

Introduction to Collections

Simon Robinson
<http://TechieSimon.com>
@TechieSimon



pluralsight 
hardcore developer training

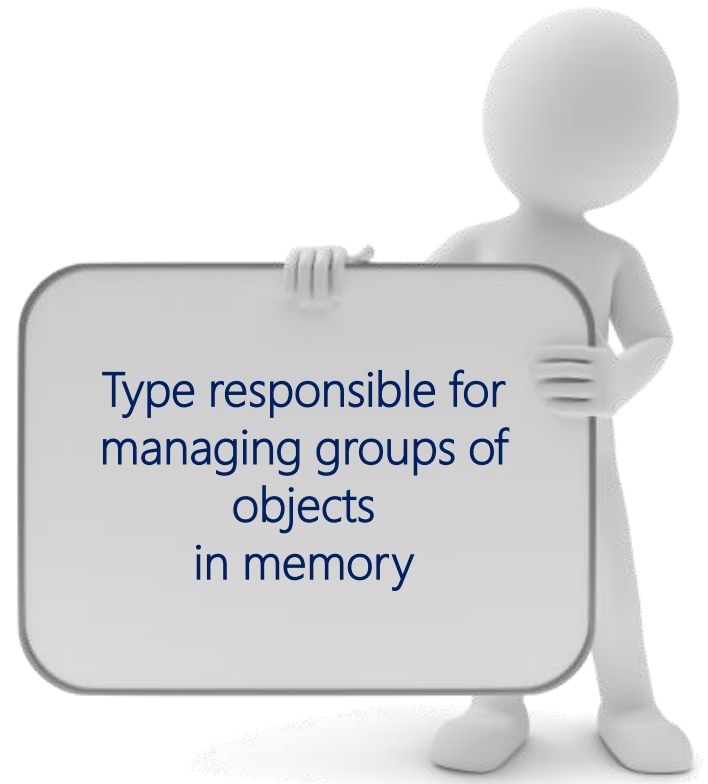
Overview

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

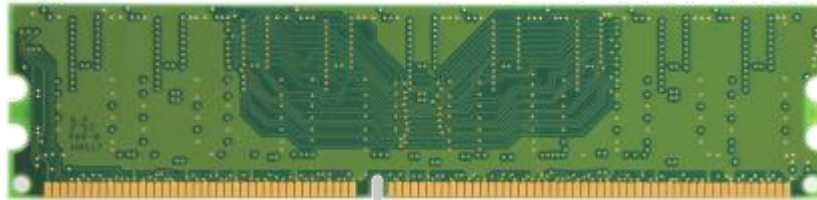
High level
– won't look at individual classes (yet)



What is a Collection?



In memory!

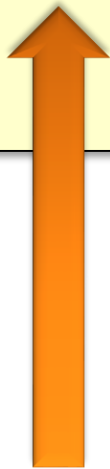


An integer!



A collection
(of integers)

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

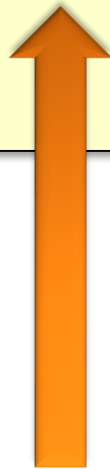


Not necessarily a collection



Yields employee instances
one at a time

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



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



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

Many collection classes...

`ReadOnlyCollection<T>`
`SortedList<TKey, TValue>` `LinkedList<T>`
`HashSet<T>` `Stack<T>` `ObservableCollection<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 10 days.



Course	Author	Level	Rating	Duration	Released
1. C# Fundamentals				2h 17m	26 Mar 2019
2. Building iOS Applications				1h 40m	26 Aug 2019
3. Building a Site with Bootstrap, AngularJS, ASP.NET, SQL and Azure	Steve Withermuth	Intermediate	★★★★★	2h 28m	17 Jul
4. AngularJS Fundamentals	James Cooper	Intermediate	★★★★★	2h 14m	17 Mar 2019
5. Building Applications with ASP.NET MVC 5	Scott Allen	Intermediate	★★★★		
6. ASP.NET MVC 5 Fundamentals	Scott Allen	Intermediate	★★★★		
7. jQuery Fundamentals	Don Mahler	Beginner	★★★★		
8. C# from Scratch	James Liberty	Beginner	★★★★		
9. Design Patterns: Library	Steve Smith, et al	Intermediate	★★★★		
10. MVC Fundamentals	James Steward	Beginner	★★★★		
11. Introduction to ASP.NET MVC 5	Scott Allen	Beginner	★★★★		
12. Getting Started with Unity Framework 5	John Larrigan	Intermediate	★★★★★	2h 20m	
13. Math for Programmers	Simon St Laurent	Intermediate	★★★★★		

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.



54



0



Share

Index: The position in order

Course

Author

Level

Rating

Duration

Released

1. C# Fundamentals – Part 1

Index = 0

Beginner

★★★★★

[06:17:48]

26 Mar 2010

2. Building End-to-End Multi-Client Service Oriented Applications

Miguel Castro

Intermediate

★★★★★

[11:43:23]

30 Aug 2010

3. Building a Site with Bootstrap, AngularJS, ASP.NET, EF

Shawn Wildermuth

Intermediate

★★★★★

[06:29:37]

31 Jul 2010

4. JavaScript Fundamentals

Index = 1

Intermediate

★★★★★

[06:14:53]

17 May 2010

5. Building Applications with ASP.NET MVC 4

Scott Allen

Intermediate

★★★★★

6. ASP.NET MVC 4 Fundamentals

Scott Allen

Intermediate

★★★★★

7. Building Applications with ASP.NET MVC 4

Dan Wahlin

Beginner

★★★★★

8. Building Applications with ASP.NET MVC 4

Jesse Liberty

Beginner

★★★★★

9. Building Applications with ASP.NET MVC 4

Steve Smith, et al.

Intermediate

★★★★★

10. Building Applications with ASP.NET MVC 4

Aaron Skonnard

Beginner

★★★★★

11. Building Applications with ASP.NET MVC 4

Scott Allen

Beginner

★★★★★

12. Building Applications with ASP.NET MVC 4

Julie Lerman

Intermediate

★★★★★

13. Building Applications with ASP.NET MVC 4

Simon Robinson

Intermediate

★★★★★

Zero-based indexing

This is a list

List Types in .NET:

`T[]`

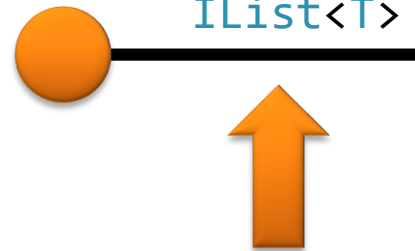
`List<T>`

`Collection<T>`

`ReadOnlyCollection<T>`

`ObservableCollection<T>`

`ICollection<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>

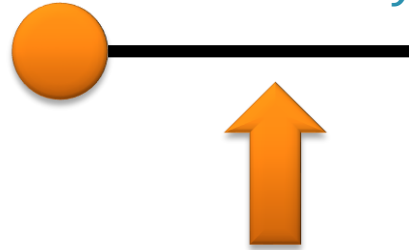


Most widely used dictionary type



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:

