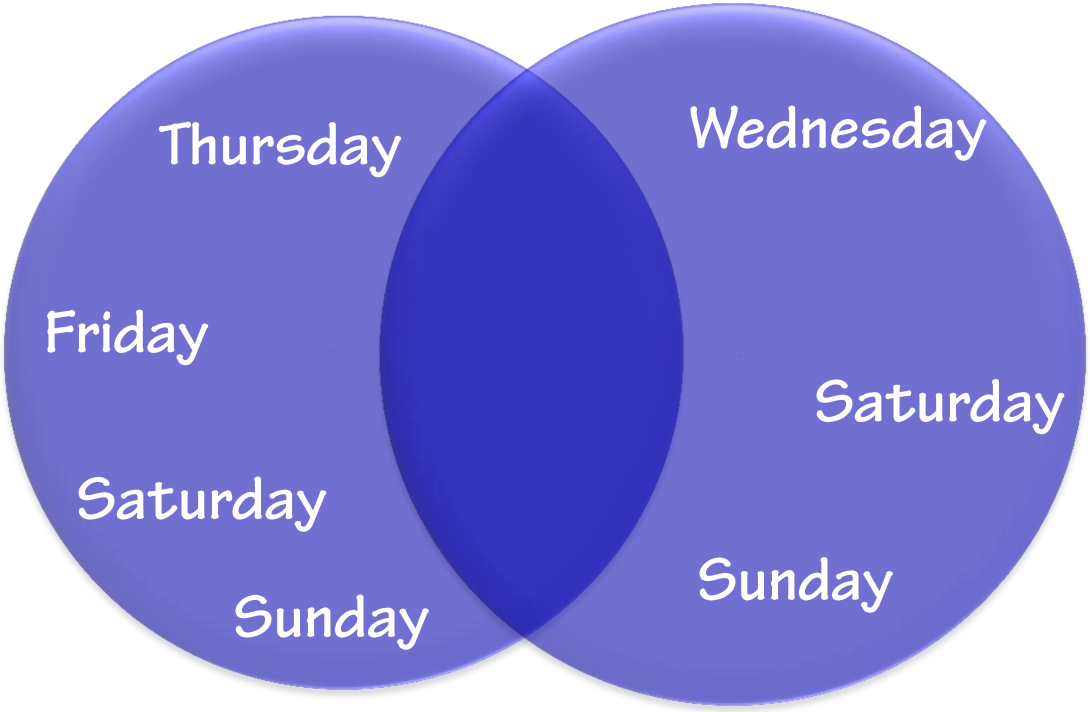
Wednesday

Saturday

Sunday

Sets

What days are in both collections?



Thursday

Friday

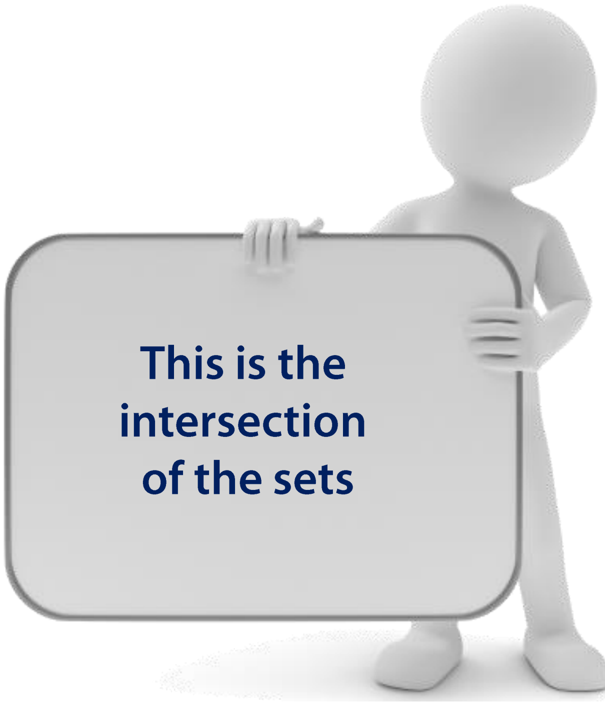
Saturday

Sunday

Wednesday

Saturday

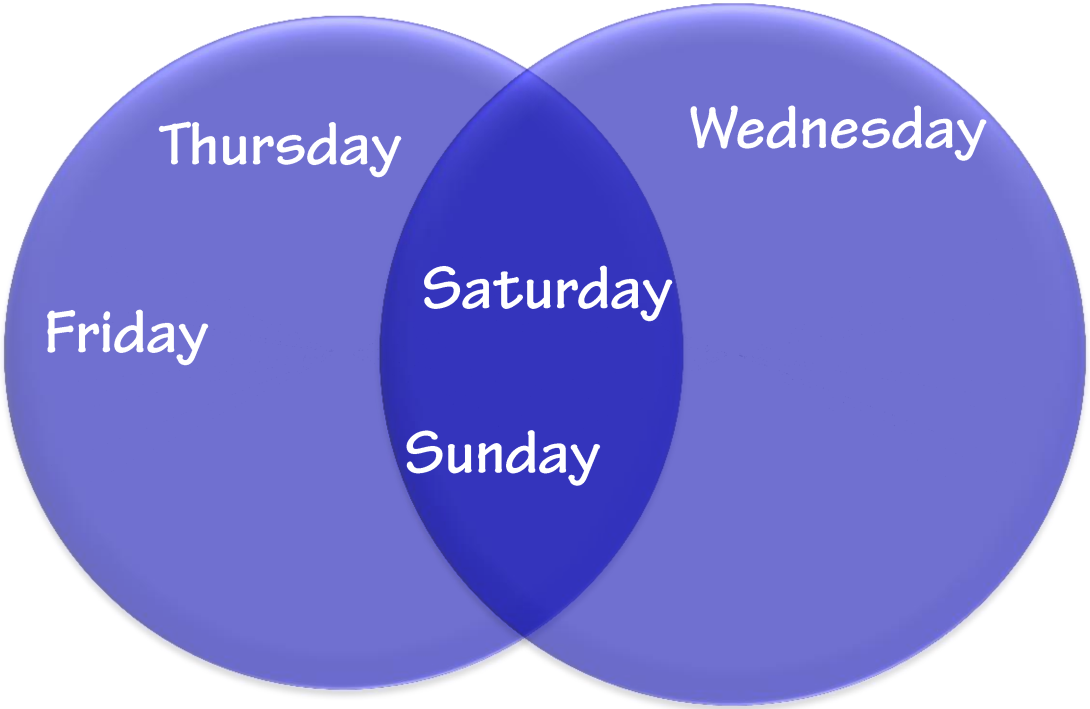
Sunday



This is the intersection of the sets

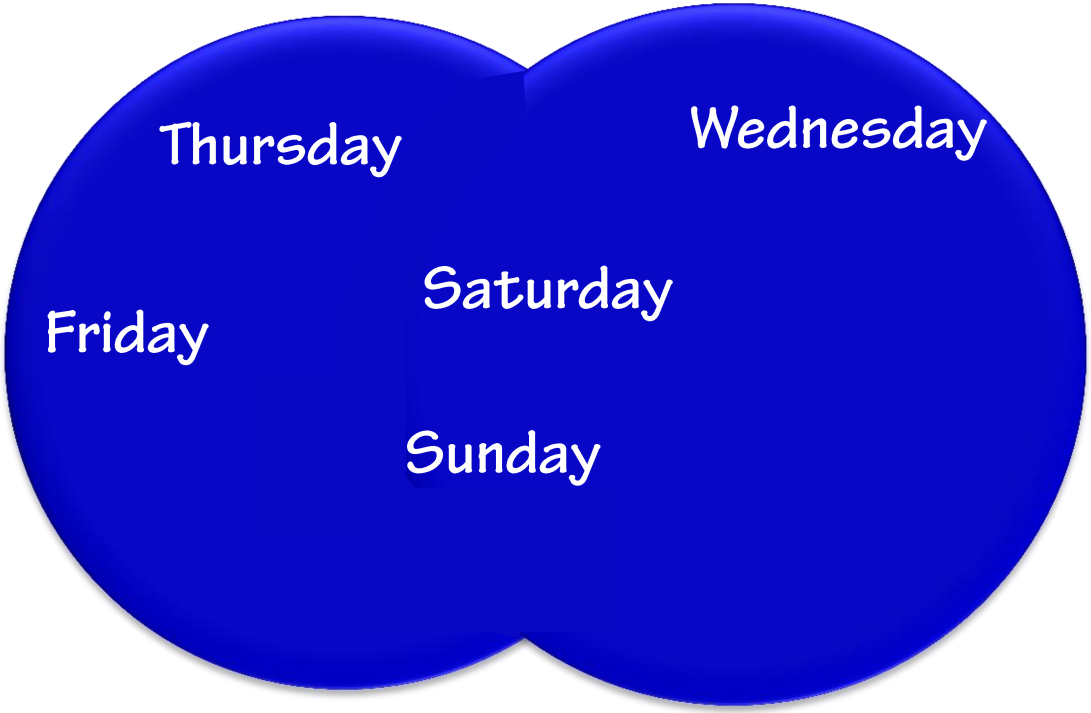
Sets

What days are in either collection?



