Introduction to Collections

Simon Robinson
http://TechieSimon.com
@TechieSimon

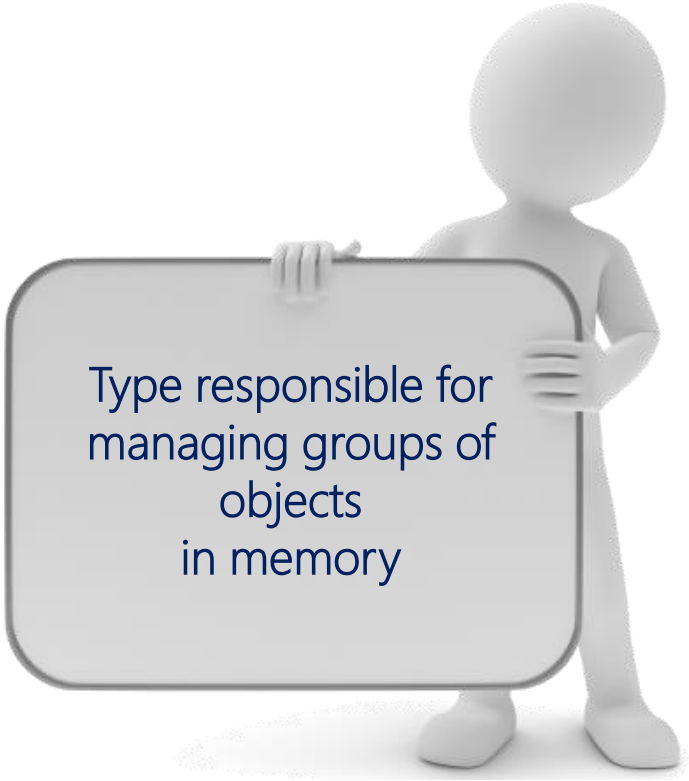
pluralsight
hardcore developer training

Overview

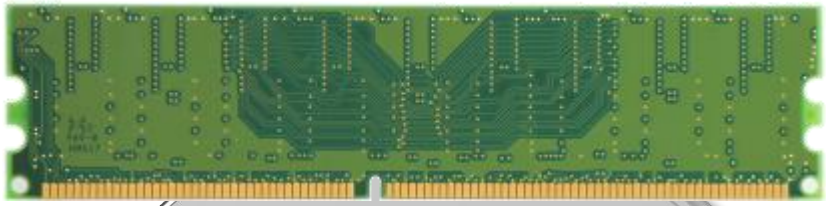
- What is a collection
  - Collection operations
- Types of collection
  - Lists
  - Dictionaries
  - Sets
- Collections in .NET
  - A brief history
  - The current collection landscape

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

What is a Collection?



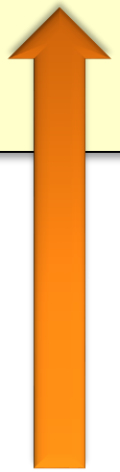
In memory!



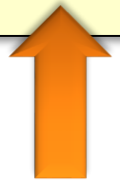
An integer!

A collection
(of integers)

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

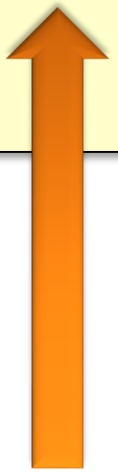


Not necessarily a collection

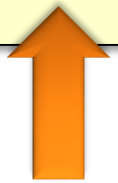


Yields employee instances
one at a time

```
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>

Stack<T>

ObservableCollection<T>

HashSet<T>

Array

List<T>

SortedSet<T>

Collection<T>

Dictionary<TKey, TValue>

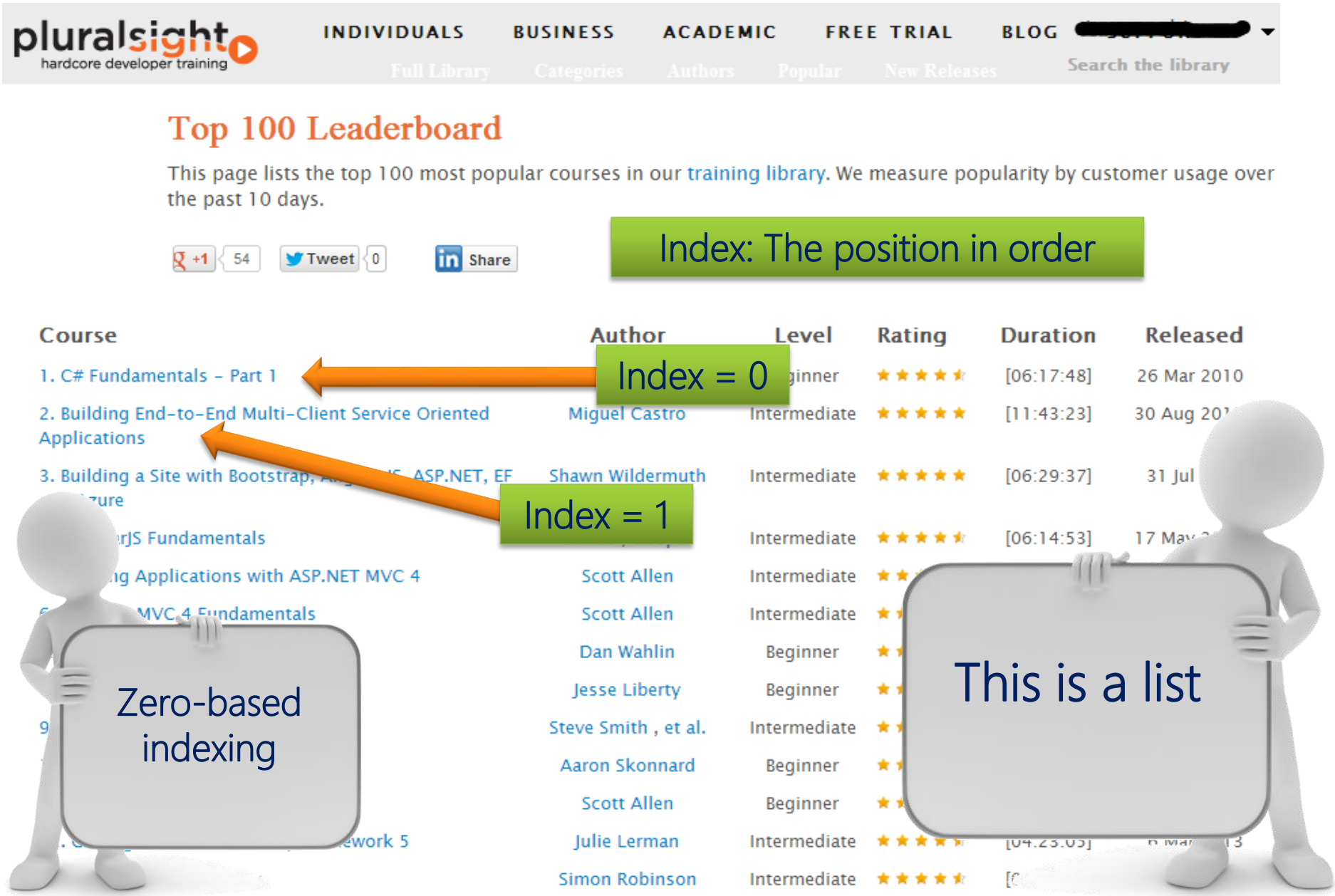
KeyedCollection<TKey, TItem>

Lists

Dictionaries

Sets

Need to access elements by position in order


pluralsight
hardcore developer training
INDIVIDUALS
BUSINESS
ACADEMIC
FREE TRIAL
BLOG
Full Library
Categories
Authors
Popular
New Releases
Search the library
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.
+1 54
Tweet 0
Share
Index: The position in order
Course
Author
Level
Rating
Duration
Released
1. C# Fundamentals – Part 1
Index = 0
2. Building End-to-End Multi-Client Service Oriented Applications
Miguel Castro
Intermediate
[11:43:23]
3. Building a Site with Bootstrap, ASP.NET, EF
Shawn Wildermuth
Intermediate
[06:29:37]
Index = 1
Intermediate
[06:14:53]
Scott Allen
Intermediate
Scott Allen
Intermediate
Dan Wahlin
Beginner
Jesse Liberty
Beginner
Steve Smith , et al.
Intermediate
Aaron Skonnard
Beginner
Scott Allen
Beginner
Julie Lerman
Intermediate
Simon Robinson
Intermediate
Zero-based indexing
This is a list

List Types in .NET:

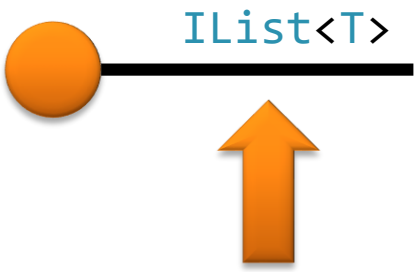
`T[]`

`List<T>`

`Collection<T>`

`ReadOnlyCollection<T>`

`ObservableCollection<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…

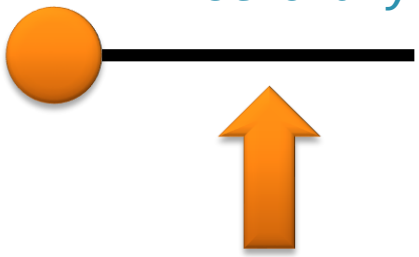
Access elements using a **key**

`Dictionary<TKey, TValue>`

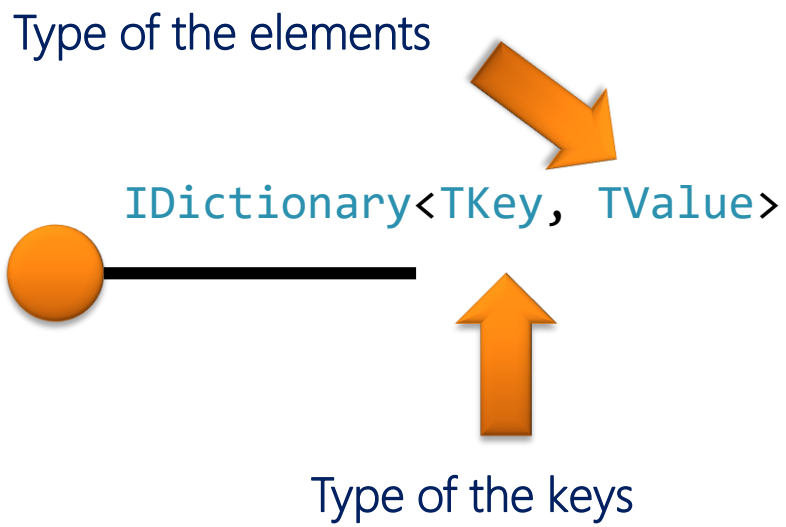
Most widely used dictionary type

Implemented using a hash table

`IDictionary<TKey, TValue>`



Contract for dictionaries


Type of the elements
IDictionary<TKey, TValue>
Type of the keys

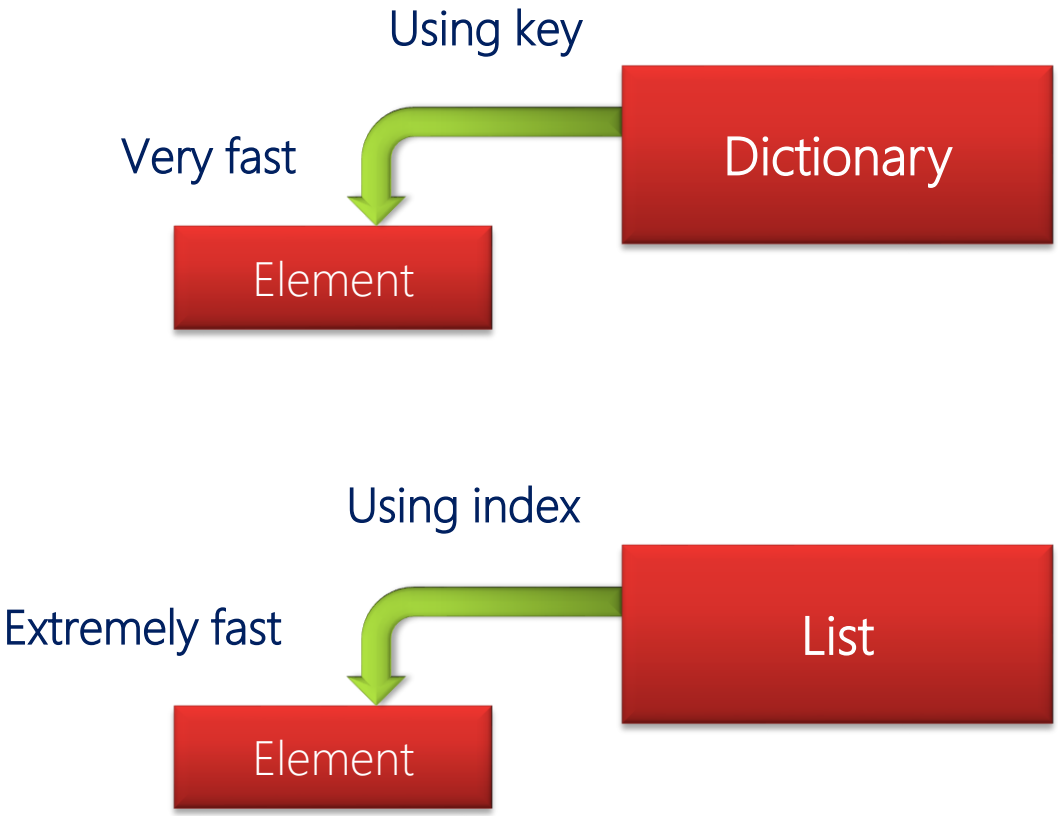
Declare a dictionary that allows looking up by name:

```
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

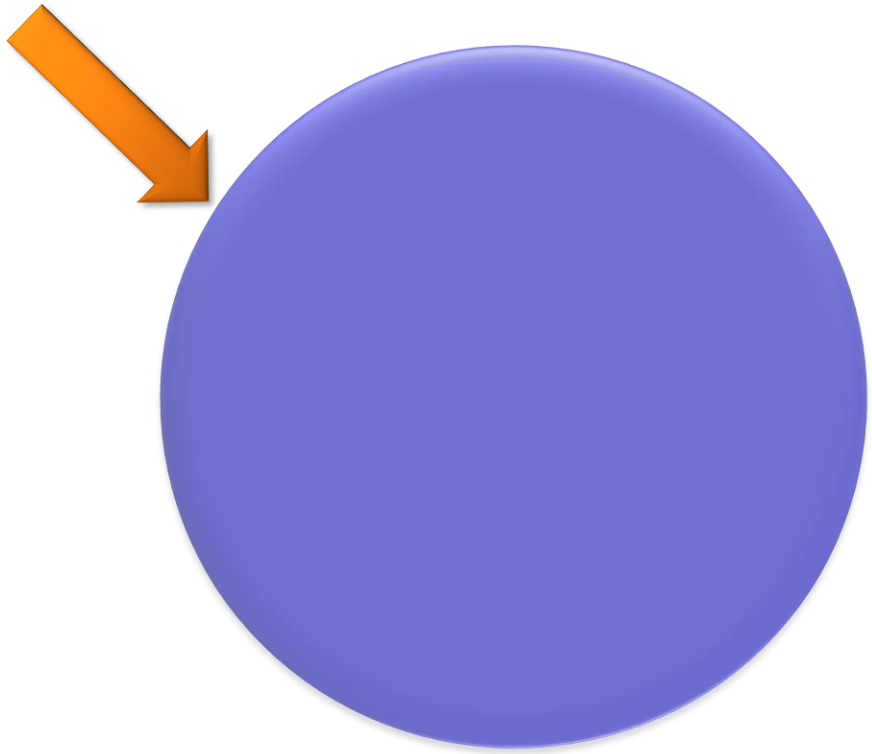
Easy to misuse

Expect keys to behave a certain way

(Cover later in this course)