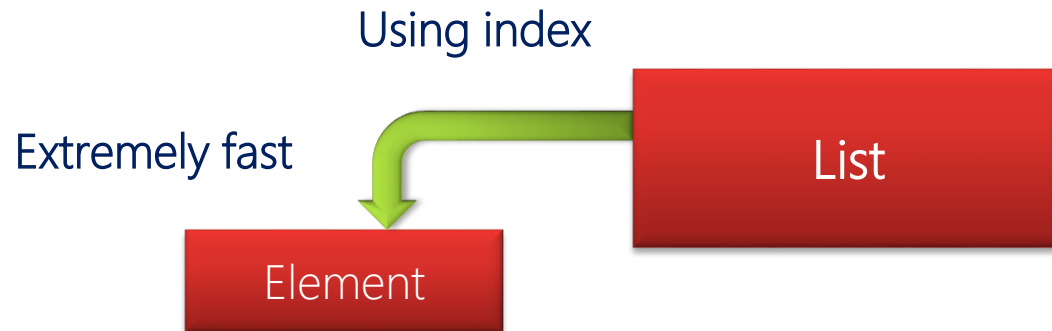
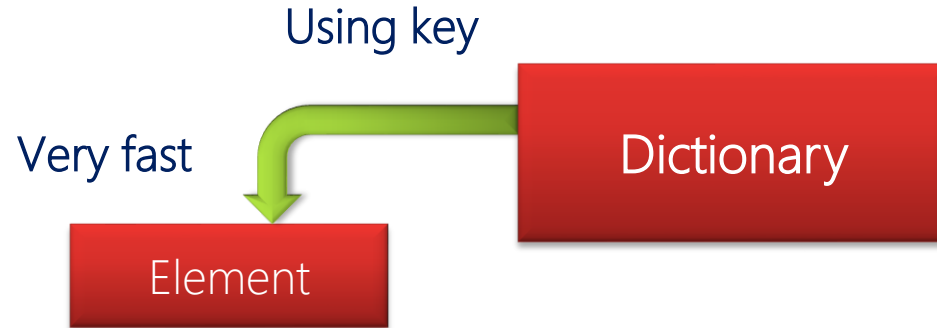
```
var employees =  
    new Dictionary<string, Employee>();
```



For looking up by social security no., declare:

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

Looking up an element



Dictionaries



Very useful



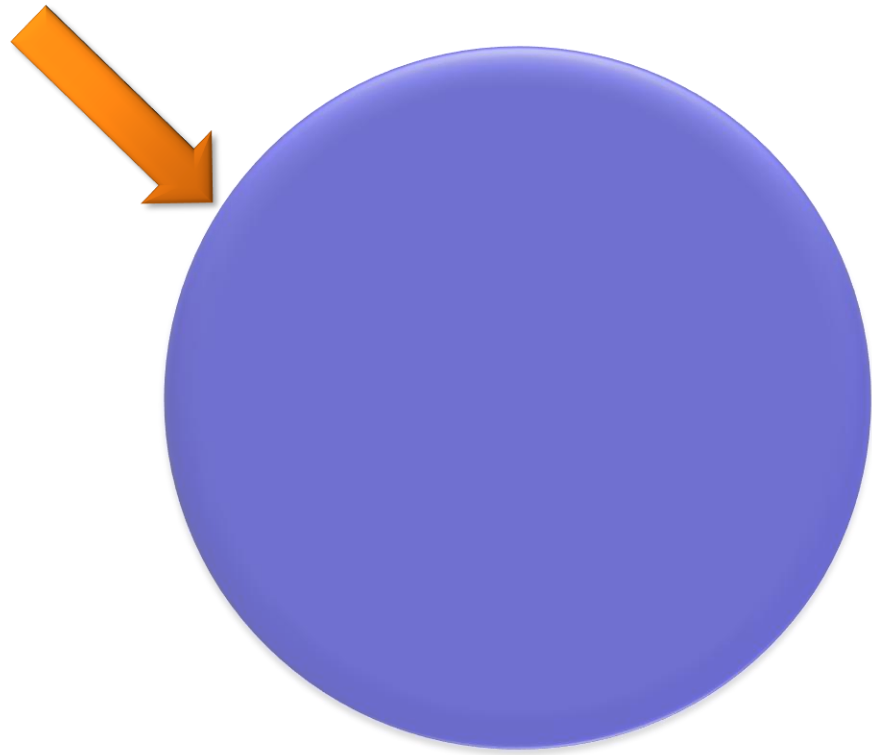
Easy to misuse

Expect keys to behave a certain way
(Cover later in this course)



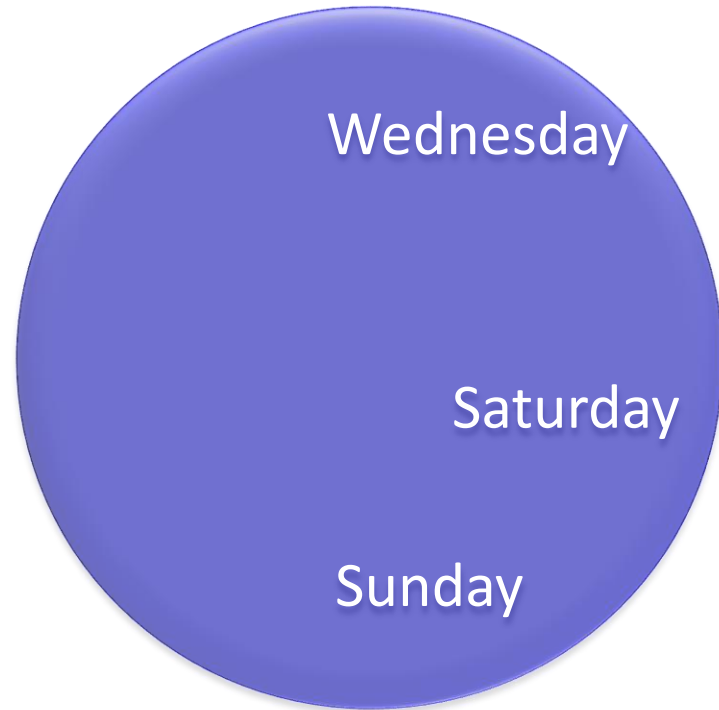
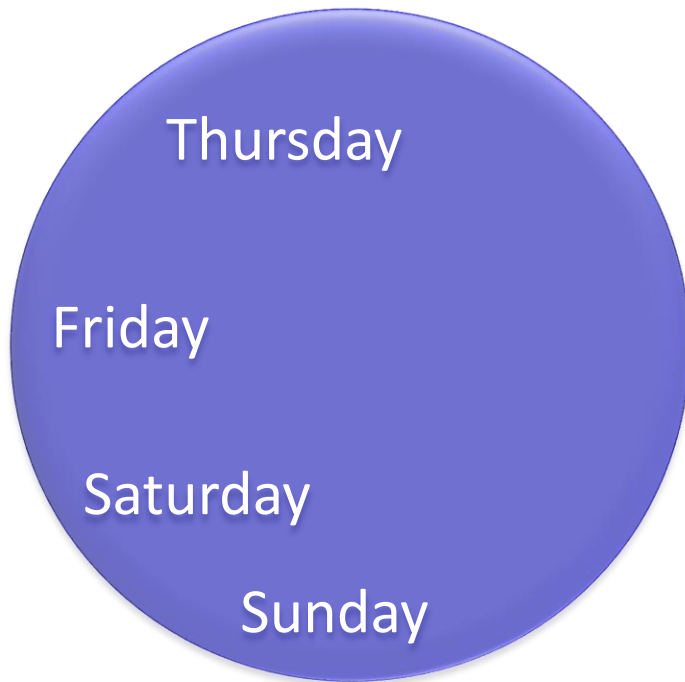
Sets