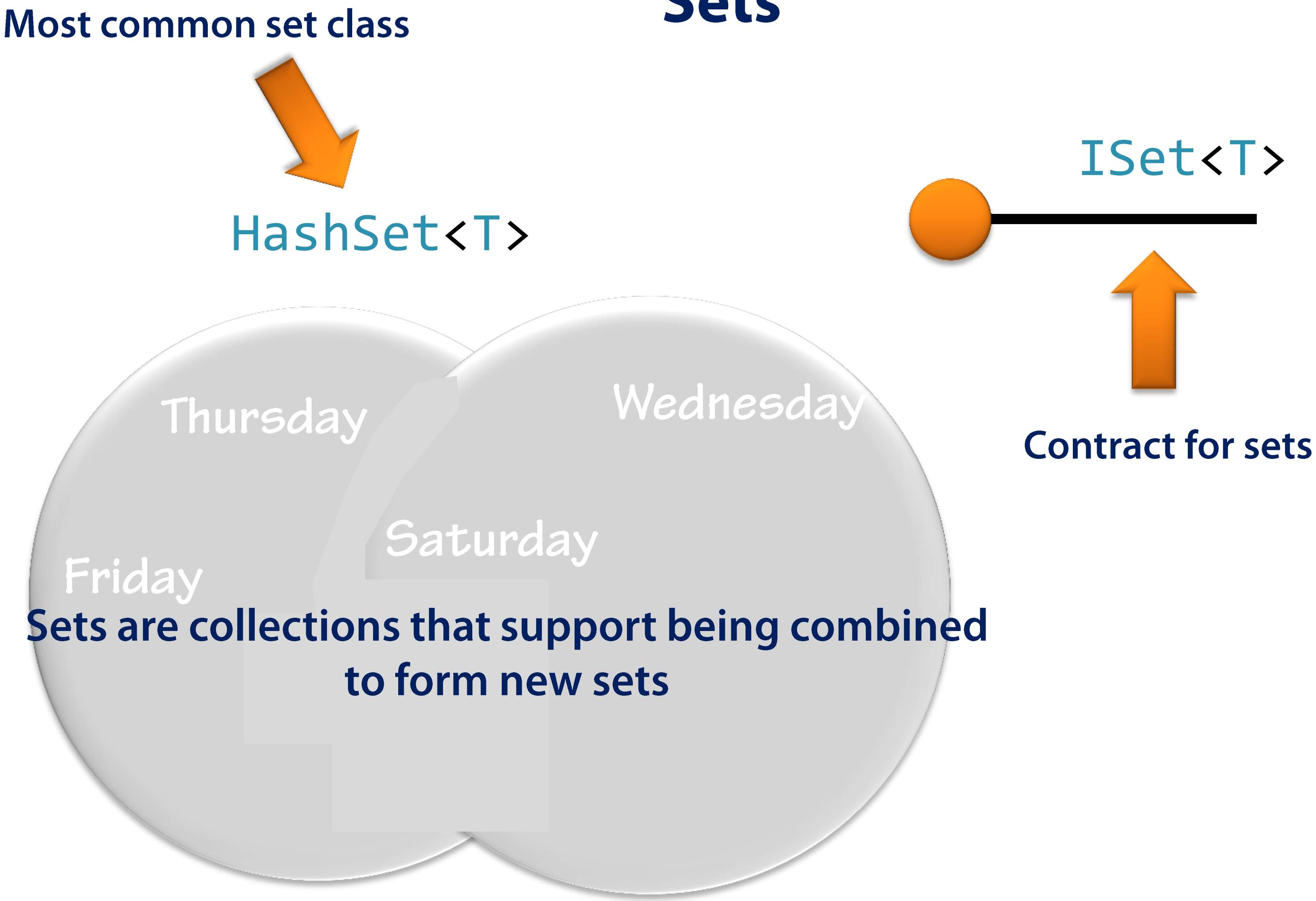
Sets

What days are in either collection?



Like SQL
`UNION DISTINCT`

Sets



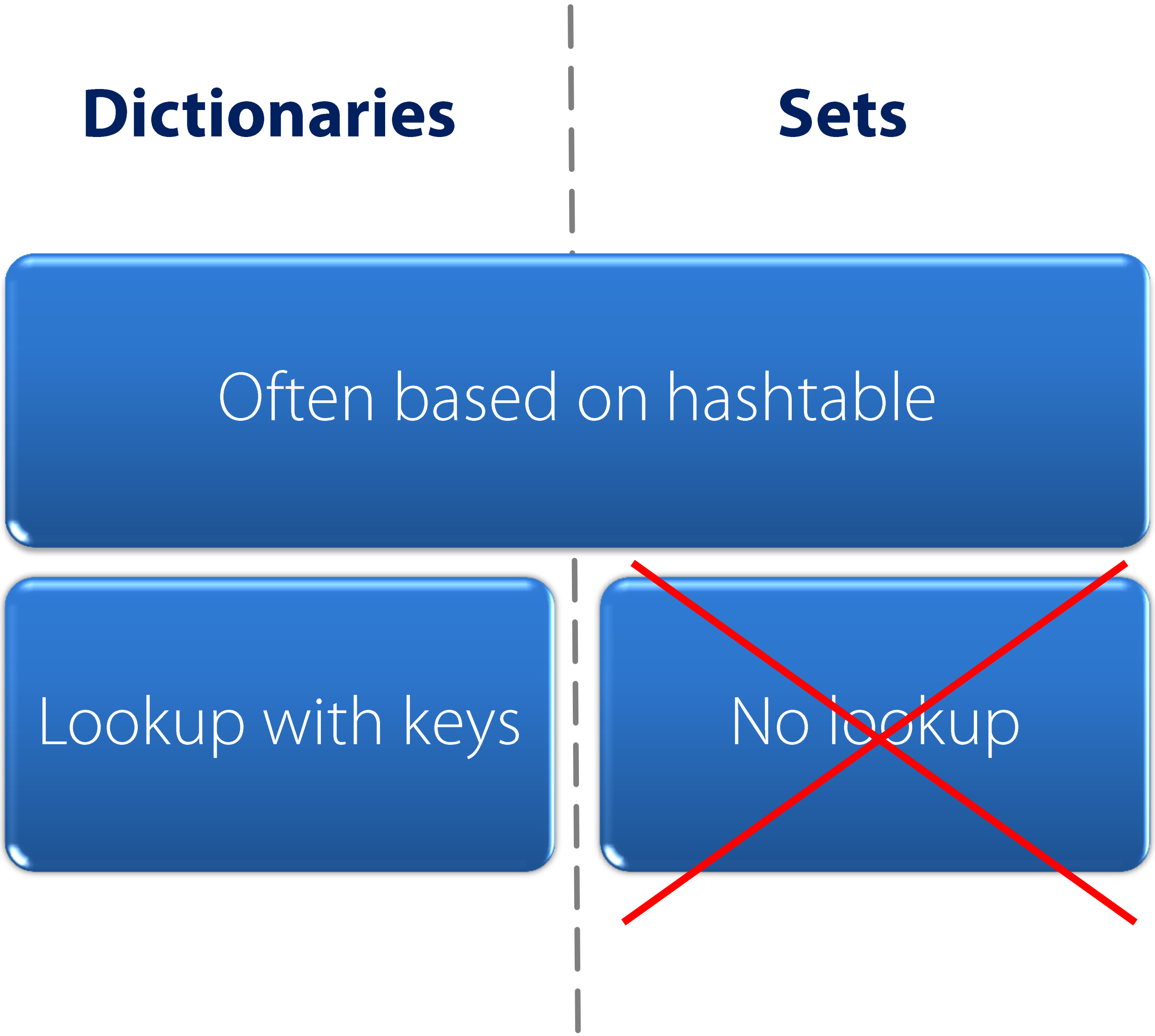
Most common set class

`HashSet<T>`

`ISet<T>`

Contract for sets

Sets are collections that support being combined to form new sets



Dictionaries

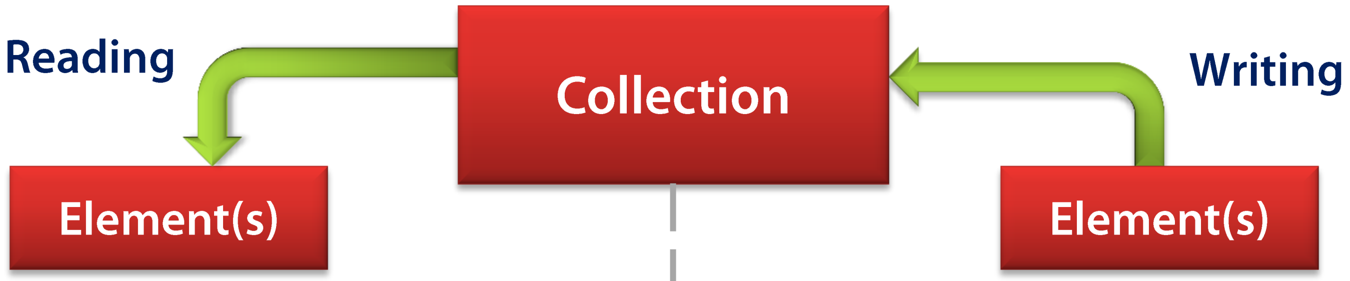
Sets

Often based on hashtable

Lookup with keys

~~No lookup~~

Collection Operations



Reading

Collection

Writing

Element(s)