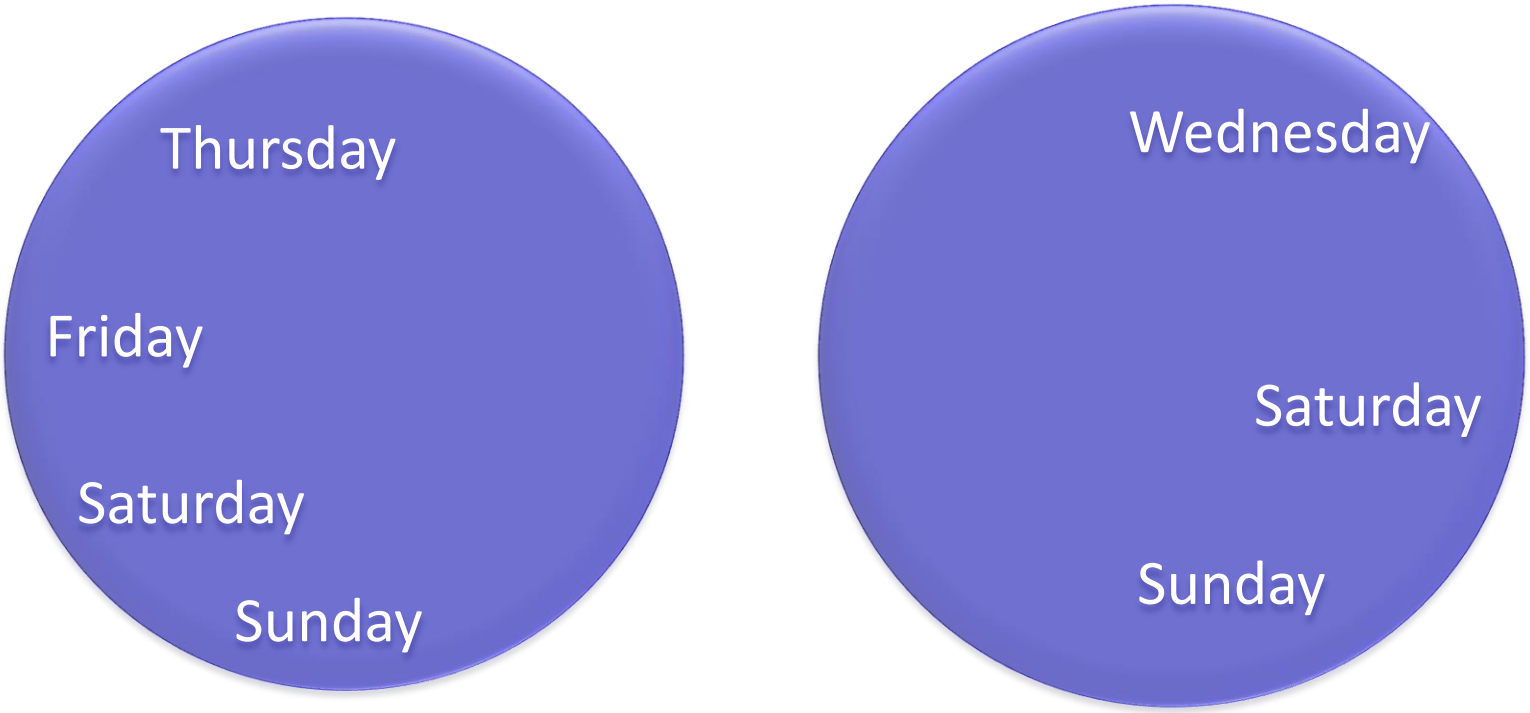
Sets

Sets treat the collection as a whole



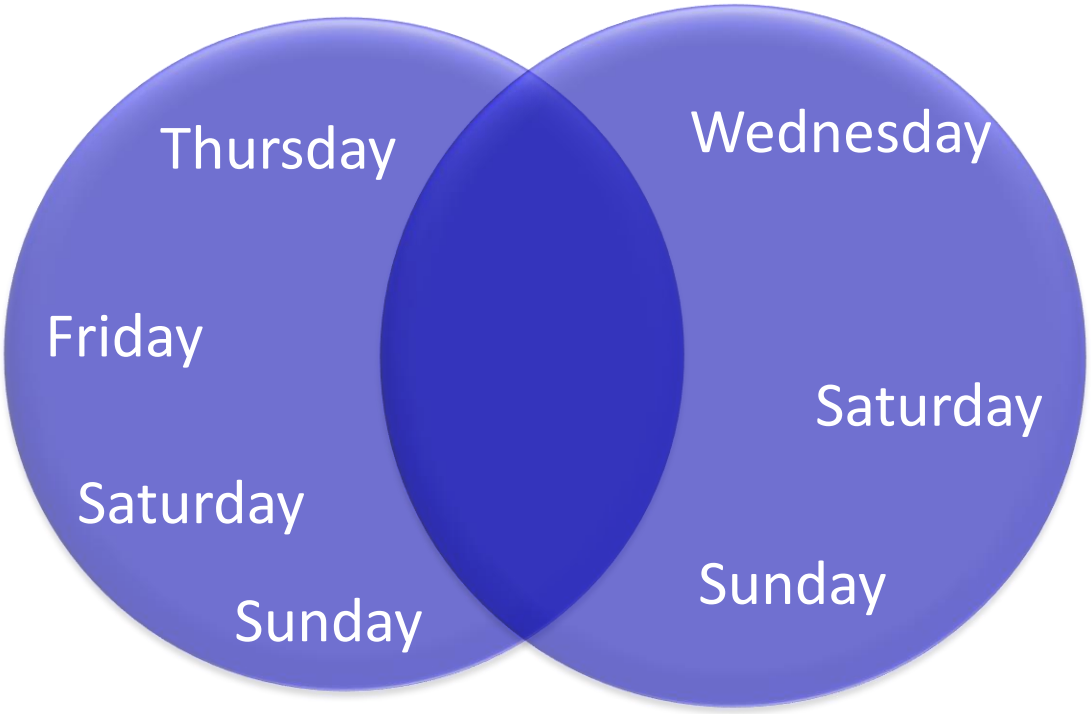
Sets

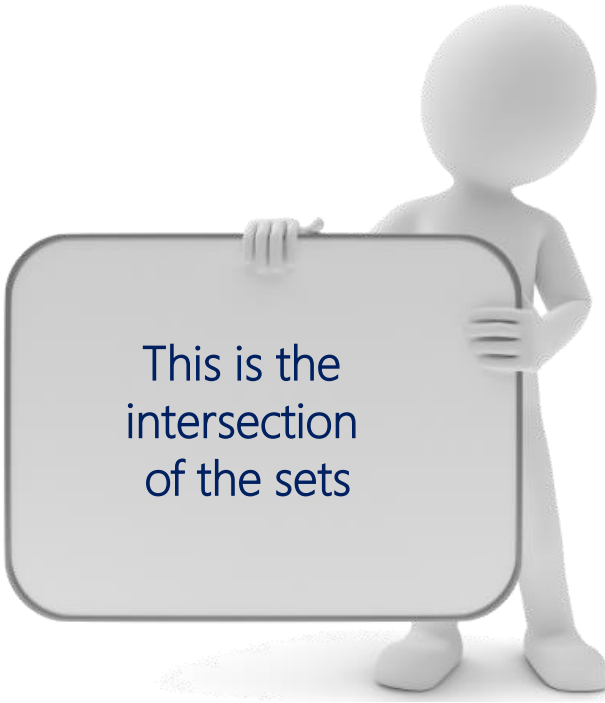
What days are in both collections?



Sets

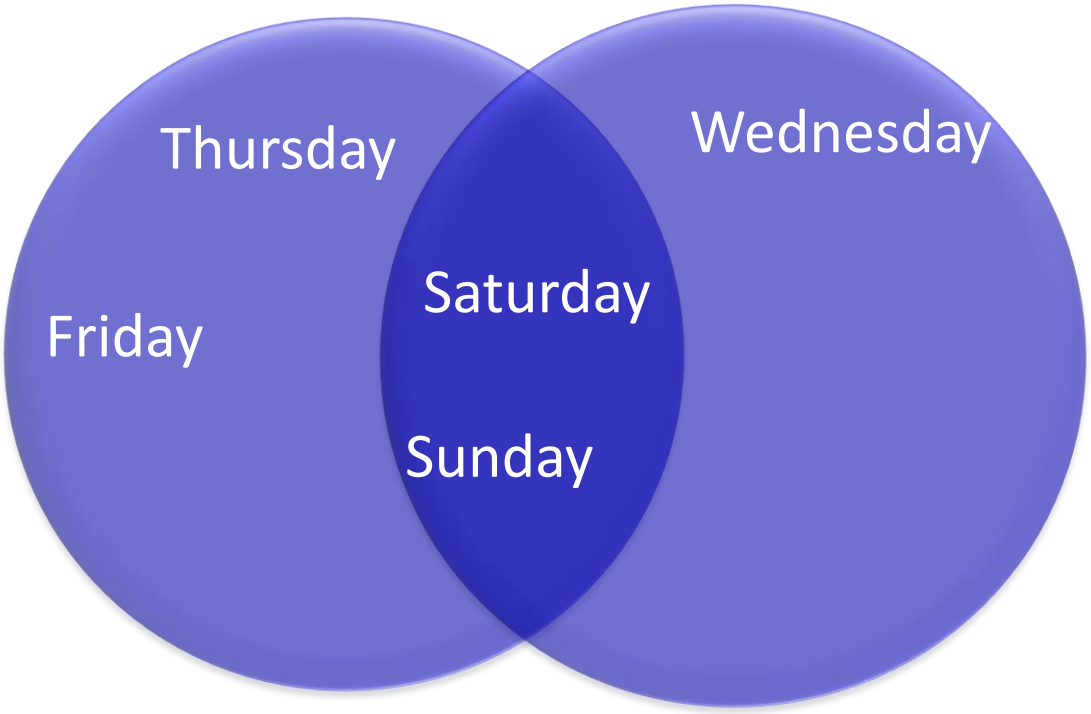
What days are in both collections?





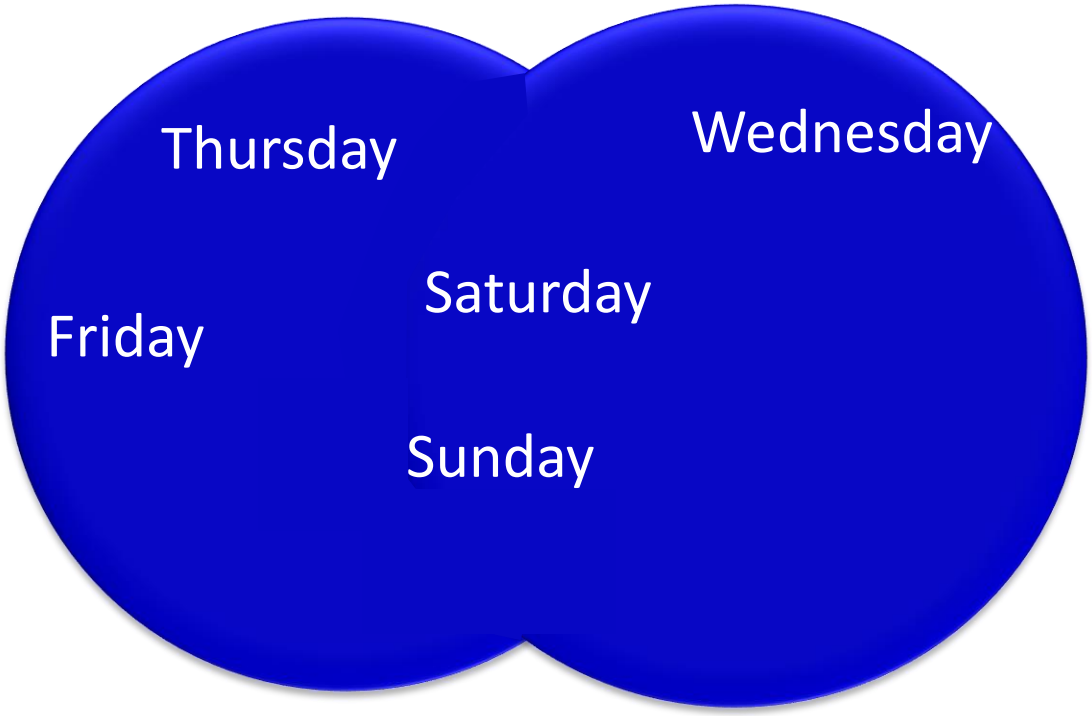
Sets

What days are in either collection?



Sets

What days are in either collection?