Sets treat the collection as a whole



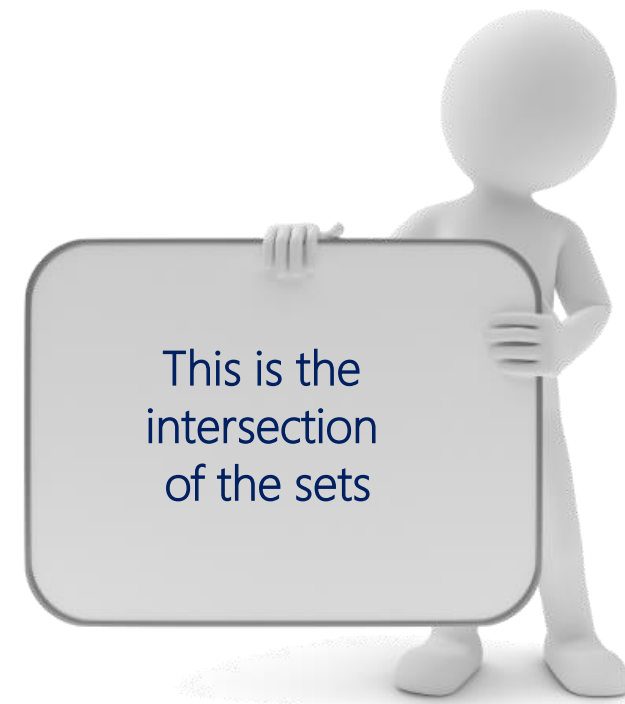
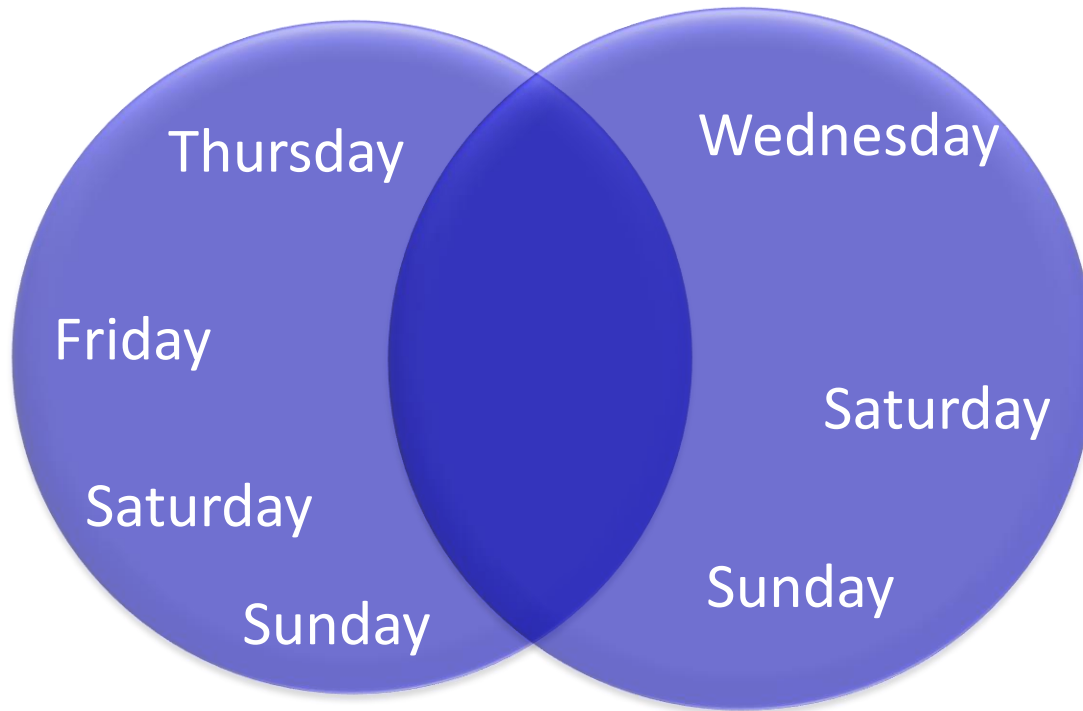
Sets

What days are in both collections?



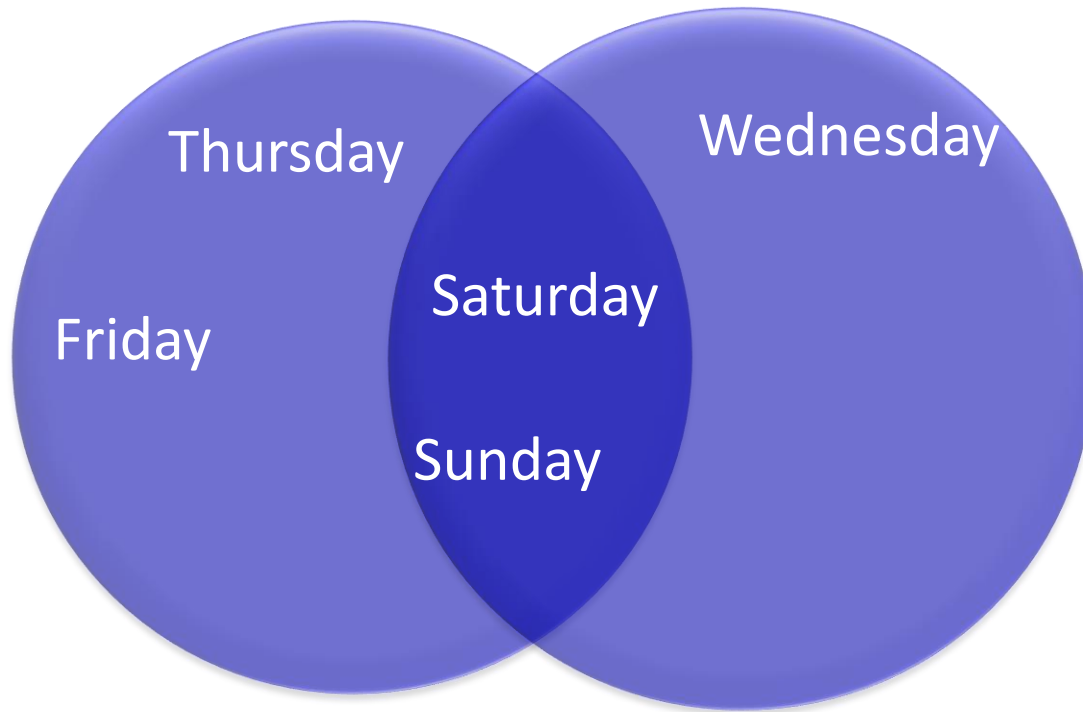
Sets

What days are in both collections?



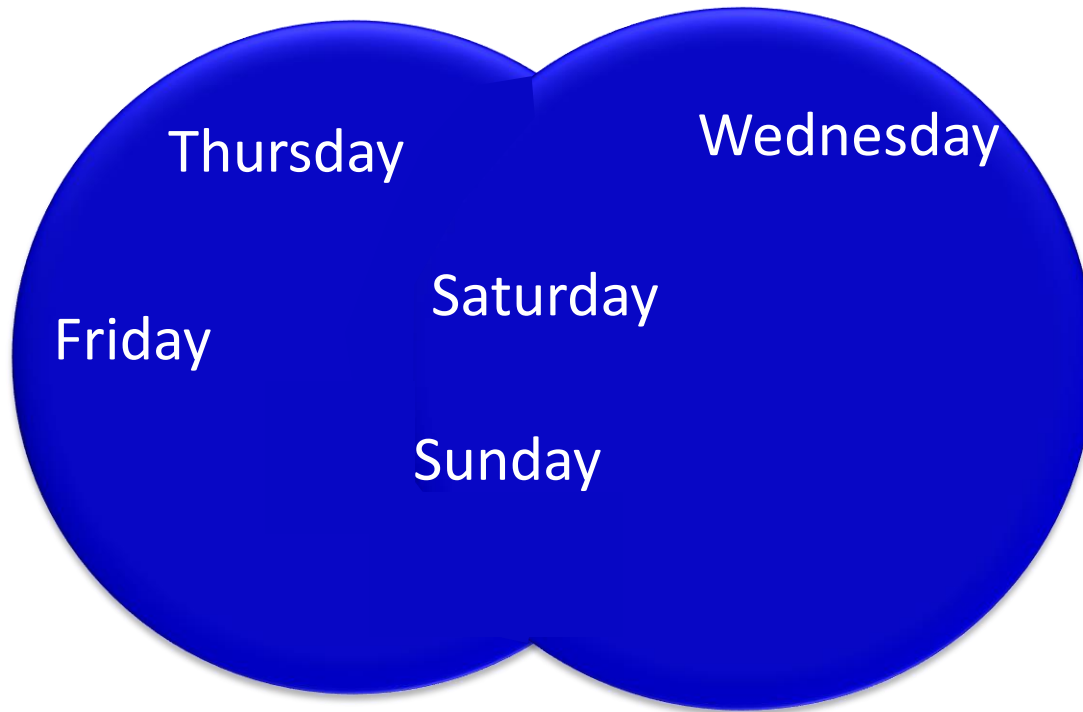
Sets

What days are in either collection?



Sets

What days are in either collection?