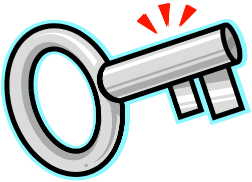
Element(s)

Look up an element

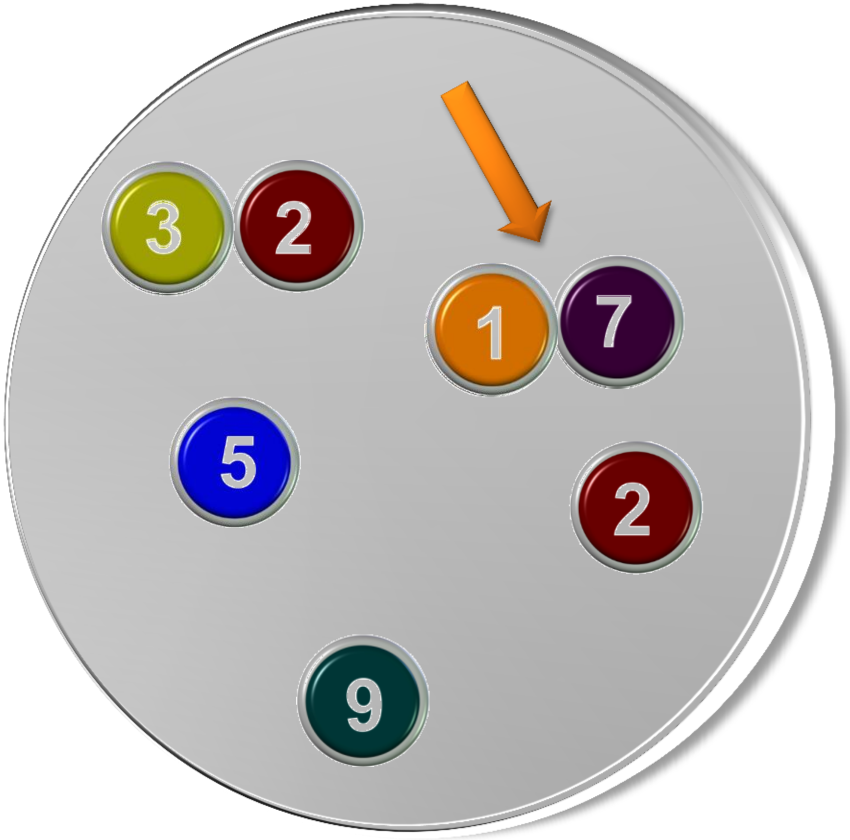
(by index or key)

Enumerate the elements

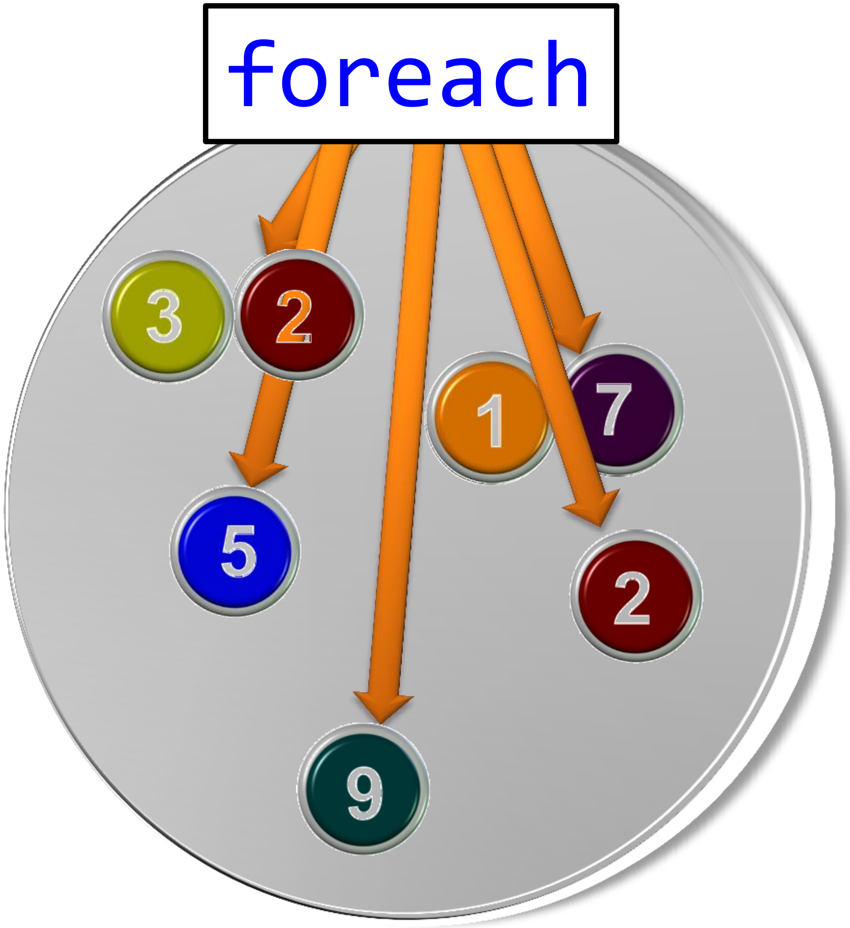
Looking up an item



(Key or index)



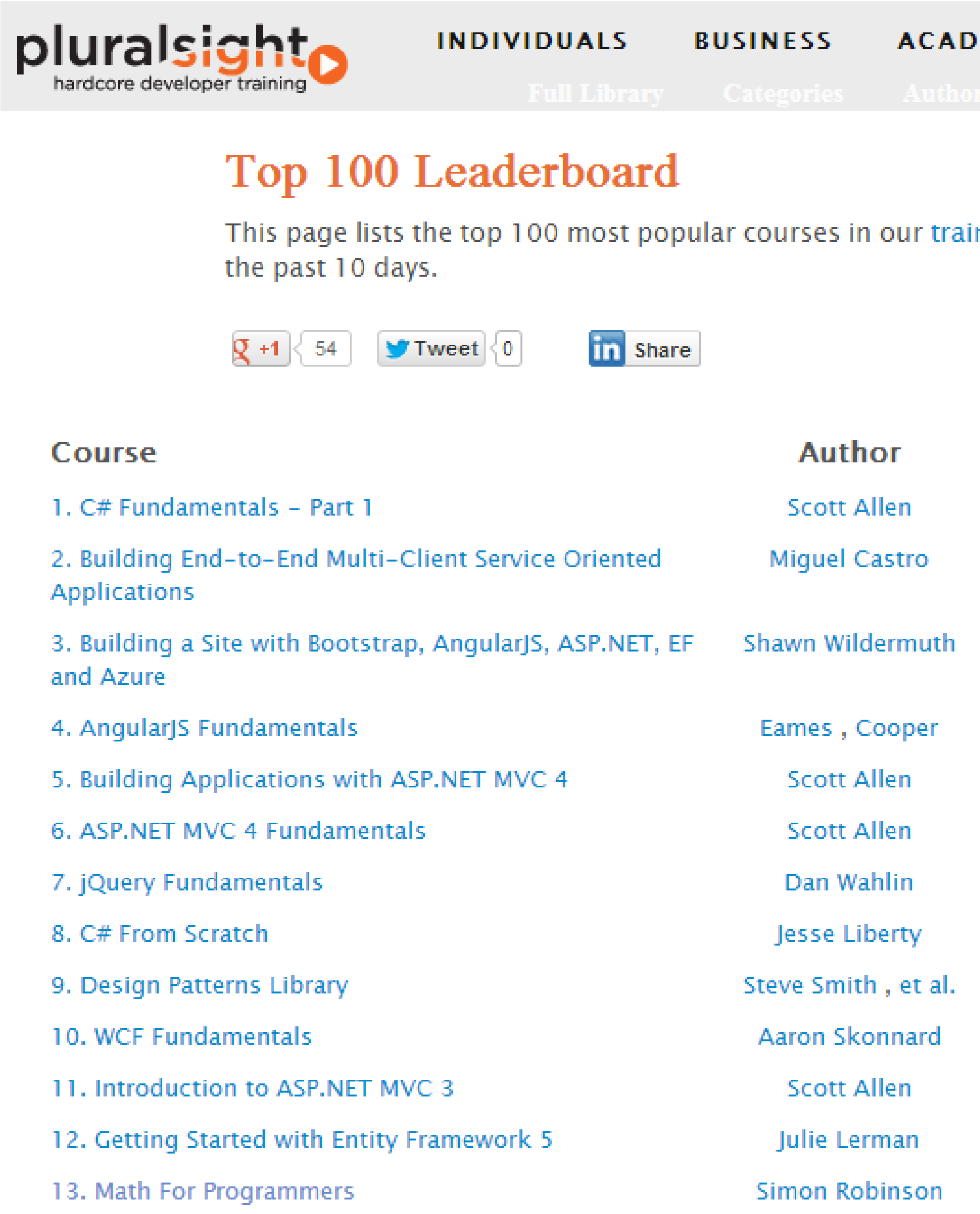
Enumerating a collection



Enumerating a collection

List

Items enumerated in index order



Top 100 Leaderboard

This page lists the top 100 most popular courses in our trai
the past 10 days.