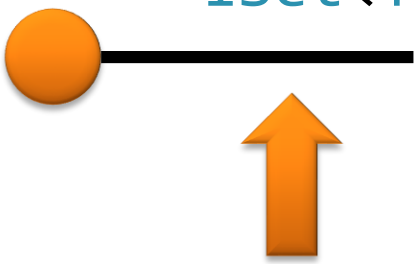
Like SQL
`UNION DISTINCT`

Sets

Most common set class

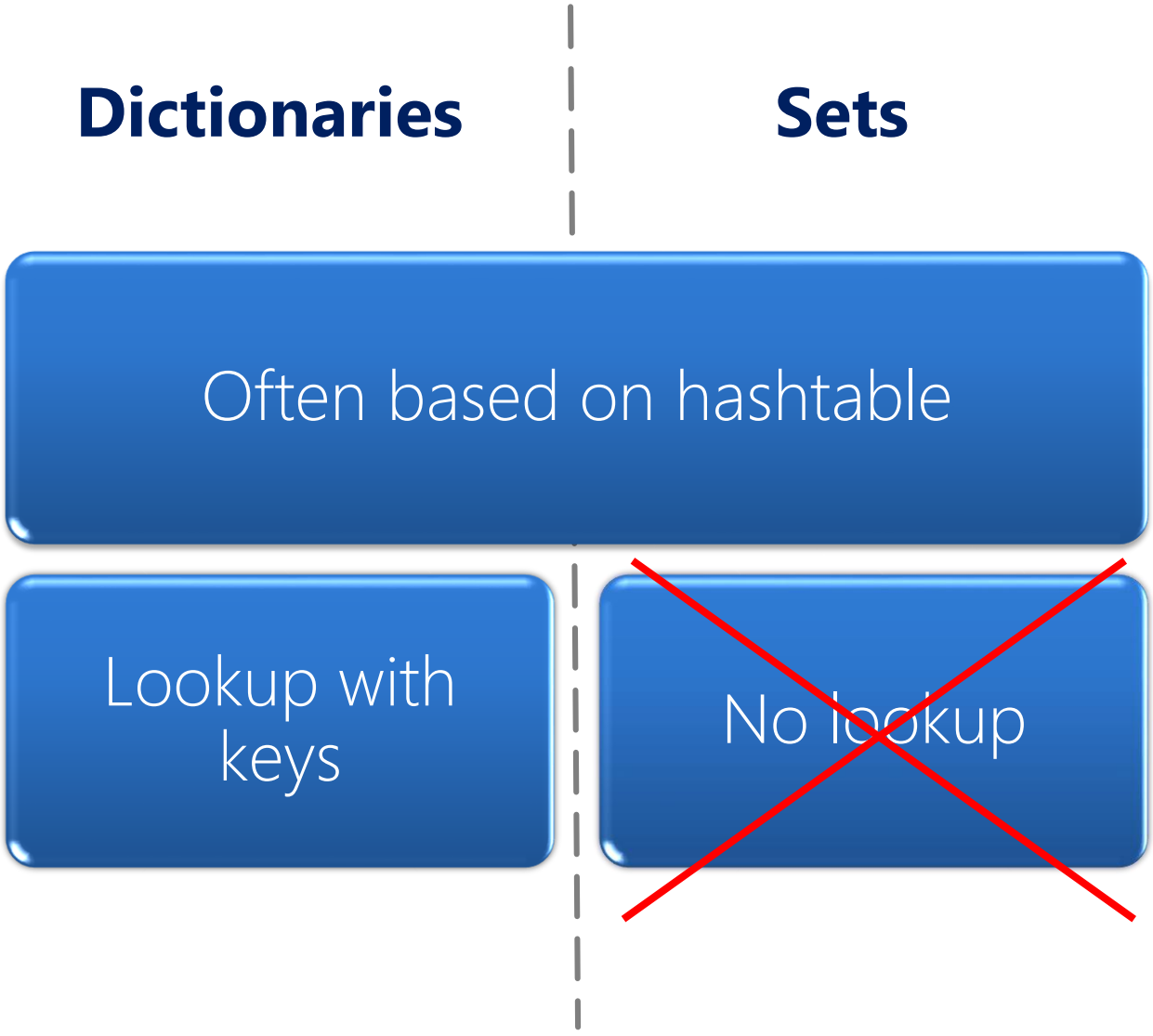
`HashSet<T>`

`ISet<T>`



Contract for sets

Sets are collections that support being combined to form new sets


Dictionaries
Sets
Often based on hashtable
Lookup with keys
No lookup

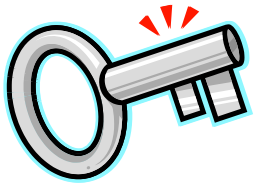
Collection Operations

Look up an element

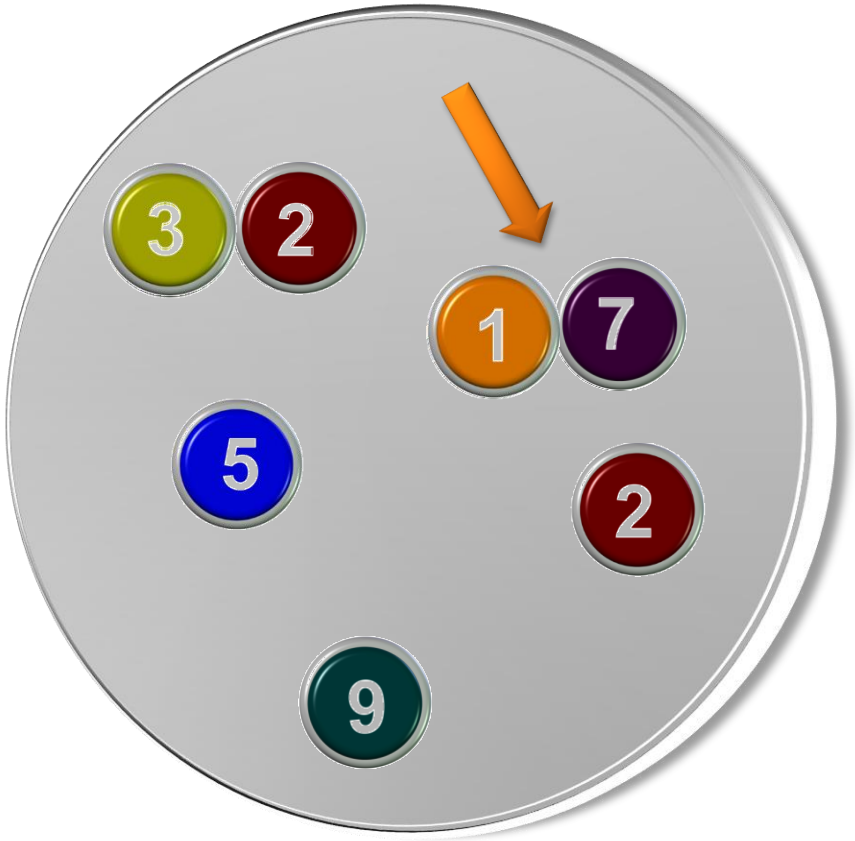
(by index or key)

Enumerate the elements

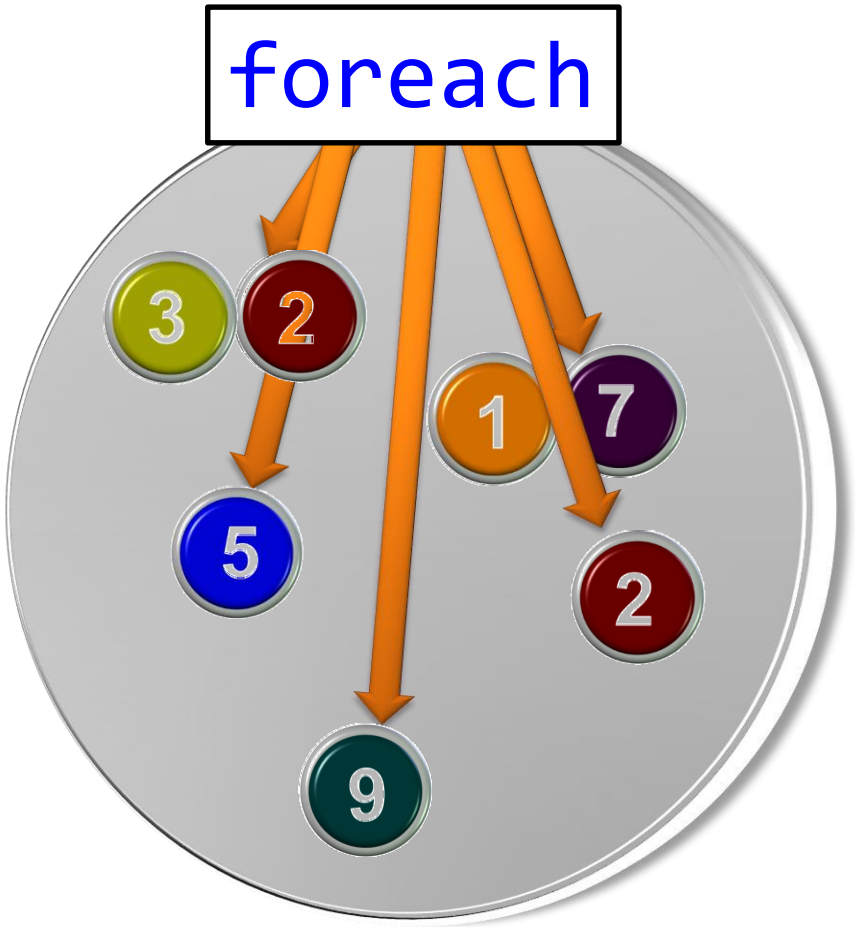
Looking up an item



(Key or index)



Enumerating a collection



Enumerating a collection

List

Items enumerated in index order

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

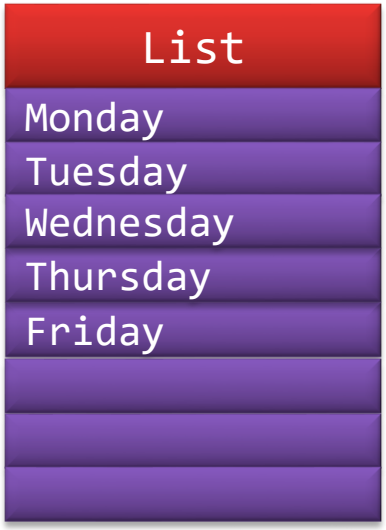
(by index or key)

Enumerate the elements

Add an element

Remove an element

Lists: Add item at a particular place



This is called inserting

Add an item:

Don't care where it goes

Collection Operations

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

Look up an element

(by index or key)

Enumerate the elements

Add an element

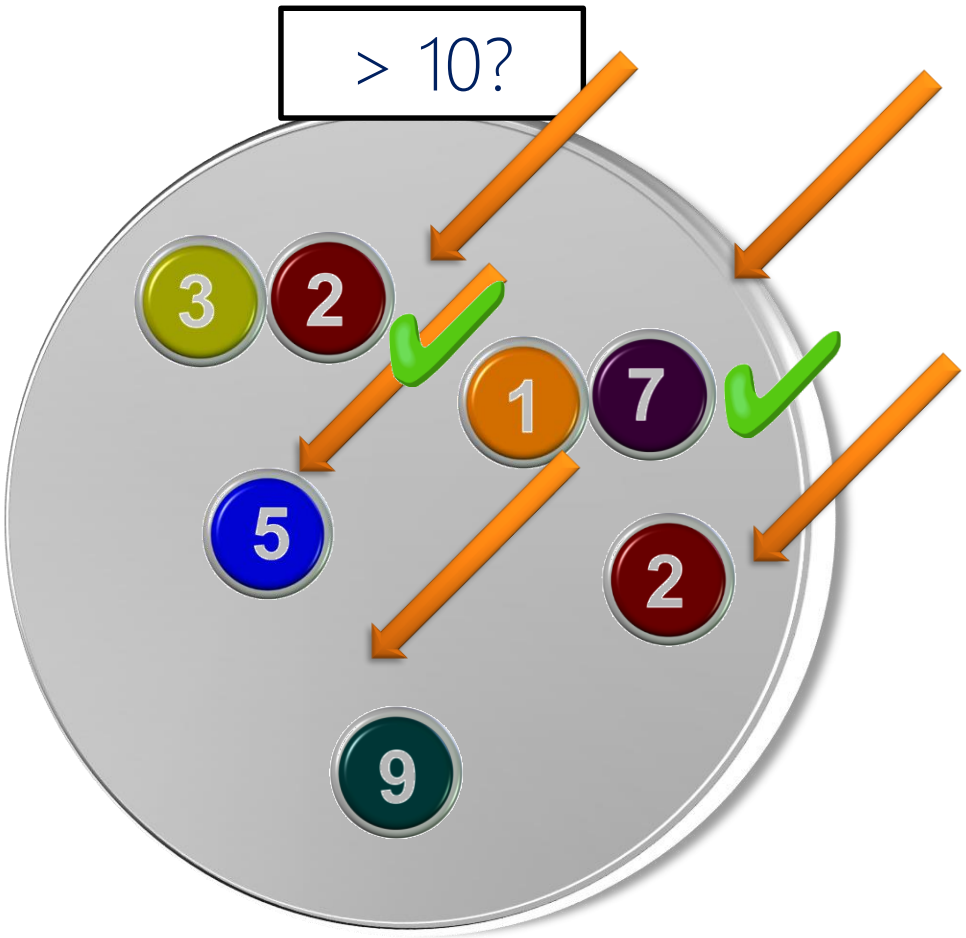
Remove an element

LISTS: Insert an element

(Replace an element)