Like SQL
UNION DISTINCT

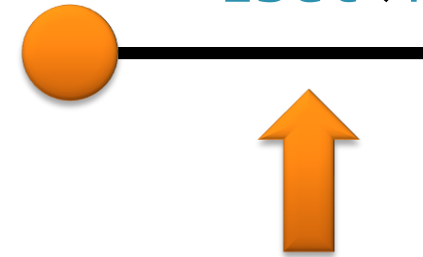


Sets

Most common set class


HashSet<T>

ISet<T>



Contract for sets



Dictionary

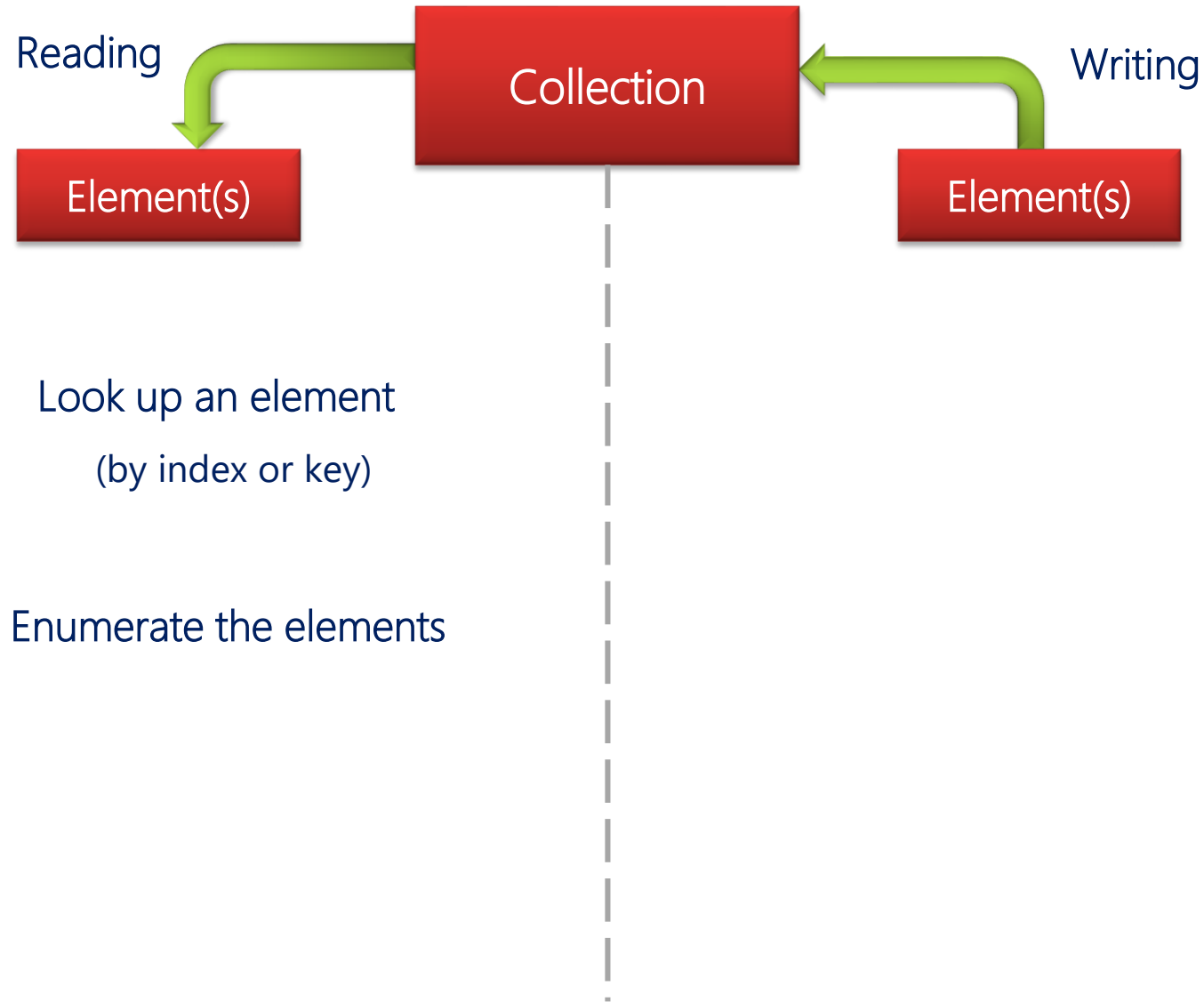
Sets

Often based on hashtable

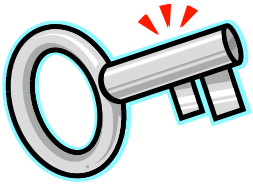
Lookup with
keys

~~No lookup~~

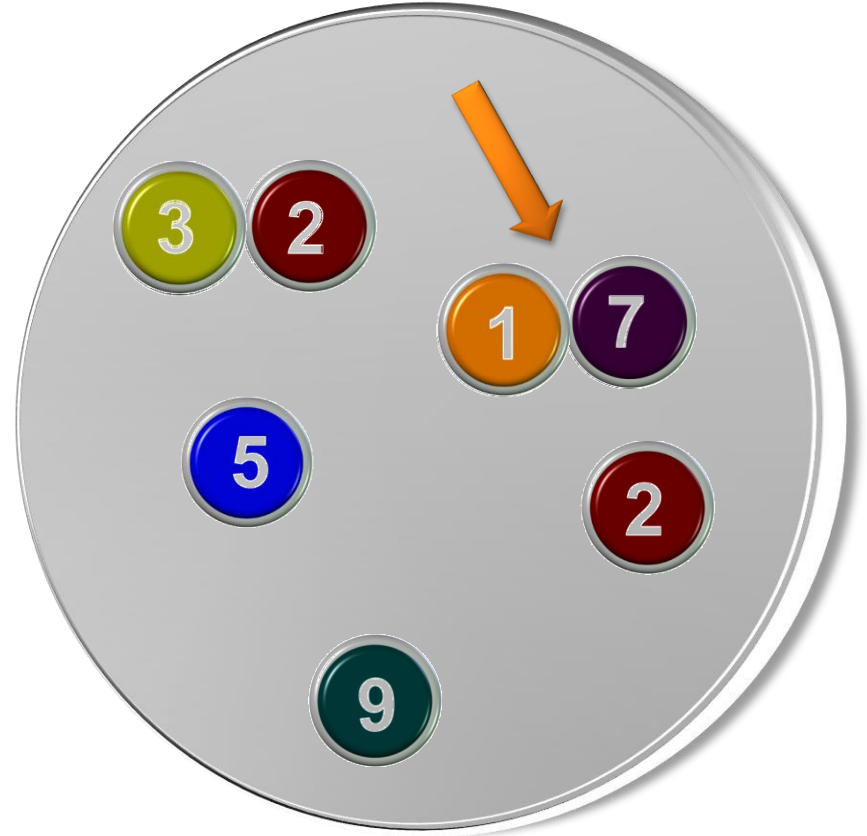
Collection Operations



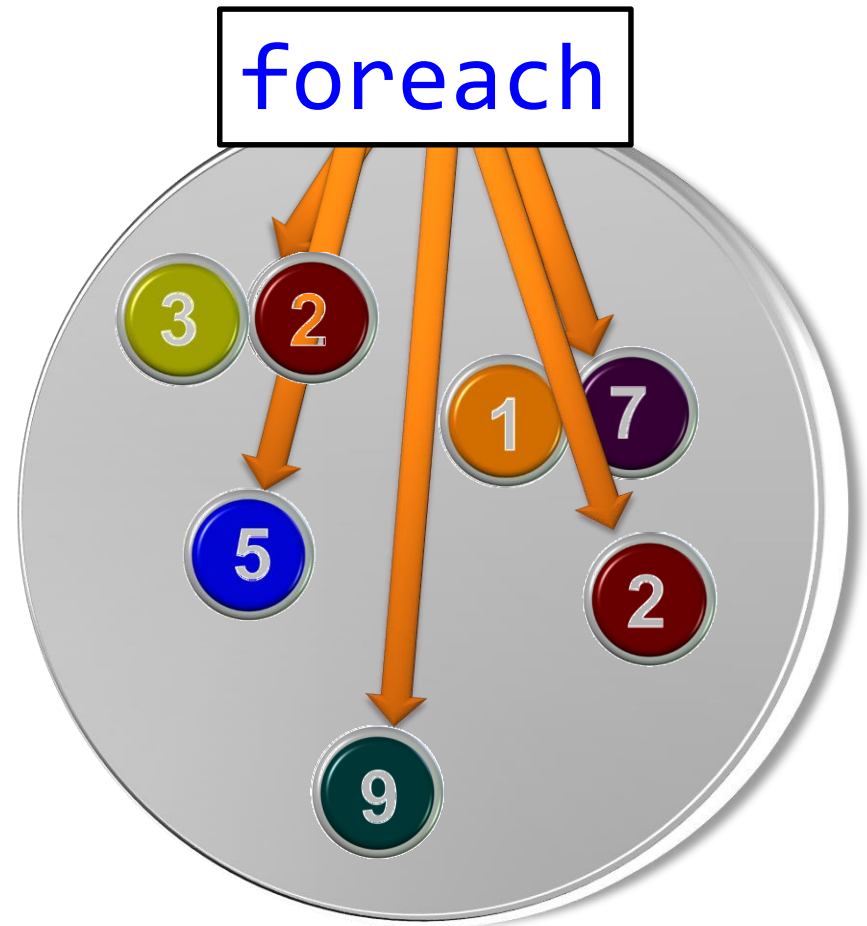
Looking up an item



(Key or index)



Enumerating a collection



Enumerating a collection

List

Items enumerated
in index order



INDIVIDUALS

BUSINESS

ACAD

Full Library

Categories

Author

Top 100 Leaderboard

This page lists the top 100 most popular courses in our [training library](#) over 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

