Collection Interfaces

Simon Robinson
SOFTWARE DEVELOPER

@techiesimon www.simonrobinson.com

Overview

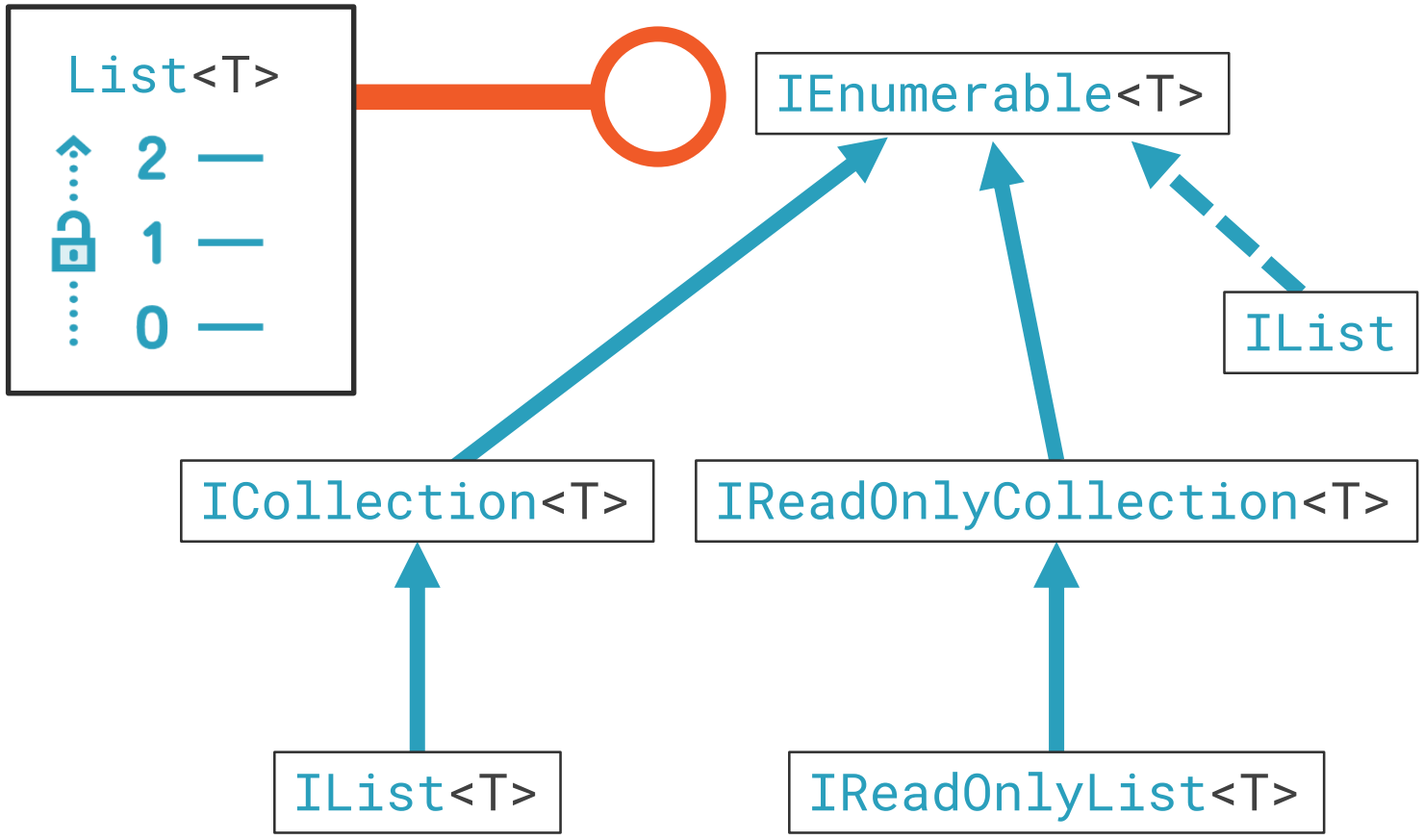