Course	Author
1. C# Fundamentals - Part 1	Scott Allen
2. Building End-to-End Multi-Client Service Oriented Applications	Miguel Castro
3. Building a Site with Bootstrap, AngularJS, ASP.NET, EF and Azure	Shawn Wildermuth
4. AngularJS Fundamentals	Eames , Cooper
5. Building Applications with ASP.NET MVC 4	Scott Allen
6. ASP.NET MVC 4 Fundamentals	Scott Allen
7. jQuery Fundamentals	Dan Wahlin
8. C# From Scratch	Jesse Liberty
9. Design Patterns Library	Steve Smith , et al.
10. WCF Fundamentals	Aaron Skonnard
11. Introduction to ASP.NET MVC 3	Scott Allen
12. Getting Started with Entity Framework 5	Julie Lerman
13. Math For Programmers	Simon Robinson

Enumerating a collection

Dictionary

Don't rely on the order

Enumerate for eg. payroll processing…

Don't care about the order
– only that you do process all employees

Enumerating

All collections

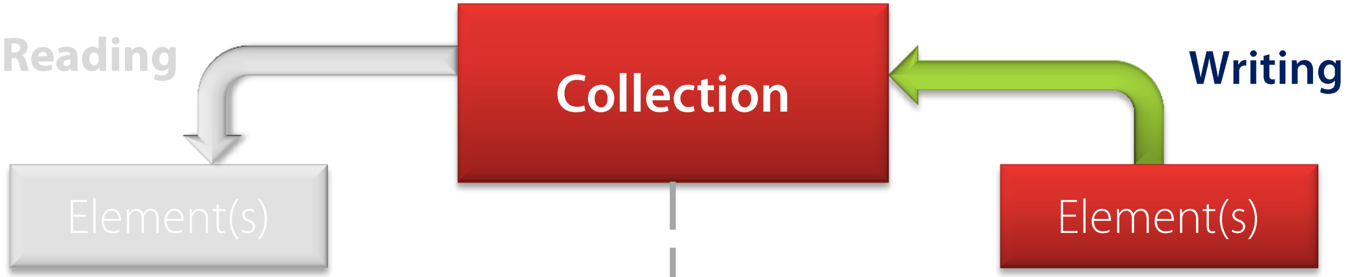
Looking up items

Many collections

NOT: Sets

NOT: Linked lists, Stacks, Queues

Collection Operations



Look up an element

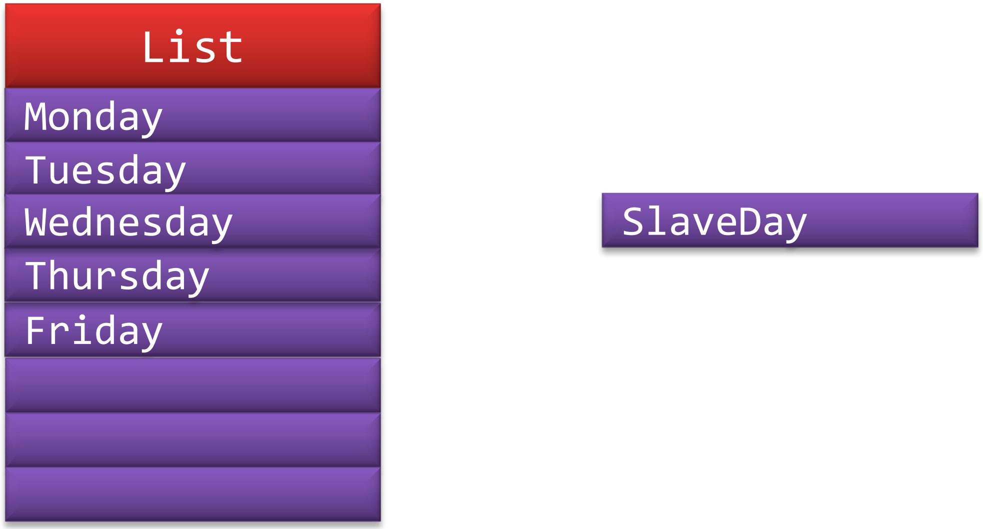
(by index or key)

Enumerate the elements

Add an element

Remove an element

Lists: Add item at a particular place



List

Monday

Tuesday

Wednesday

Thursday

Friday

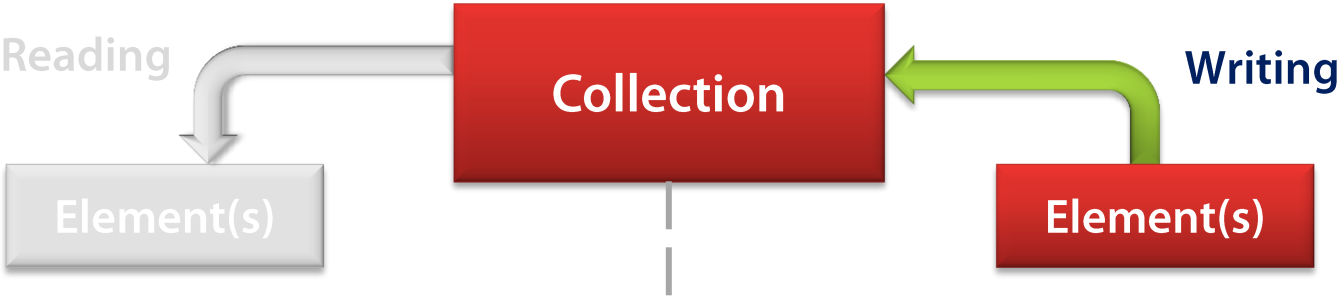
SlaveDay

This is called inserting

Add an item:

Don't care where it goes

Collection Operations



Reading

Element(s)

Collection

Writing

Element(s)

Look up an element

(by index or key)

Enumerate the elements

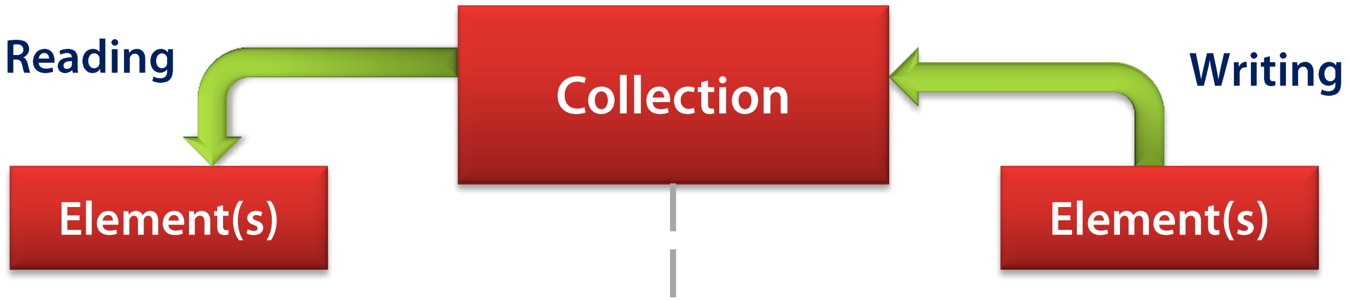
Add an element

Remove an element

LISTS: Insert an element

(Replace an element)

Collection Operations



Reading

Collection

Writing

Element(s)

Element(s)

Look up an element

(by index or key)