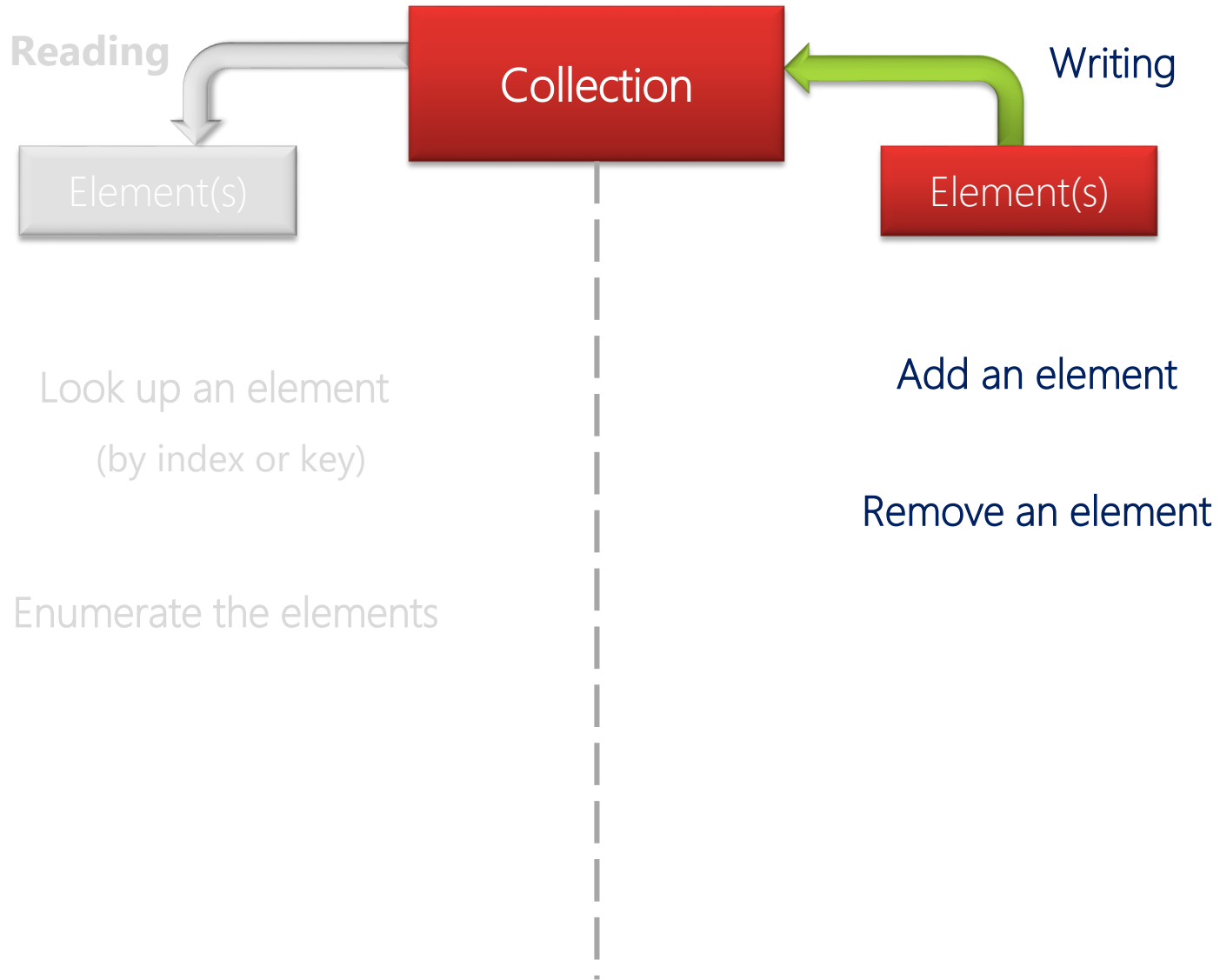
Looking up items

Many
collections

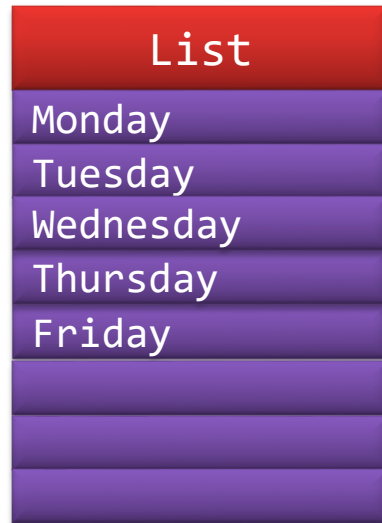
NOT: Sets

NOT: Linked lists,
Stacks, Queues

Collection Operations



Lists: Add item at a particular place