Derived Operation:

Filter a list: Enumerate

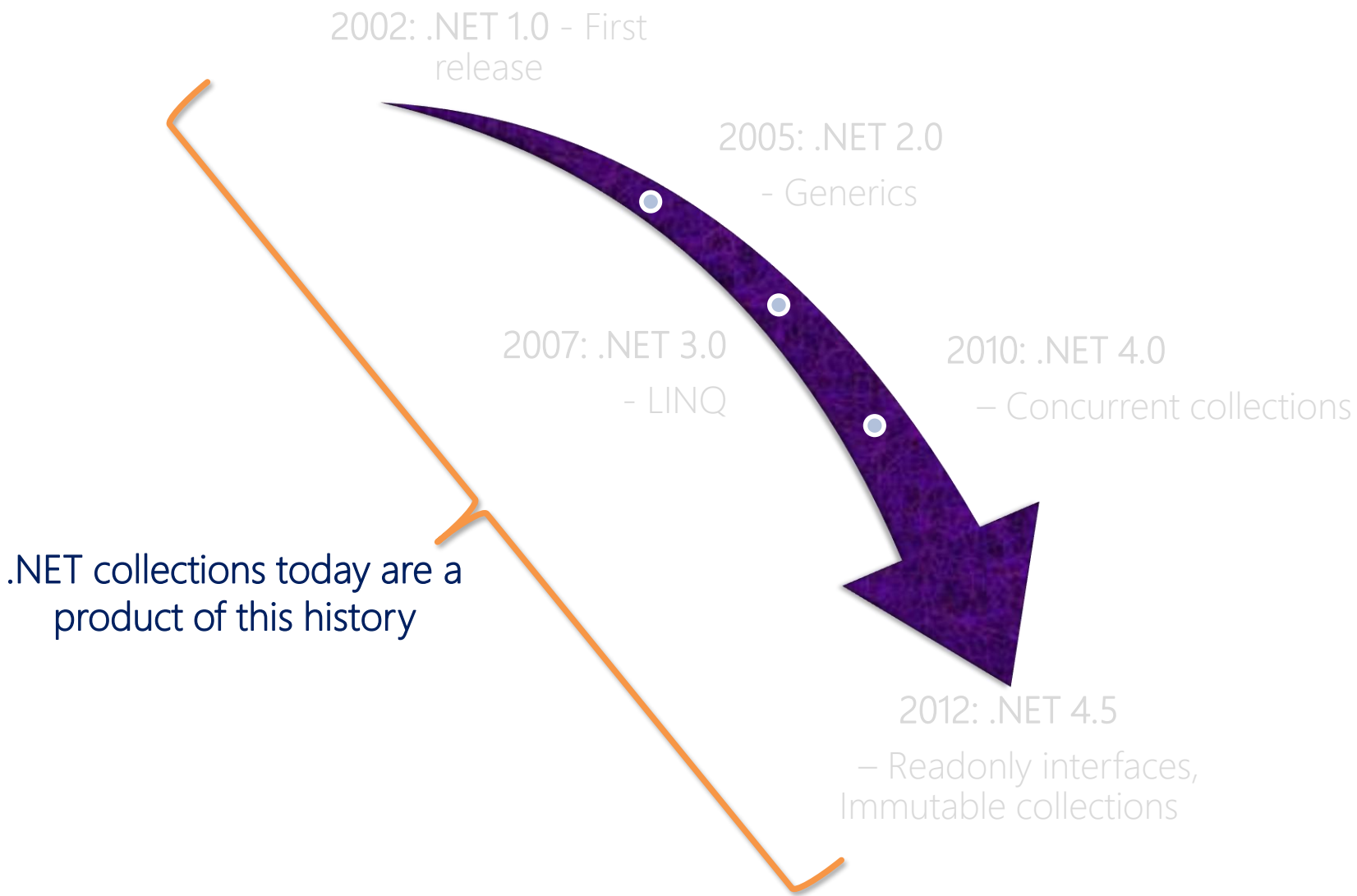


LINQ is great for these kinds of derived operations

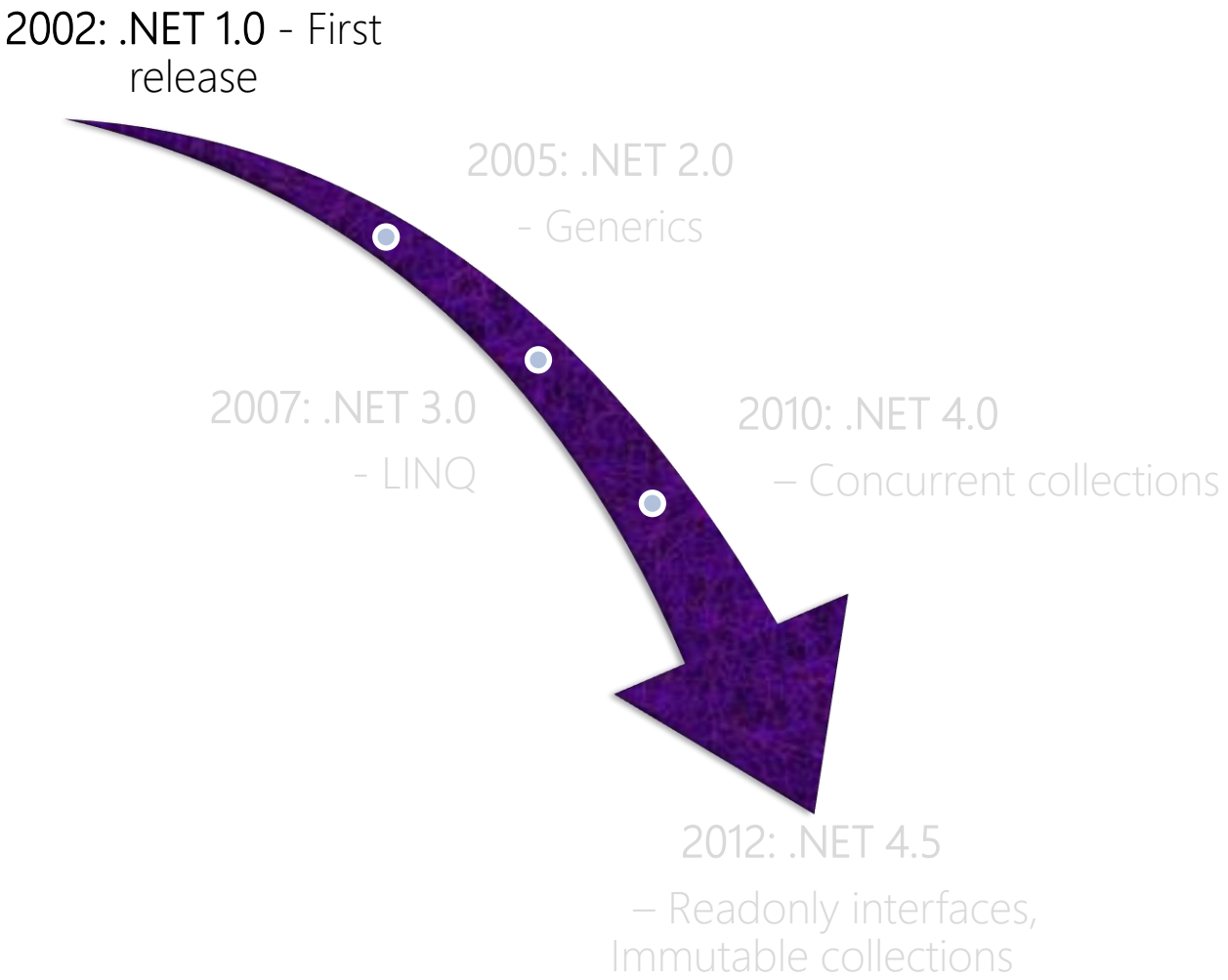
But there's native support in

`T[]` `List<T>`

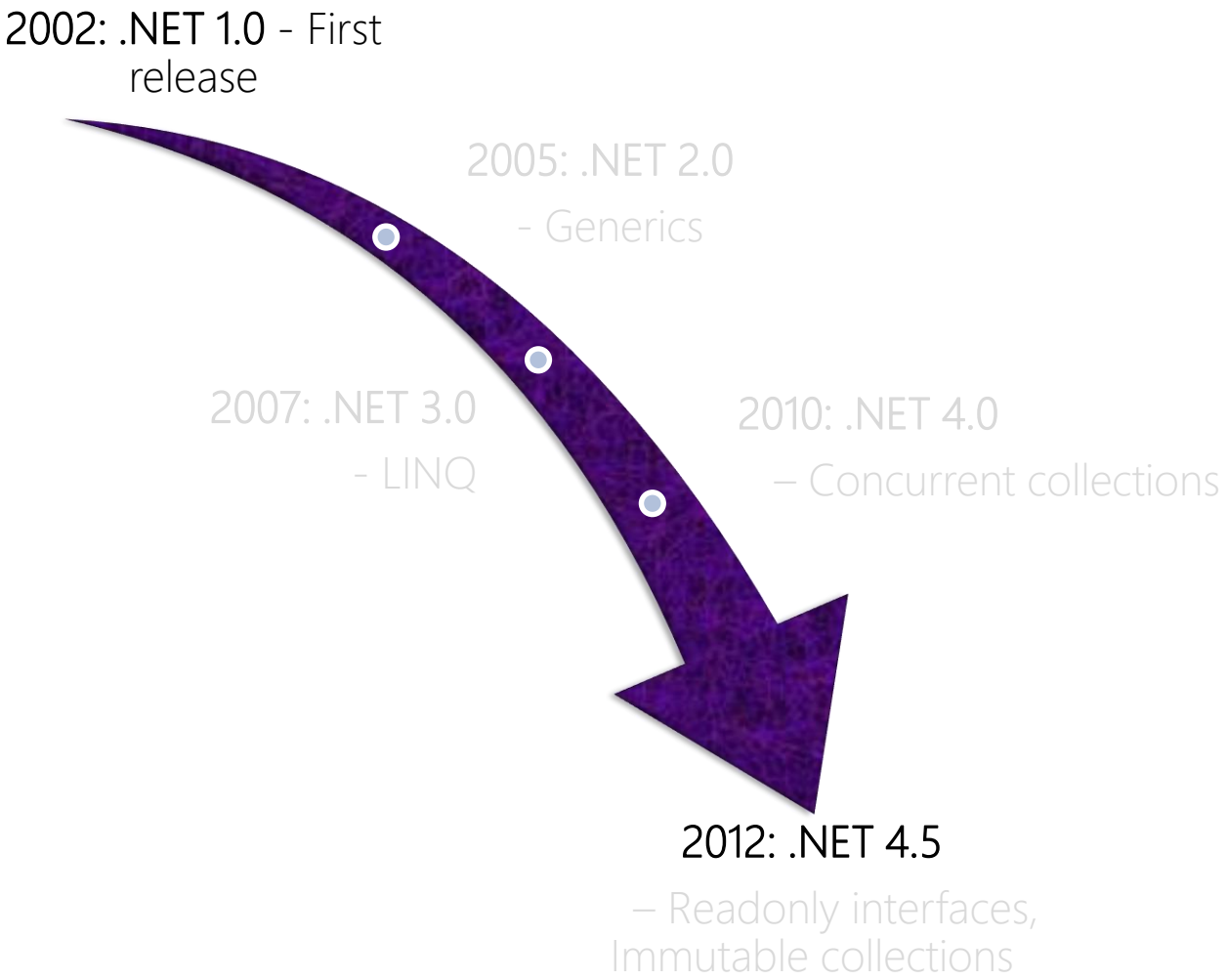
Collections in .NET



Collections in .NET



Collections in .NET



Collections are Generic