Enumerate the elements

Add an element

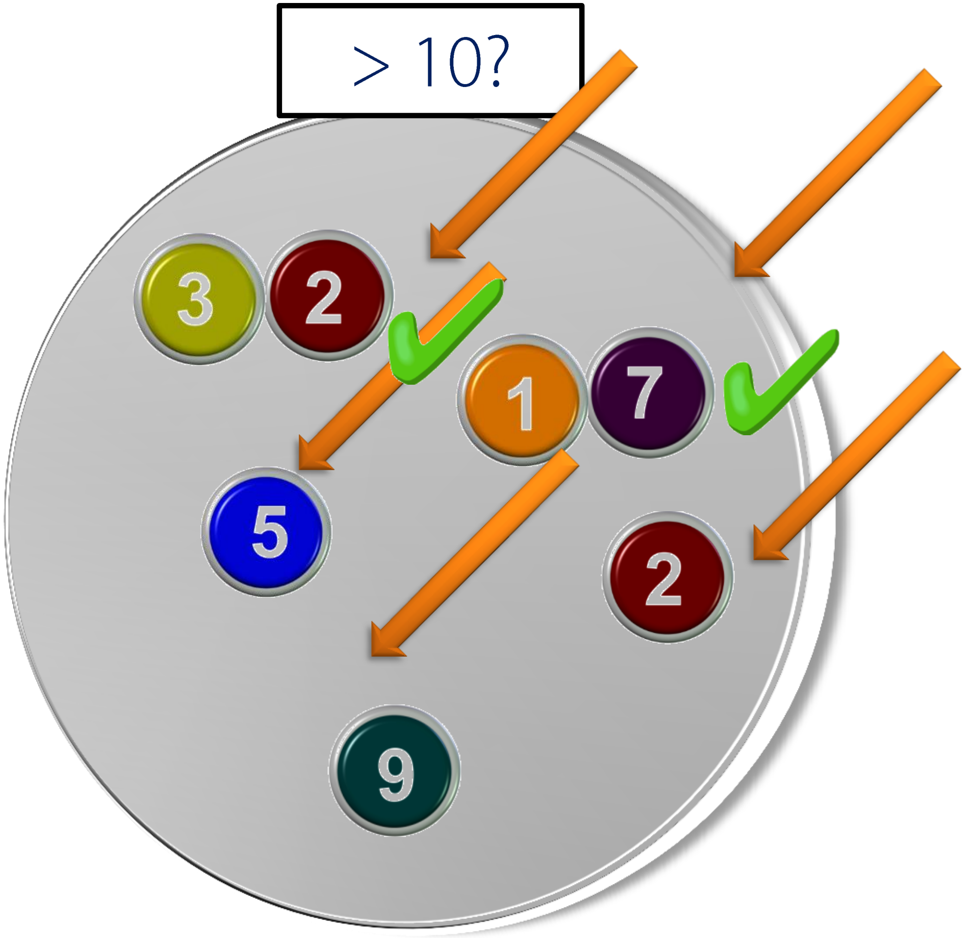
Remove an element

LISTS: Insert an element

(Replace an element)

Derived Operation:

Filter a list: Enumerate

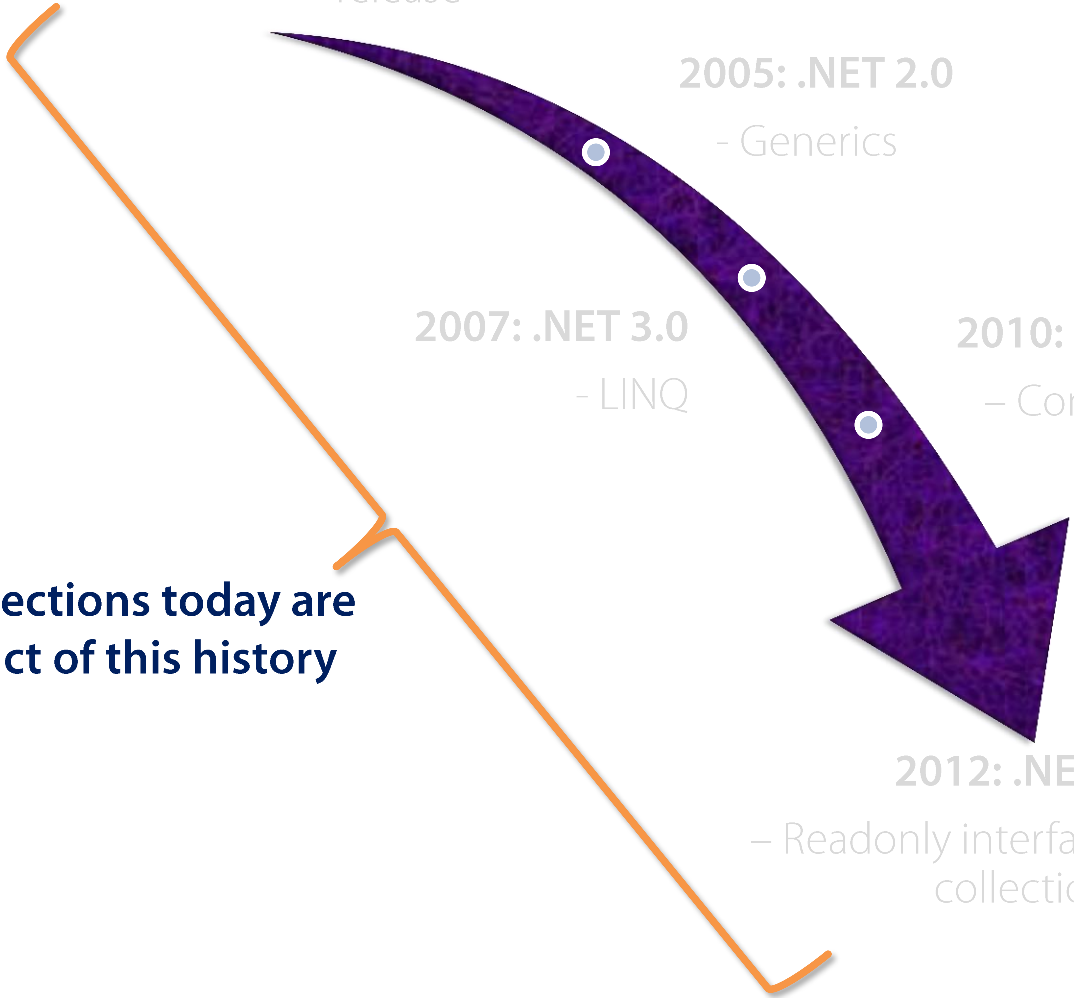


LINQ is great for these kinds of derived operations

But there's native support in

`T[]`   `List<T>`

Collections in .NET



2002: .NET 1.0 - First release

2005: .NET 2.0

- Generics

2007: .NET 3.0

- LINQ

2010: .NET 4.0

– Concurrent collections

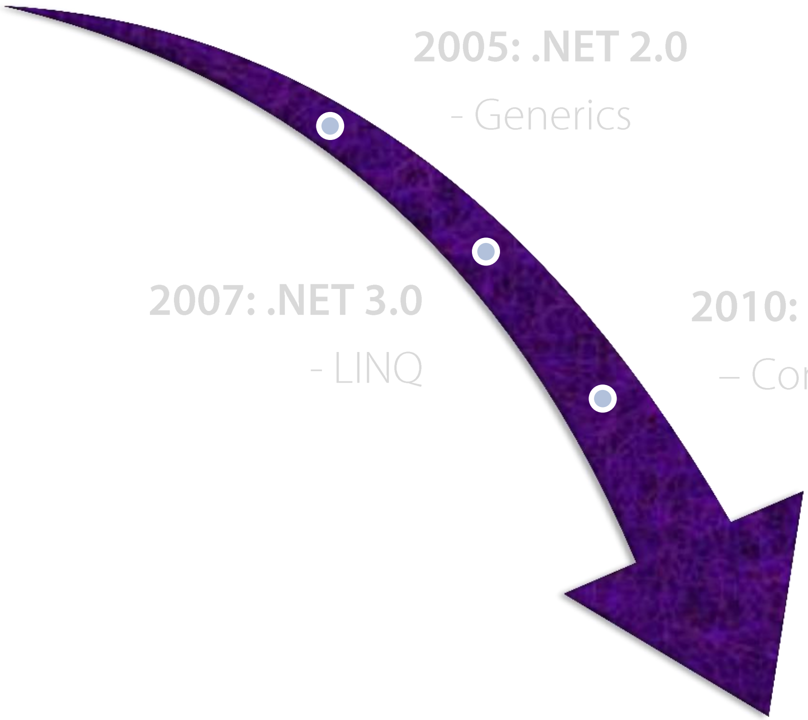
2012: .NET 4.5

– Readonly interfaces, Immutable collections

.NET collections today are a product of this history

Collections in .NET

2002: .NET 1.0 - First release



2005: .NET 2.0

- Generics

2007: .NET 3.0

- LINQ

2010: .NET 4.0

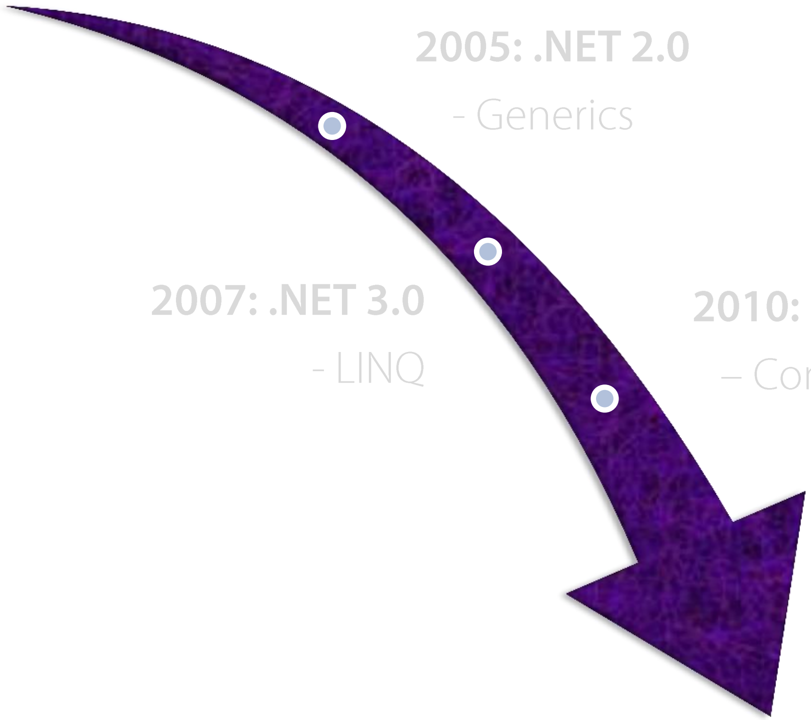
– Concurrent collections

2012: .NET 4.5

– Readonly interfaces, Immutable collections

Collections in .NET

2002: .NET 1.0 - First release



2005: .NET 2.0

- Generics

2007: .NET 3.0

- LINQ

2010: .NET 4.0

– Concurrent collections

2012: .NET 4.5

– Readonly interfaces, Immutable collections

Collections are Generic

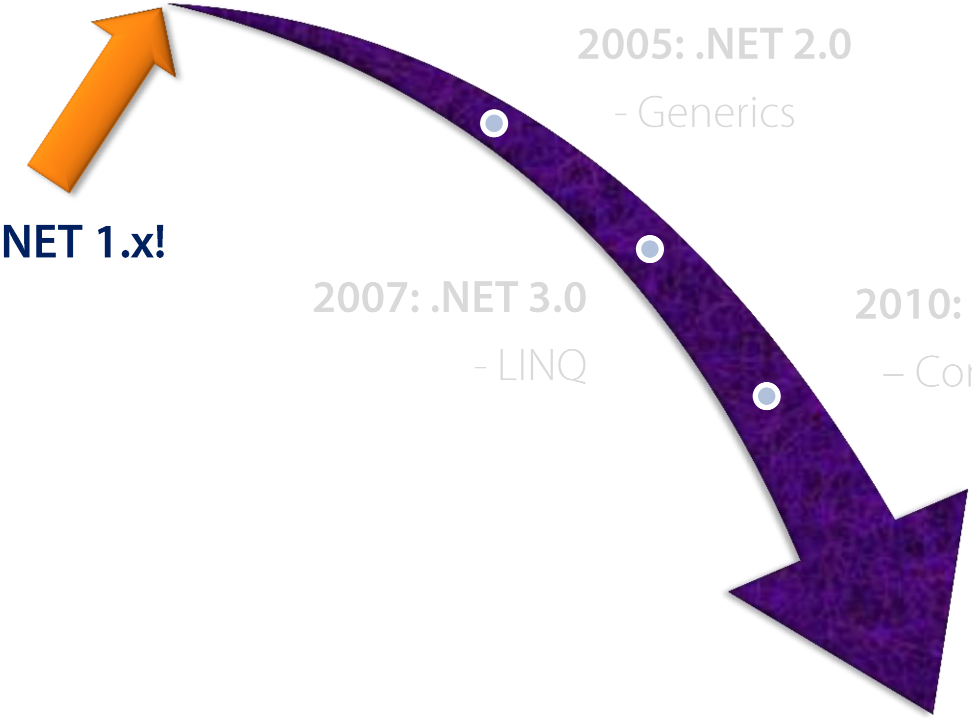
```
List<int>
```

```
List<Employee>
```

```
List<Control>
```

```
List<T>
```

What it's a list of



2002: .NET 1.0 - First release

No generics in .NET 1.x!

2005: .NET 2.0

- Generics

2007: .NET 3.0

- LINQ

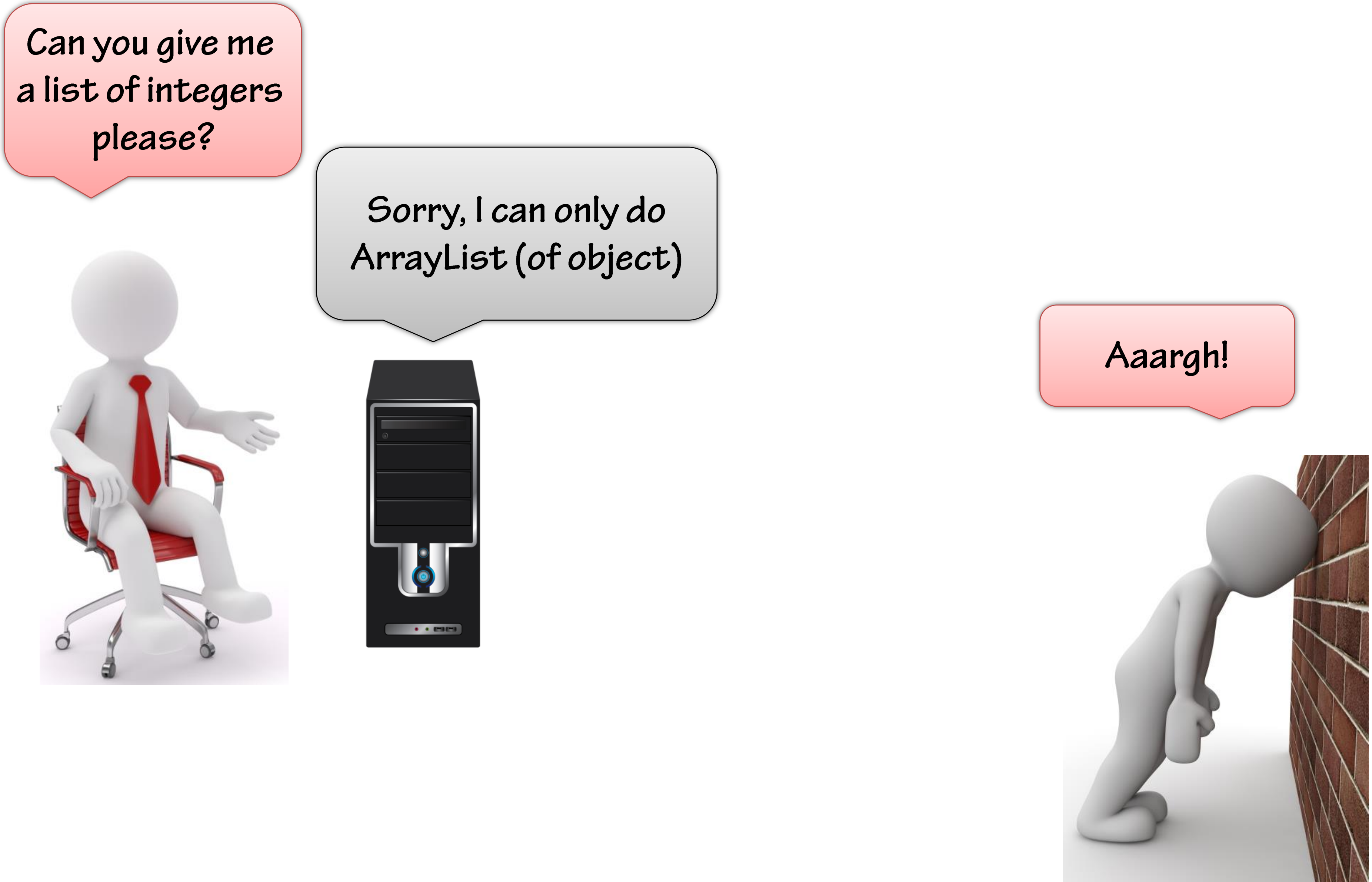
2010: .NET 4.0

– Concurrent collections

2012: .NET 4.5

– Readonly interfaces, Immutable collections

.NET 1.x


Can you give me a list of integers please?
Sorry, I can only do ArrayList (of object)
Aaargh!