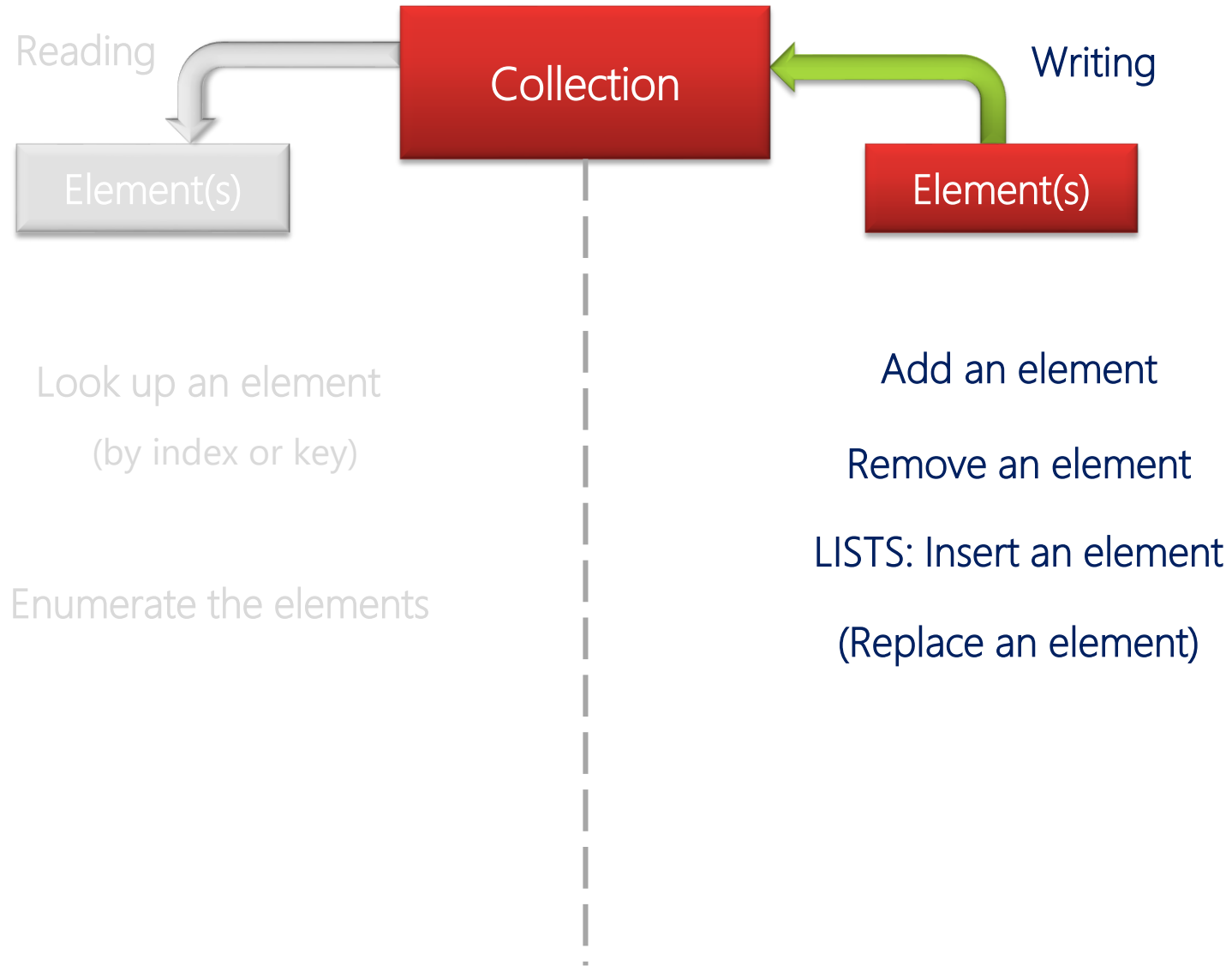
This is called inserting

Add an item:

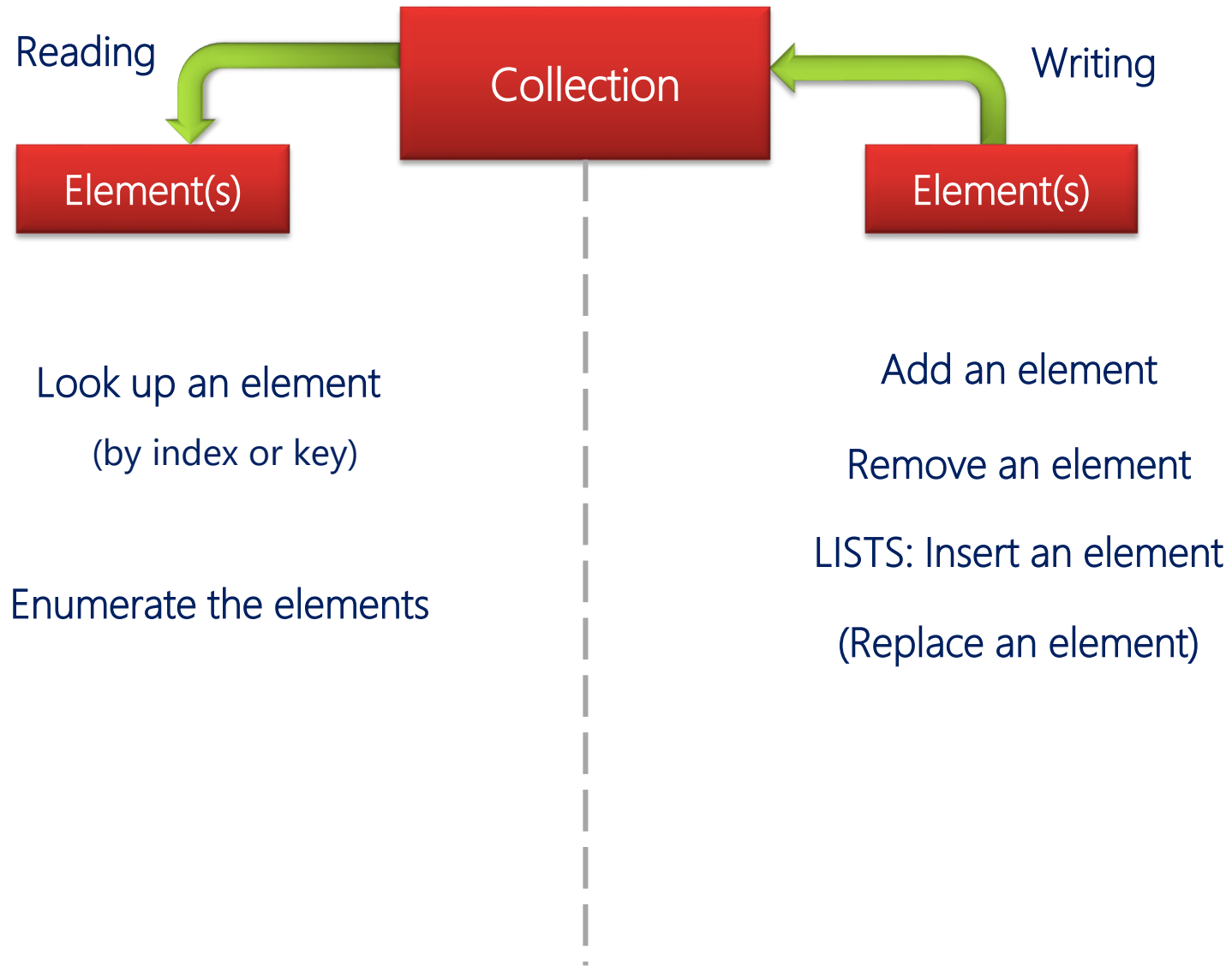
Don't care where it goes

A single purple button labeled "MagicDay".