Can you give me a list please?

A list of WHAT?

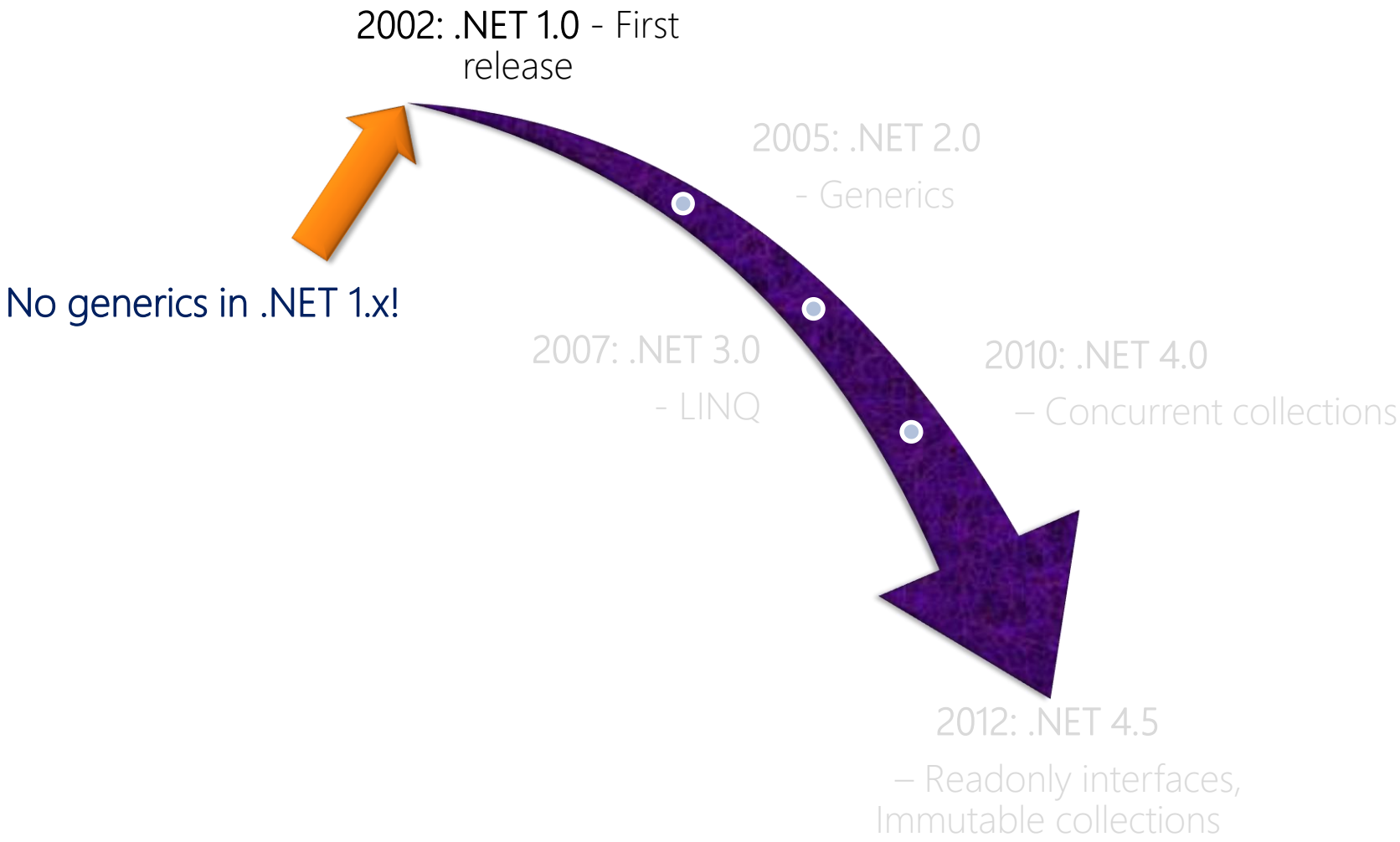
```
List<int>
```

```
List<Employee>
```

```
List<Control>
```

```
List<T>
```

What it's a list of


2002: .NET 1.0 - First release
2005: .NET 2.0
- Generics
No generics in .NET 1.x!
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)

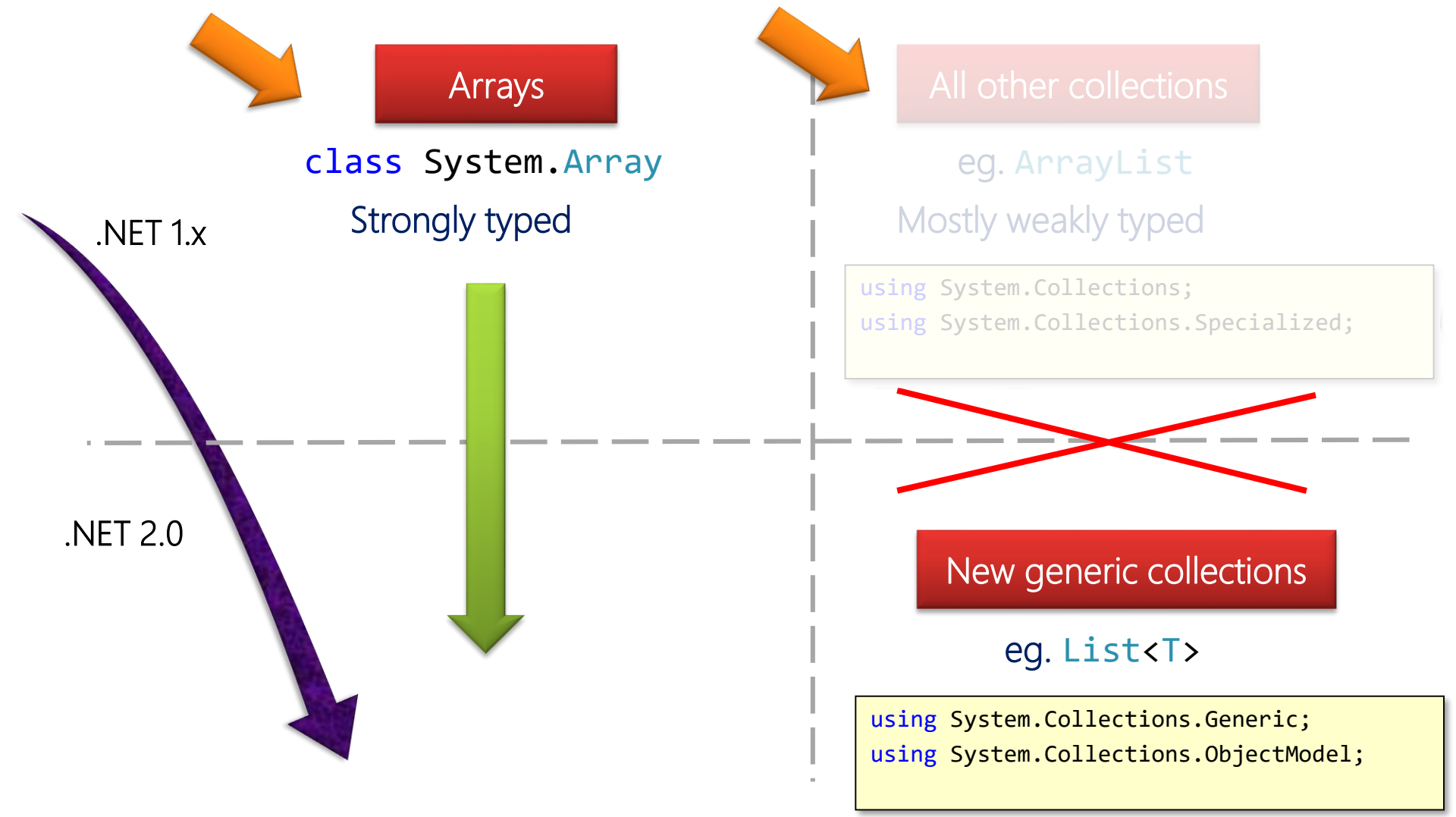
Aaargh!