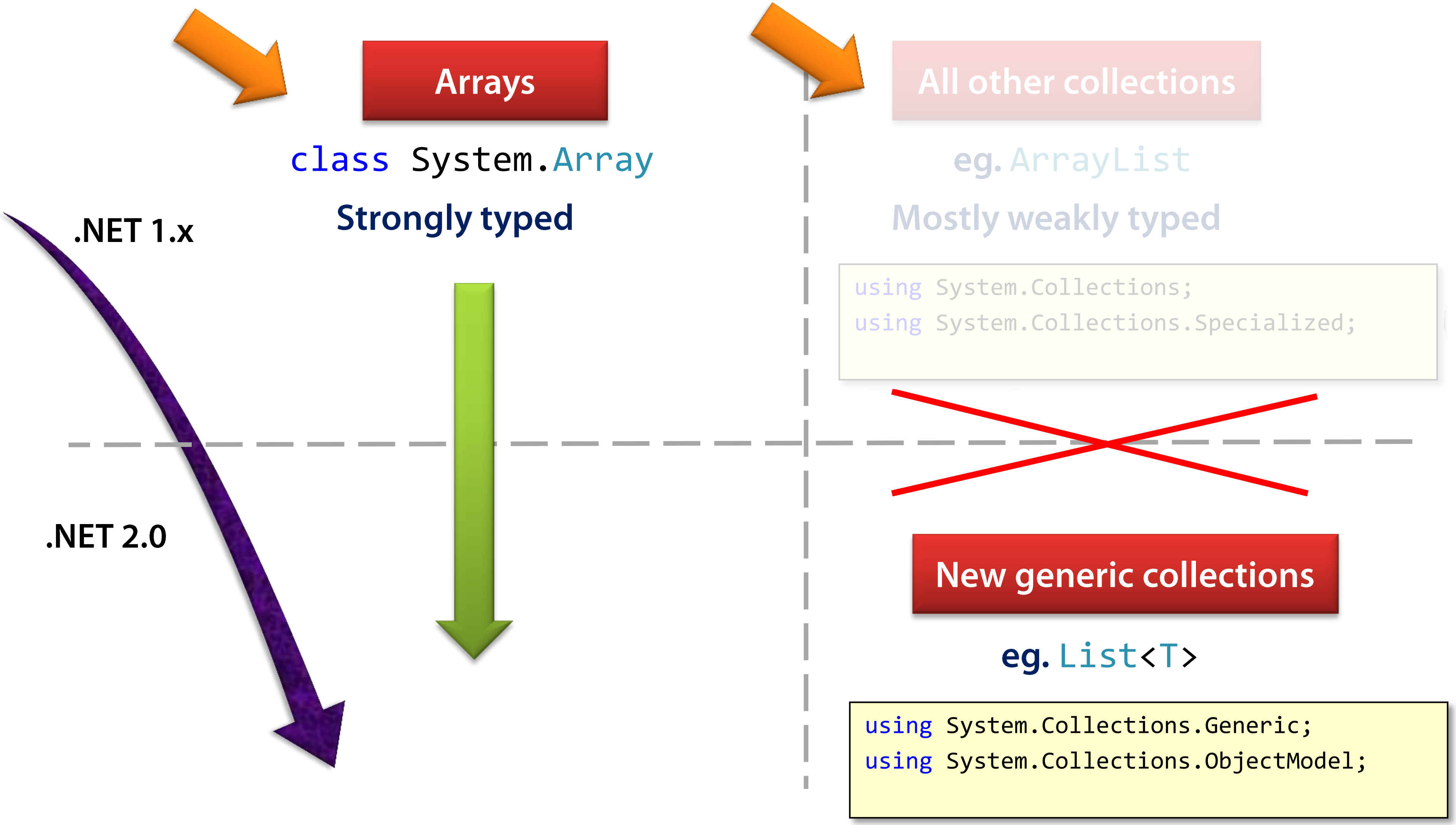
But arrays have always been strongly typed, eg.

```
int[] arrayOfInts = new int[];
```

Array of `int`

This code worked in
.NET 1.x
– only for arrays

This is NOT based on generics!



Arrays

`class System.Array`

Strongly typed

All other collections

eg. `ArrayList`

Mostly weakly typed

```
using System.Collections;
using System.Collections.Specialized;
```

.NET 1.x

.NET 2.0

New generic collections

eg. `List<T>`

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
```

Arrays

This the core material for this course

HashSet<T>

Stack<T>

Queue<T>

LinkedList<T>

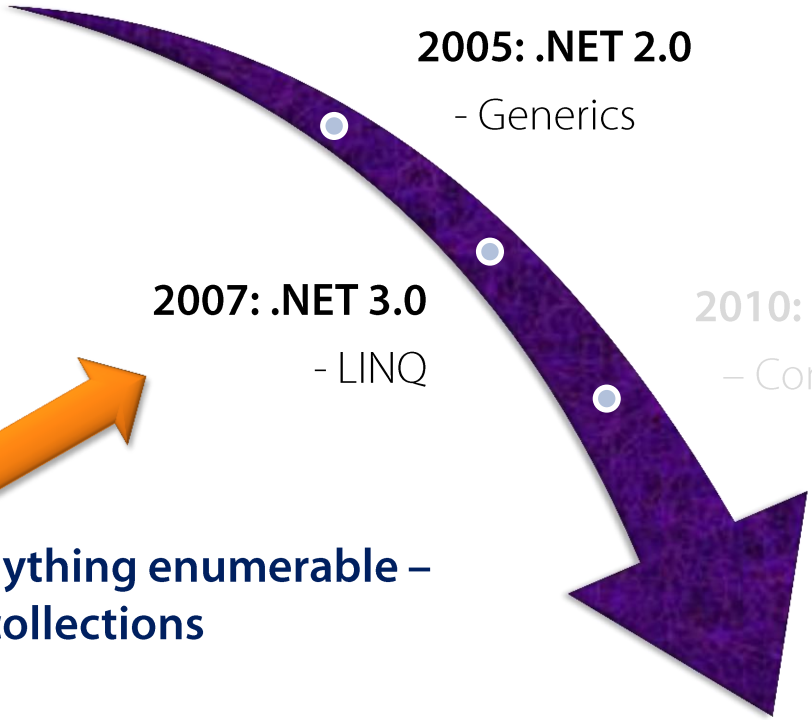
List<T>

Dictionary<TKey, TValue>

New generic collections

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
```