Collection Operations

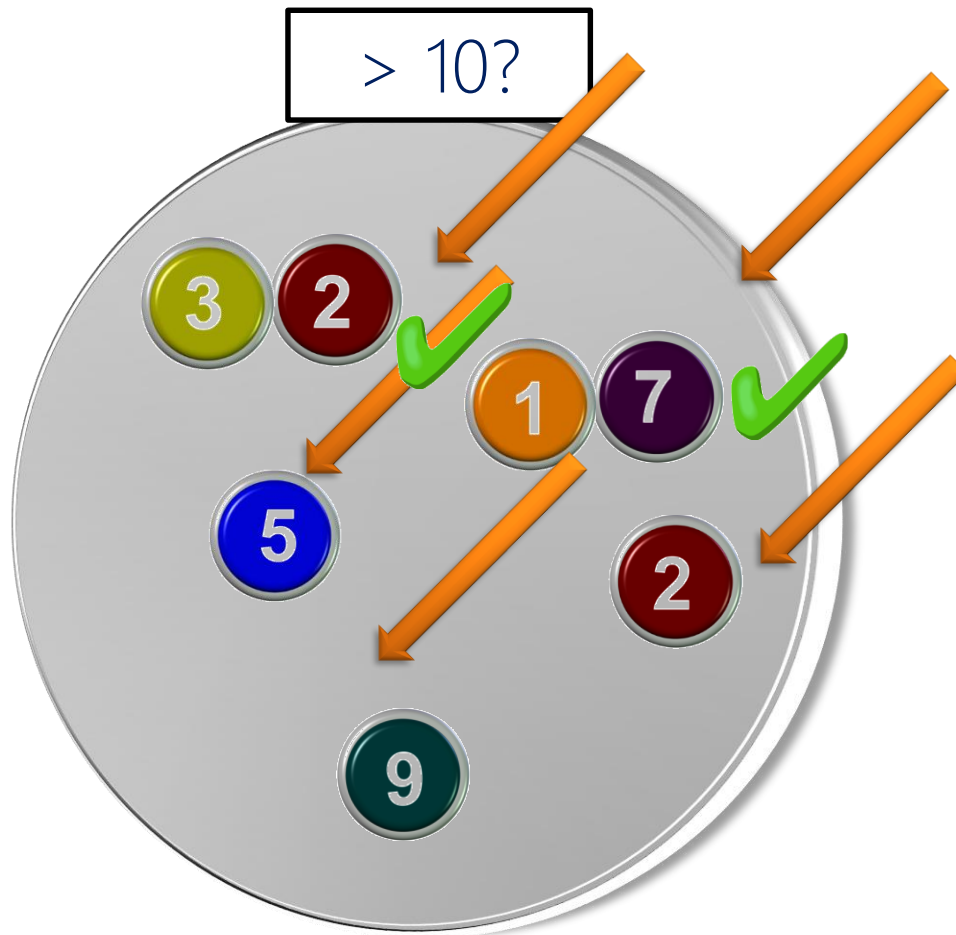


Collection Operations



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

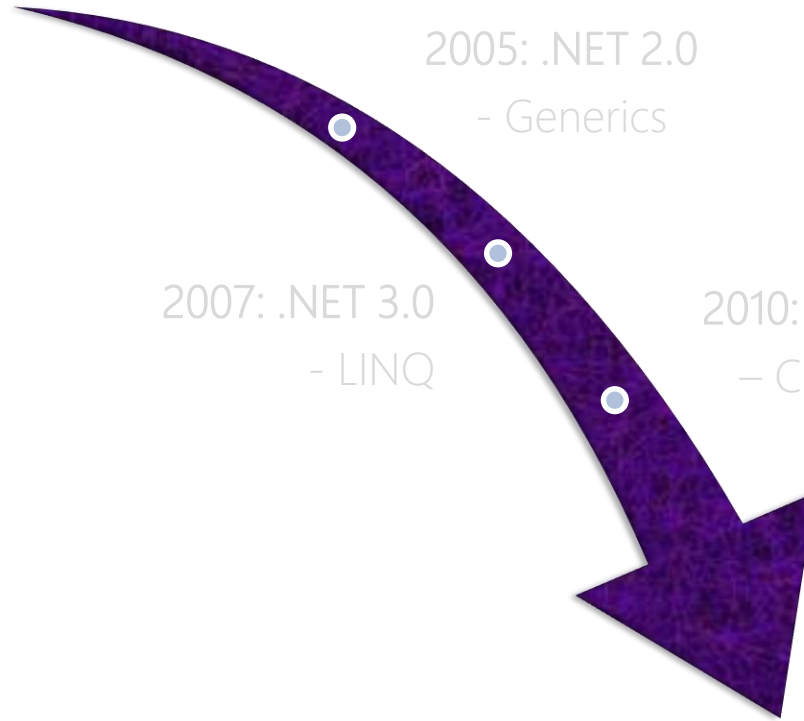
2002: .NET 1.0 - First
release

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

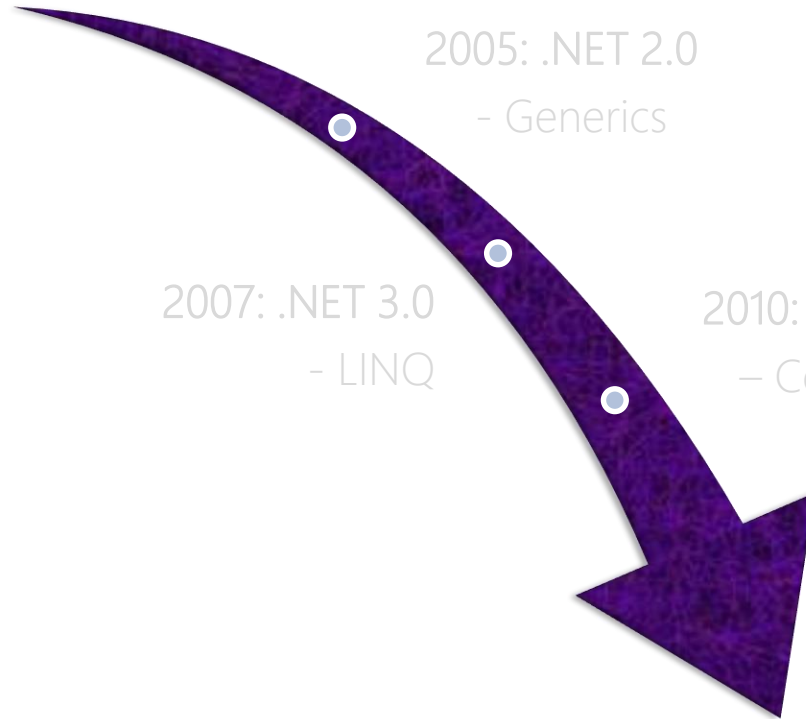
2002: .NET 1.0 - First
release

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

Can you give me
a list please?

A list of WHAT?

```
List<int>
```

```
List<Employee>
```

```
List<Control>
```

```
List<T>
```

What it's a list of



2002: .NET 1.0 - First
release



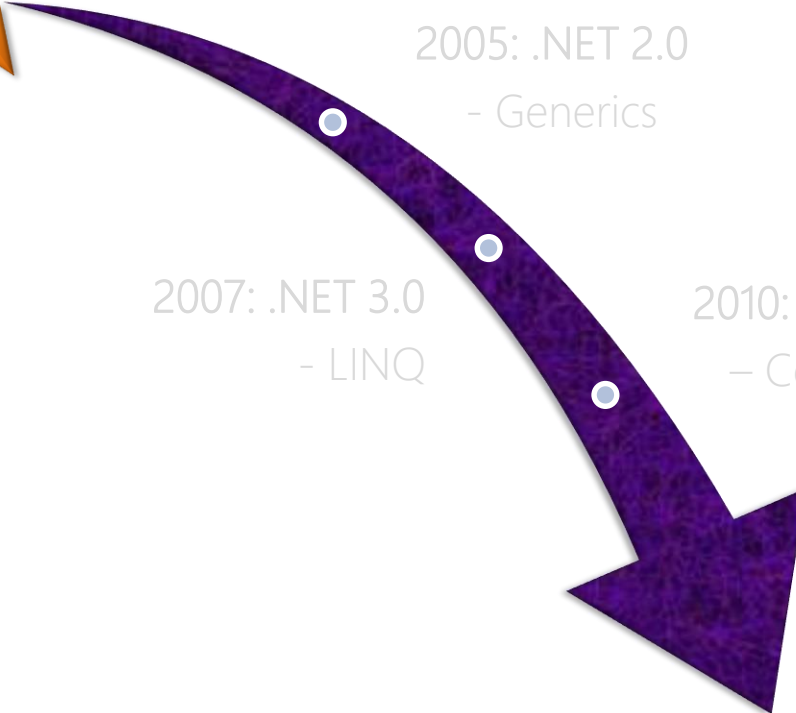
No generics in .NET 1.x!

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.