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


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

HashSet<T>

Stack<T>

Queue<T>

LinkedList<T>

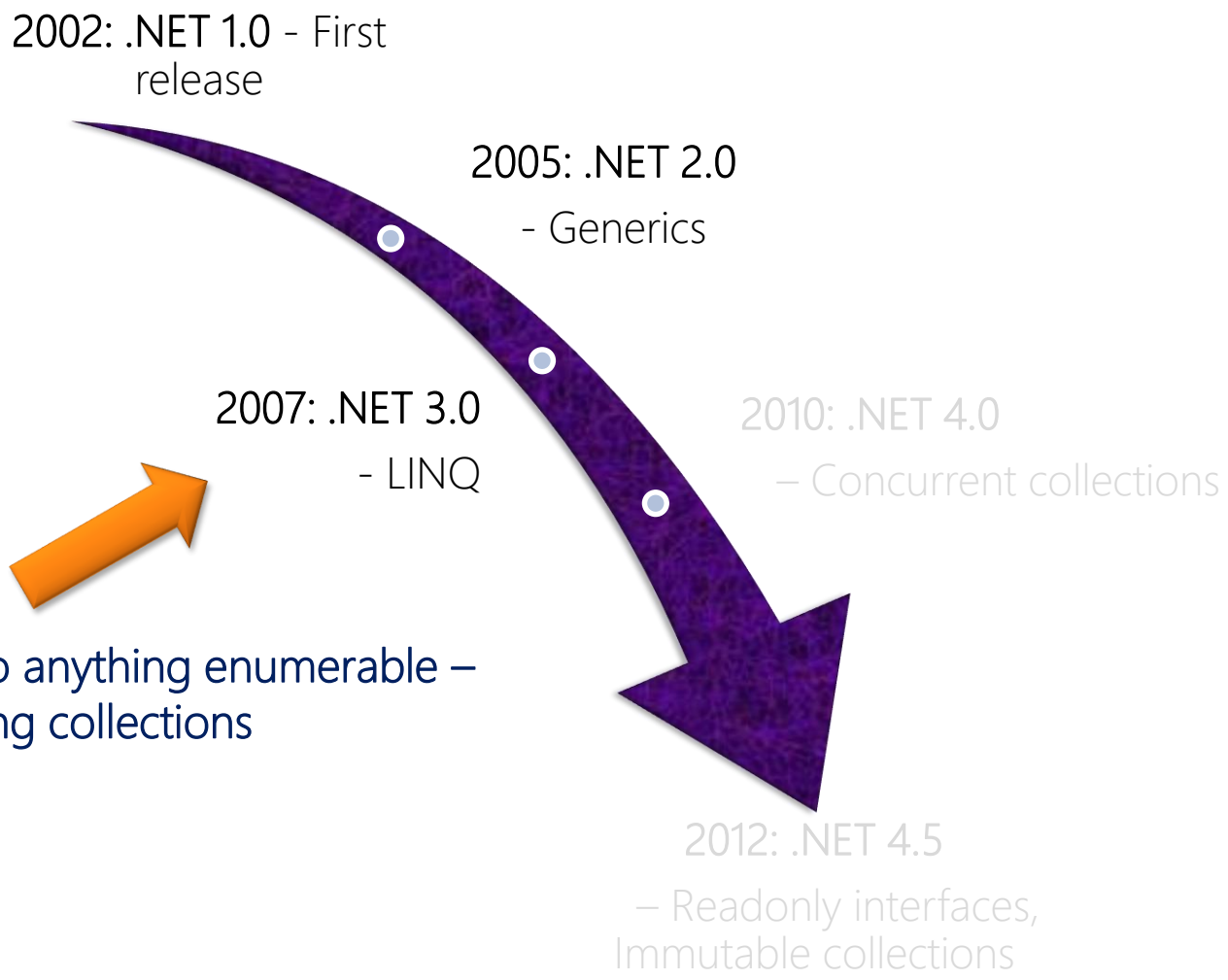
List<T>

Dictionary<TKey, TValue>

New generic collections

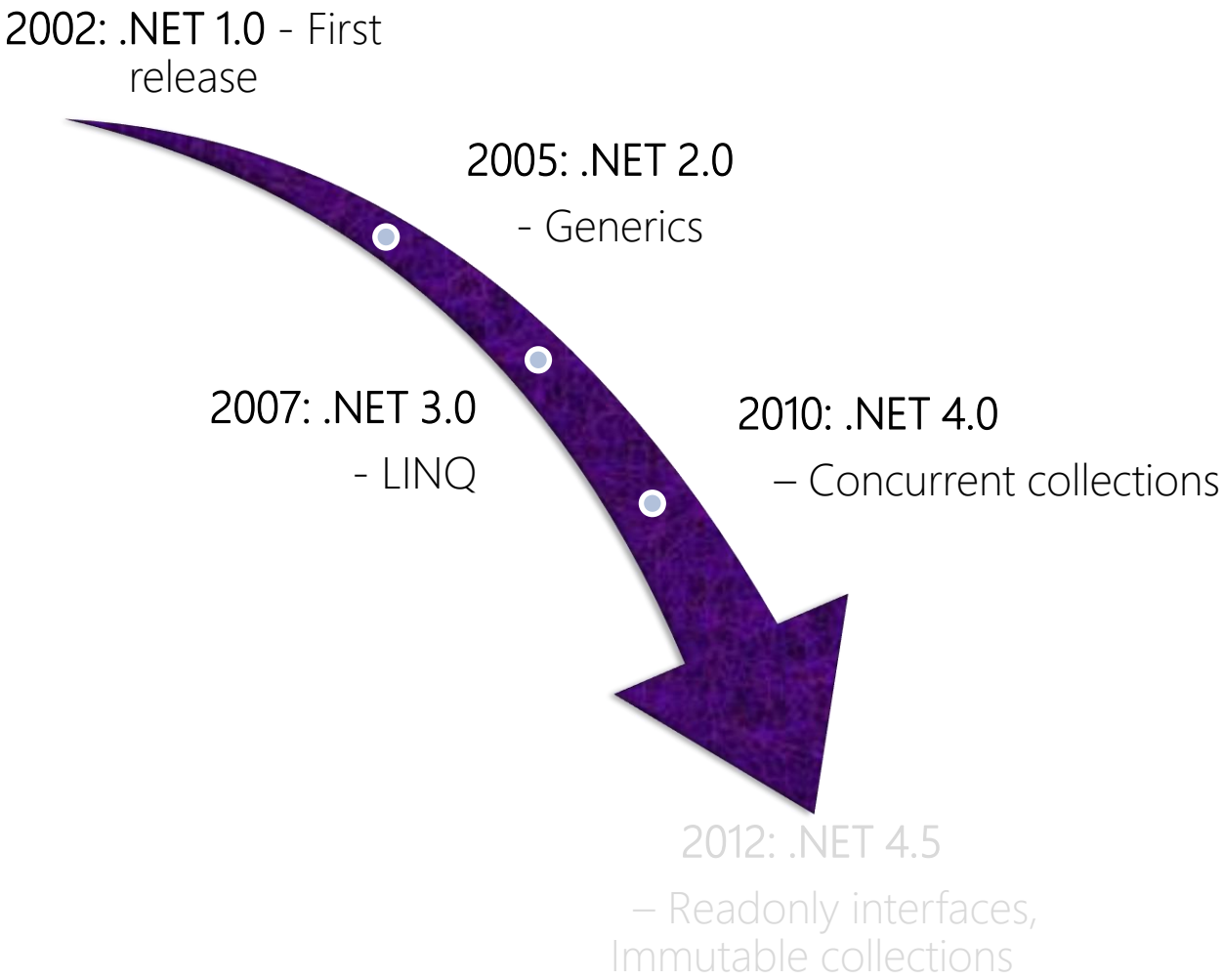

```
using System.Collections.Generic;
using System.Collections.ObjectModel;
```