Collections in .NET



2002: .NET 1.0 - First release

2005: .NET 2.0

- Generics

2007: .NET 3.0

- LINQ

Adds operations to anything enumerable – including collections

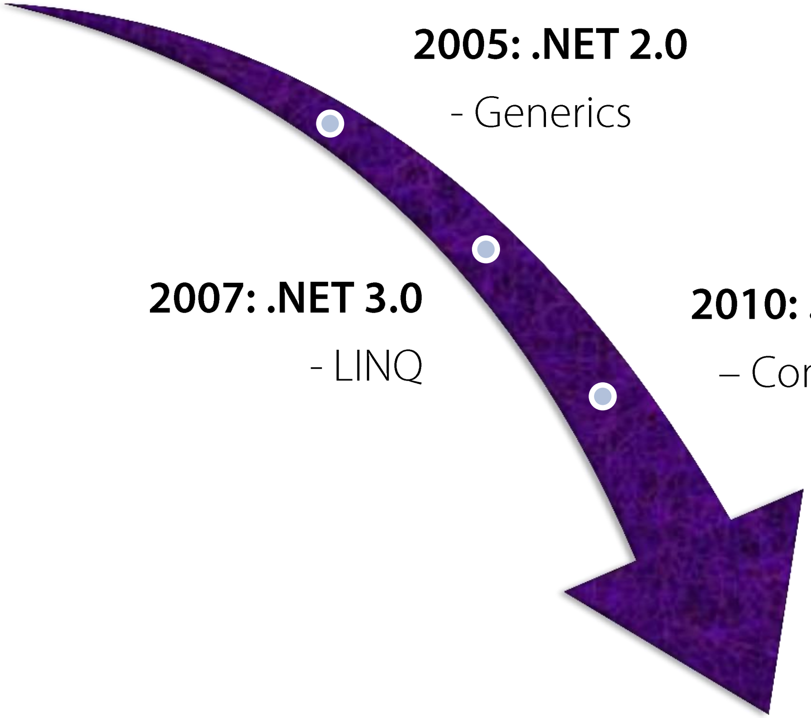
2010: .NET 4.0

– Concurrent collections

2012: .NET 4.5

– Readonly interfaces, Immutable collections

Collections in .NET



2002: .NET 1.0 - First release

2005: .NET 2.0

- Generics

2007: .NET 3.0

- LINQ

2010: .NET 4.0

– Concurrent collections

2012: .NET 4.5

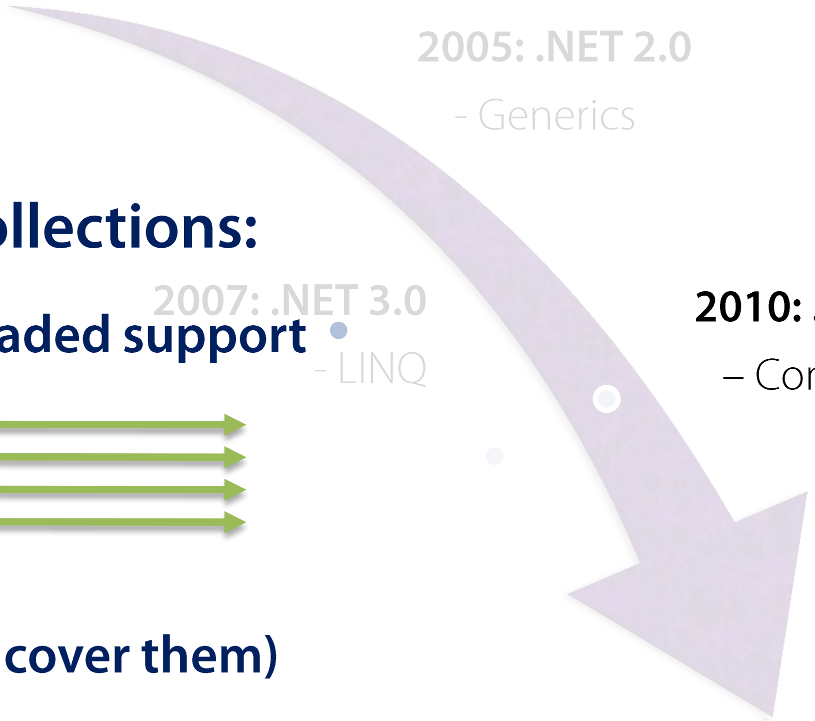
– Readonly interfaces, Immutable collections

Collections in .NET

Concurrent collections:

Multi-threaded support

(We won't cover them)



2002: .NET 1.0 - First release

2005: .NET 2.0

- Generics

2007: .NET 3.0

- LINQ

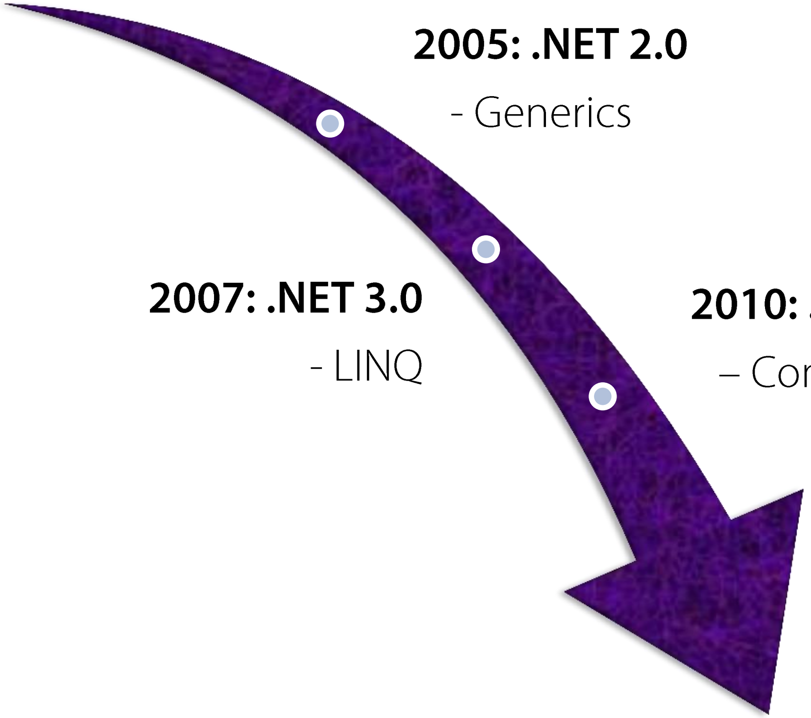
2010: .NET 4.0

– Concurrent collections

2012: .NET 4.5

– Readonly interfaces, Immutable collections

Collections in .NET



2002: .NET 1.0 - First release

2005: .NET 2.0

- Generics

2007: .NET 3.0

- LINQ

2010: .NET 4.0

– Concurrent collections

2012: .NET 4.5

– Readonly interfaces, Immutable collections

Readonly Interfaces

Eg.

`IReadOnlyList<T>`

`IReadOnlyCollection<T>`

Readonly contracts

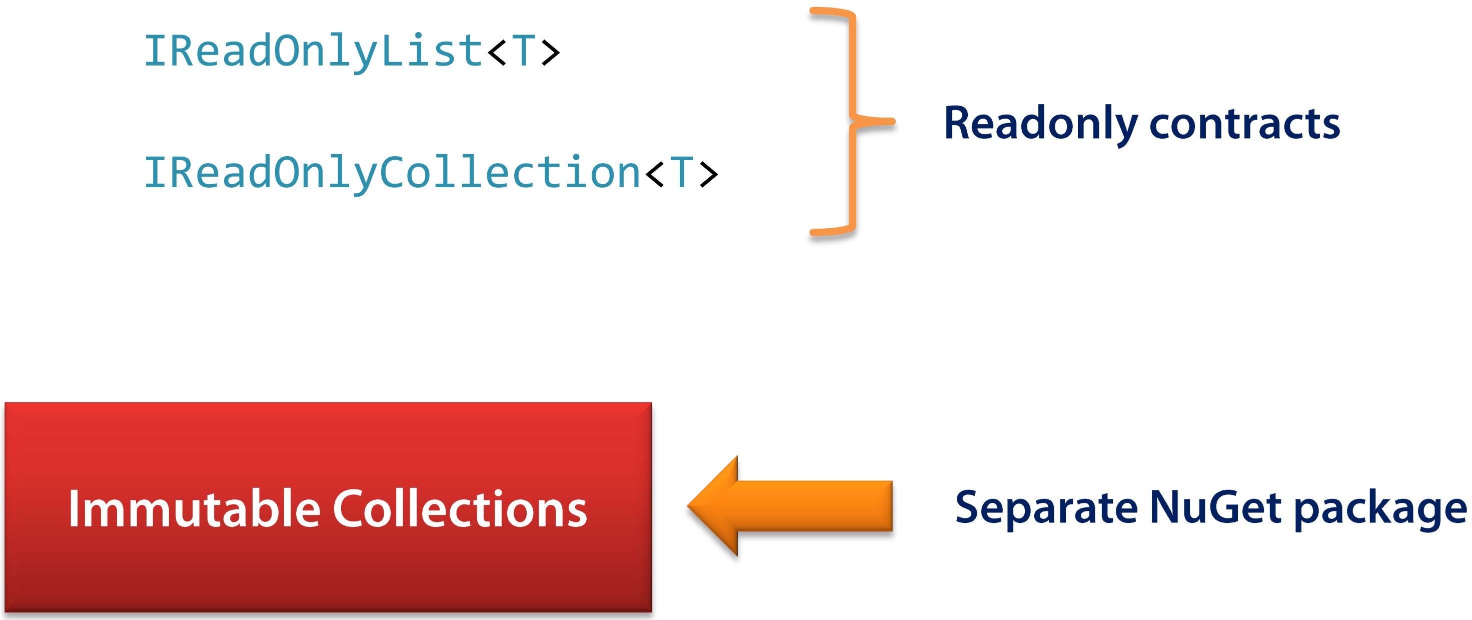
Previously, most collection interfaces had read-write contracts

`IList<T>`

`ICollection<T>`

Readonly Interfaces

Eg.



`IReadOnlyList<T>`

`IReadOnlyCollection<T>`

Readonly contracts

Immutable Collections

Separate NuGet package

C# Collections Today

Array

```
using System;
```

~~Old .NET 1.0 collections~~

```
~~using Syst          ns;~~
~~using Syst          ns.S          zed;~~
```

Obsolete

Core generic collections

```
using System.Coll          .Generic;
using System.          ons.ObjectModel;
```

Concurrent collections

(.NET 4   er only)

```
usi          tem.Collections.Concurrent;
```

Immutable collections

(.NET 4.5 only - via NUGET package)

```
using System.Collections.Immutable;
```

+ LINQ Extension methods

C# Collections Today

Array

```
using System;
```

~~Old .NET 1.0 collections~~

```
using System.Collections;
using System.Collections.Specialized;
```

Obsolete

Core generic collections

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
```

Concurrent collections

(.NET 4.0 and later only)

```
using System.Collections.Concurrent;
```

Immutable collections

(.NET 4.5 only - via NUGET package)

```
using System.Collections.Immutable;
```

+ LINQ Extension methods

Summary

- **Collections support:**
  - Adding/removing/looking up/enumerating items
- **Lists**
  - Look-up using index
  - (Some don't allow look-up)
- **Dictionaries**
  - Look-up using keys
- **Sets**
  - For operations on collections as units
- **.NET Collections landscape**
  - Core generic
  - Concurrent
  - Immutable