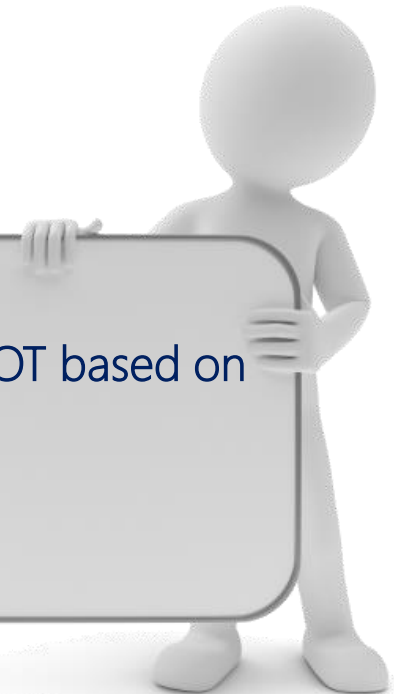
```
int[] arrayOfInts = new int[];
```



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

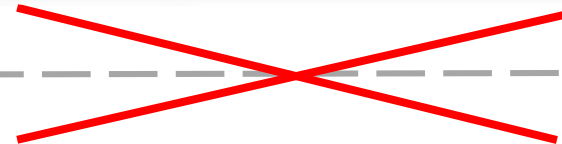


All other collections

eg. `ArrayList`

Mostly weakly typed

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



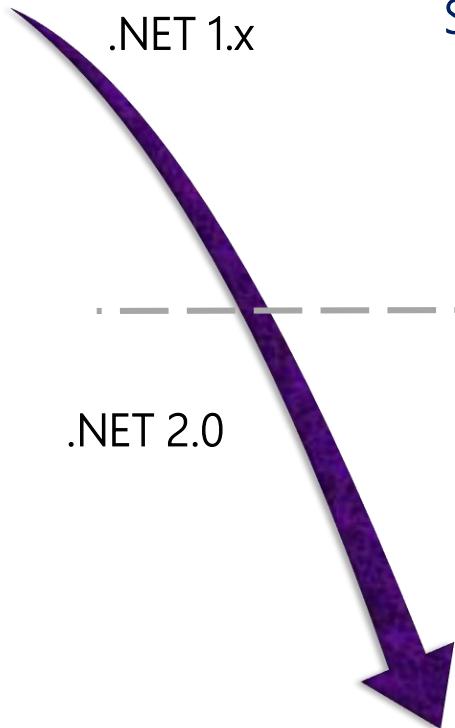
New generic collections

eg. `List<T>`

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

.NET 1.x

.NET 2.0



Arrays

HashSet<T>

Stack<T>

Queue<T>

LinkedList<T>

List<T>

Dictionary<TKey, TValue>

New generic collections



This the core
material for this
course

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

Collections in .NET

2002: .NET 1.0 - First
release

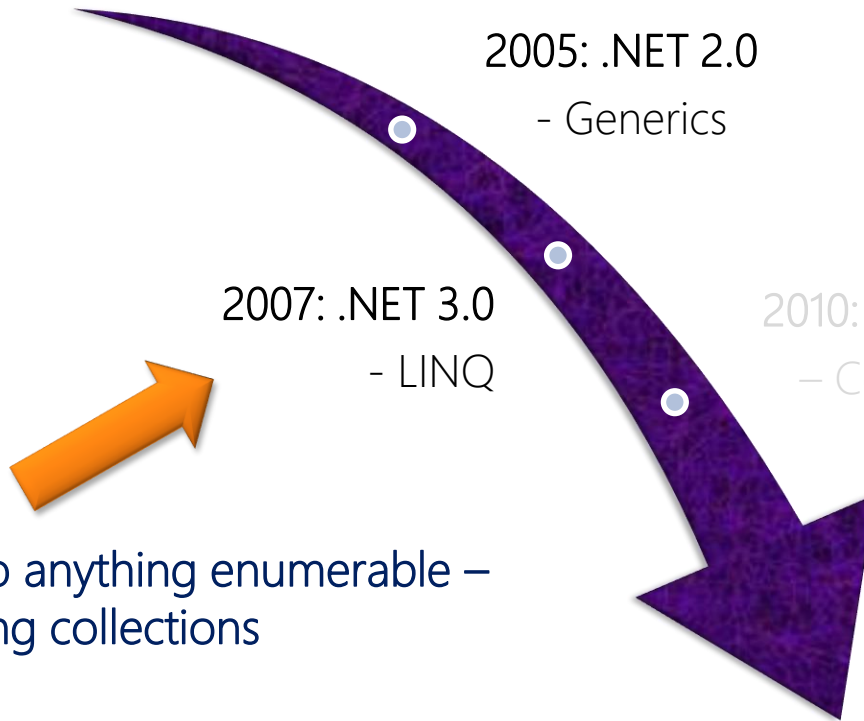
2005: .NET 2.0
- Generics

2007: .NET 3.0
- LINQ

2010: .NET 4.0
- Concurrent collections

2012: .NET 4.5
- Readonly interfaces,
Immutable collections

Adds operations to anything enumerable –
including collections



Collections in .NET

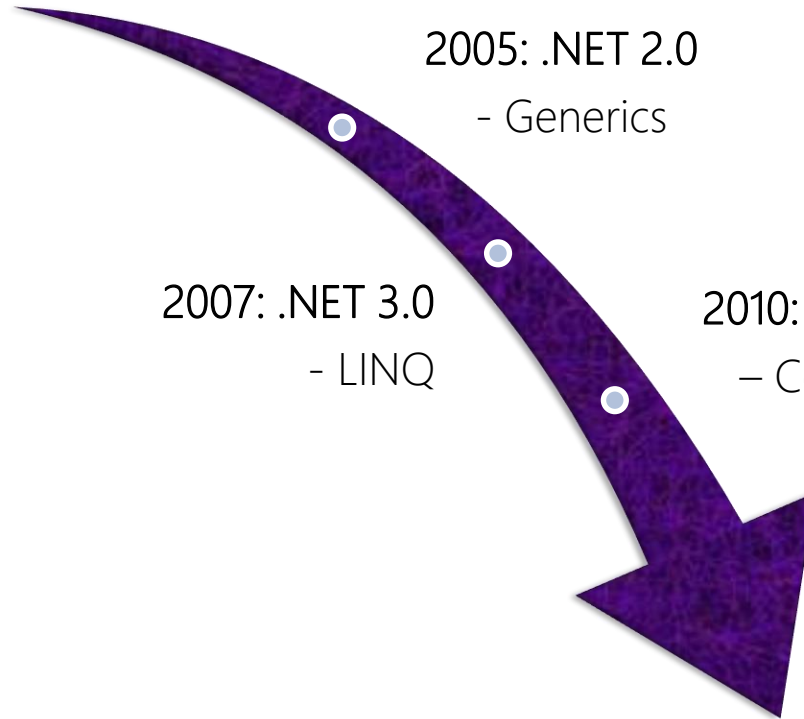
2002: .NET 1.0 - First
release

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

2002: .NET 1.0 - First
release

2005: .NET 2.0
- Generics

Concurrent collections:

Multi-threaded support



(We won't cover them)

2007: .NET 3.0
- LINQ

2010: .NET 4.0
- Concurrent collections

2012: .NET 4.5
- Readonly interfaces,
Immutable collections

Collections in .NET

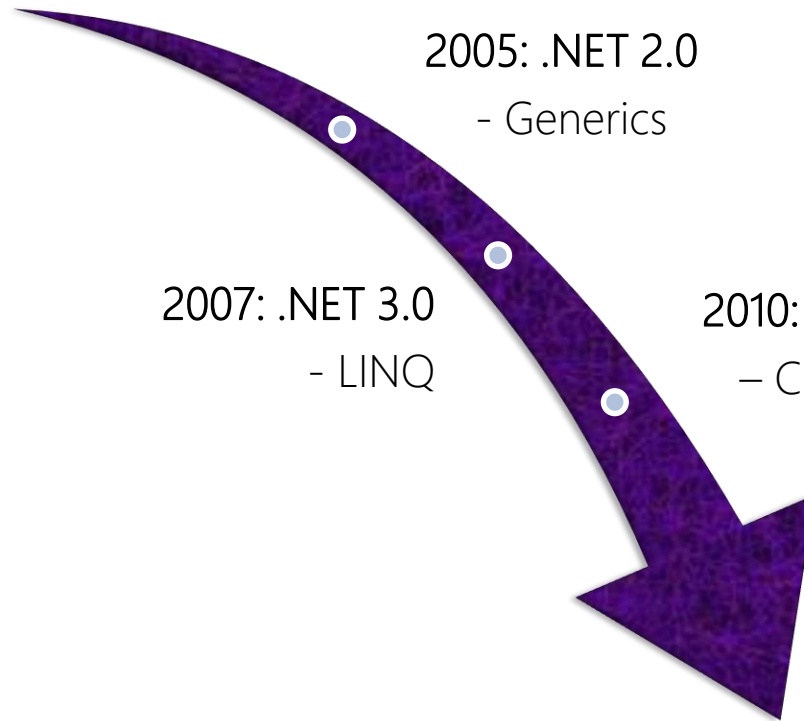
2002: .NET 1.0 - First
release

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

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

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