Collections in .NET



Adds operations to anything enumerable – including collections

Collections in .NET



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

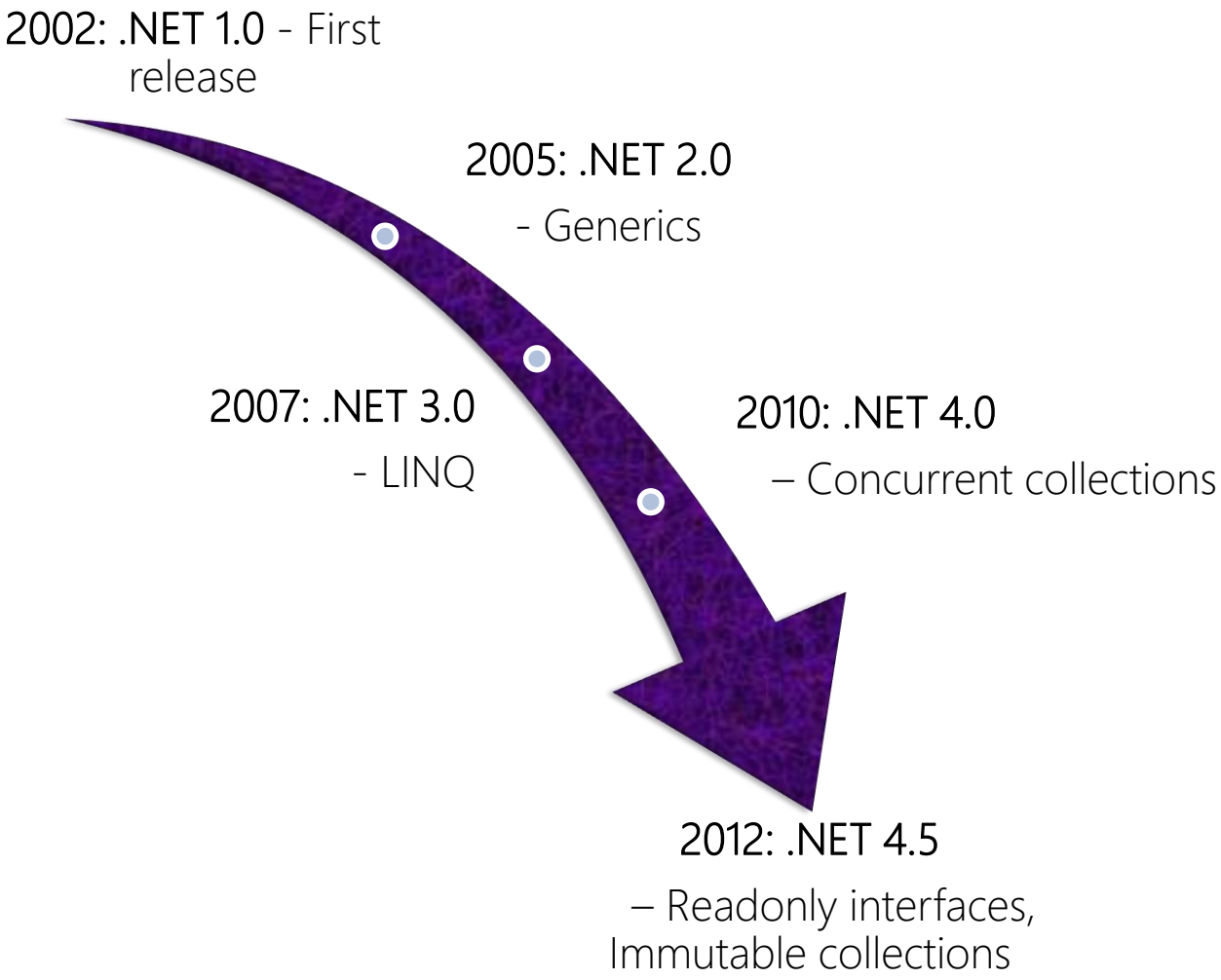
2012: .NET 4.5
– Readonly interfaces, Immutable collections

Concurrent collections:

Multi-threaded support

(We won't cover them)

Collections in .NET



Readonly Interfaces

Eg.

`IReadOnlyList<T>`

`IReadOnlyCollection<T>`

Readonly contracts

Previously, most collection interfaces had read-write contracts

`IList<T>`

`ICollection<T>`

Readonly Interfaces

Eg.

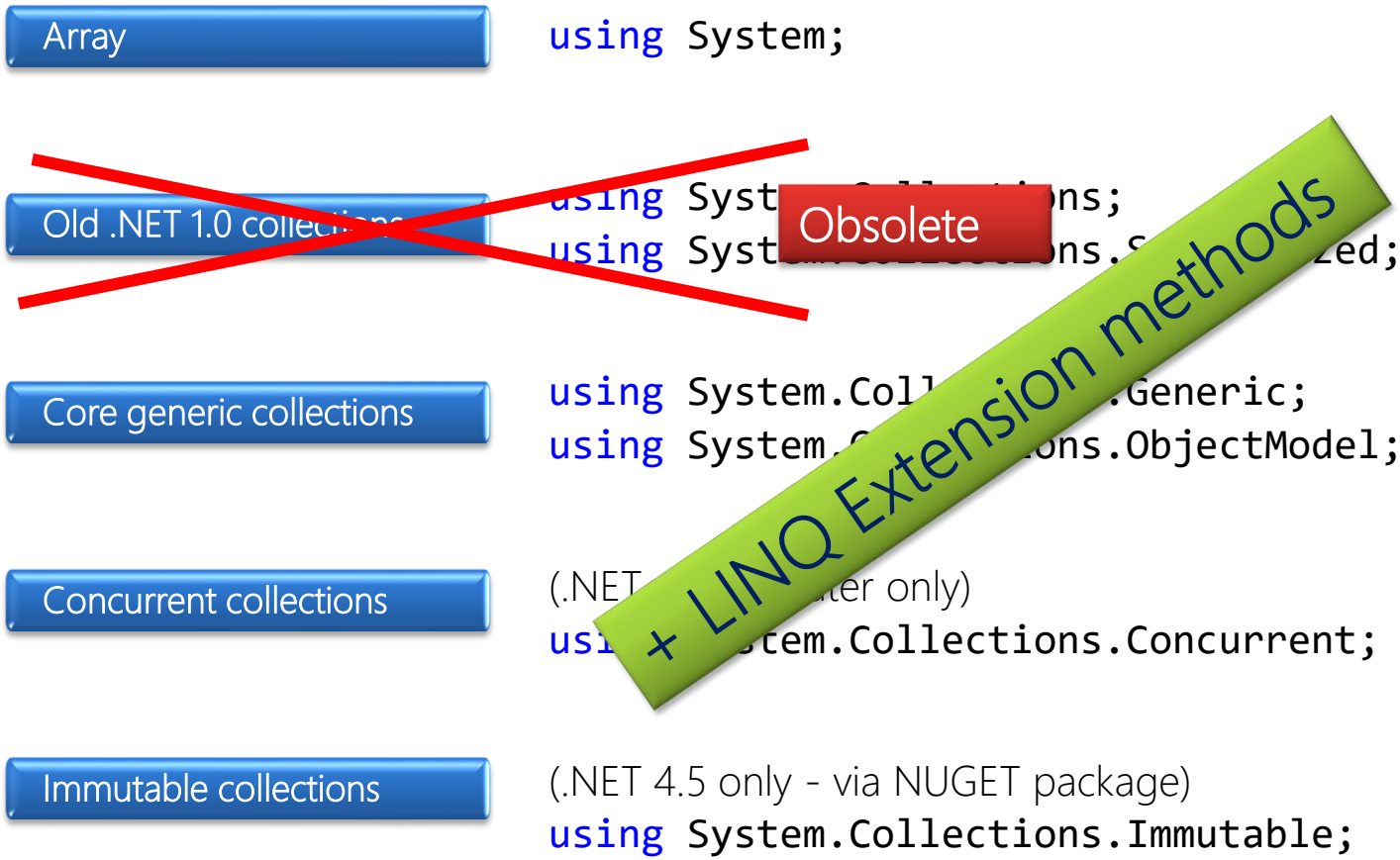
```
IReadOnlyList<T>

IReadOnlyCollection<T>
```

Readonly contracts

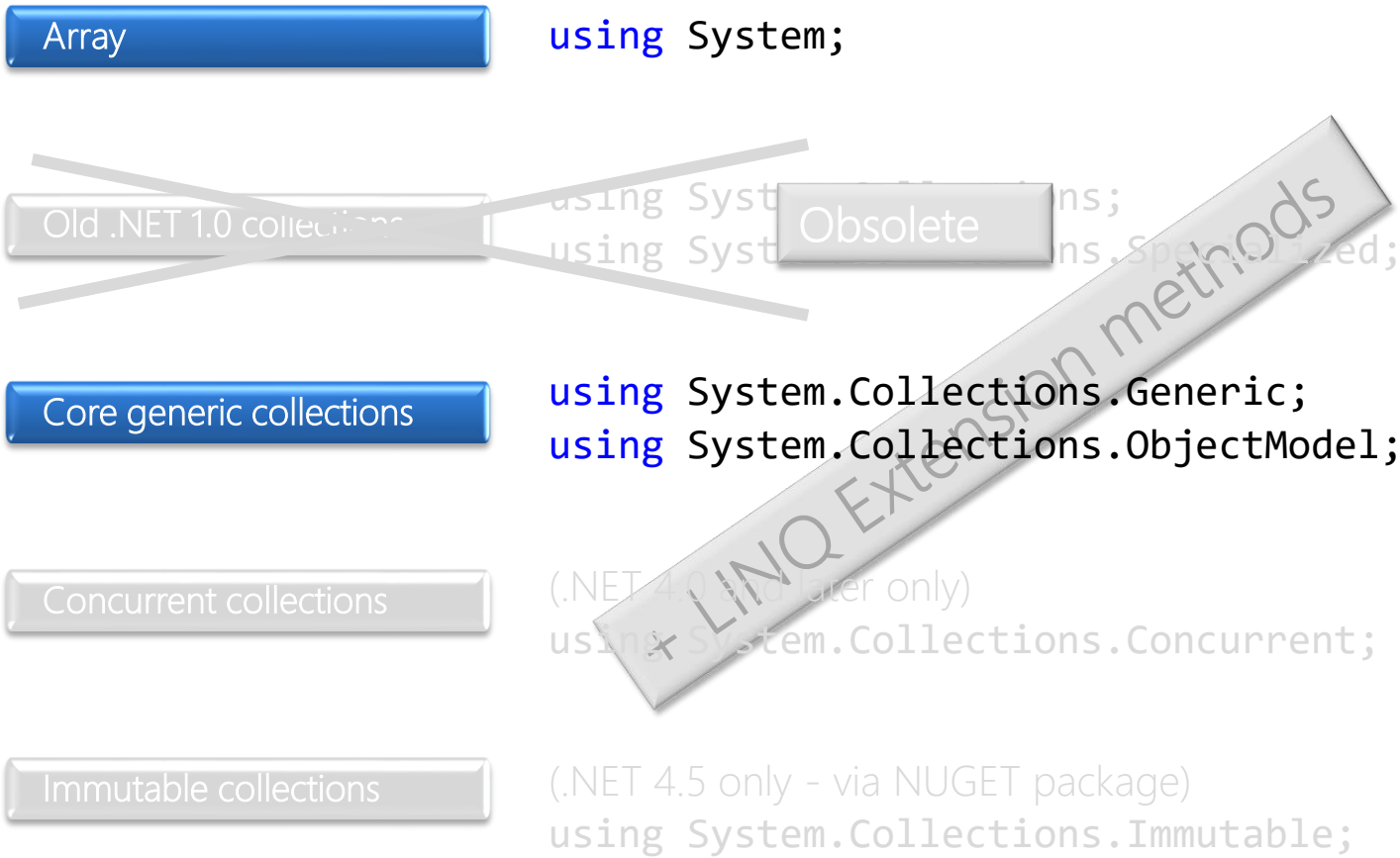
Immutable Collections

Separate NuGet package

C# Collections Today



C# Collections Today



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