Coding with interfaces

- Decouples code
- Can make code easier to maintain

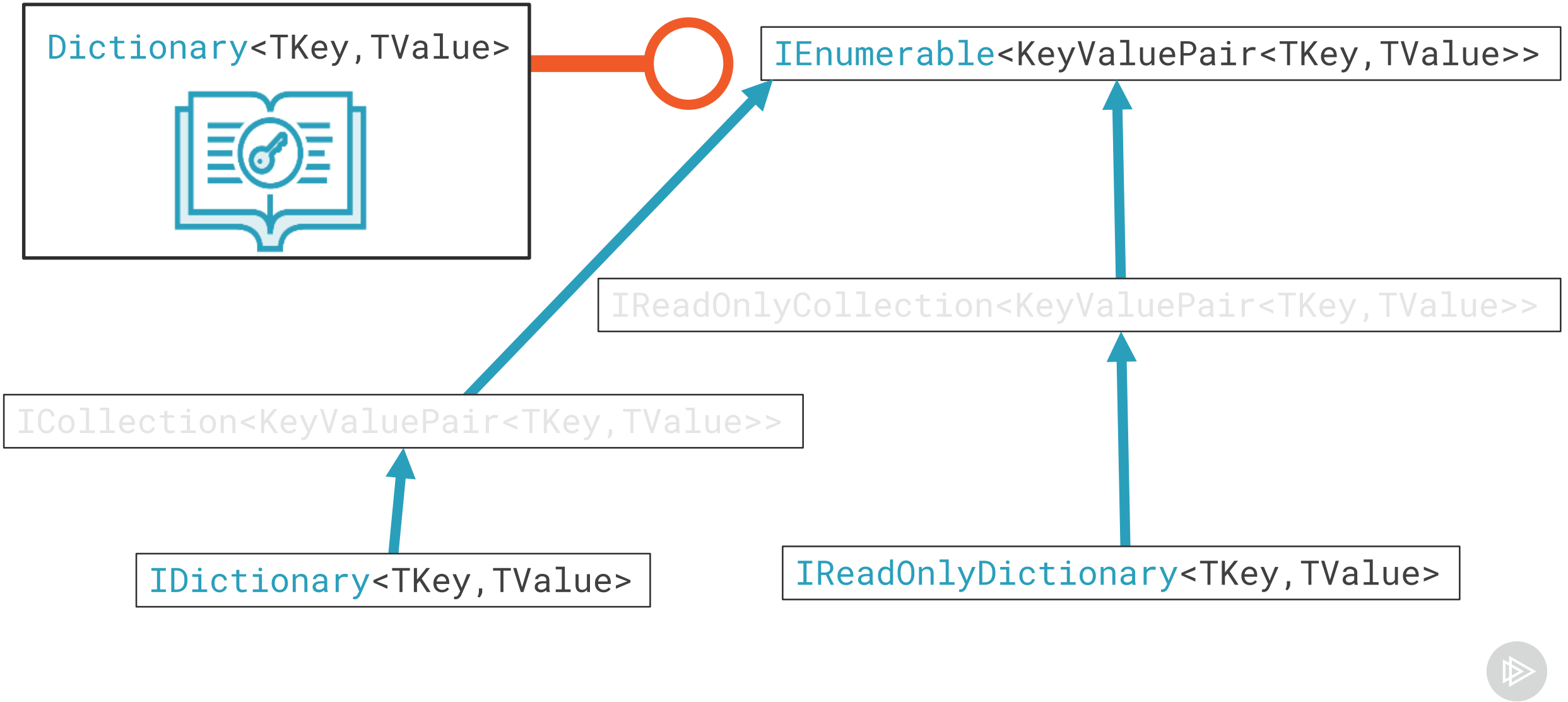
More efficient code using collection interfaces

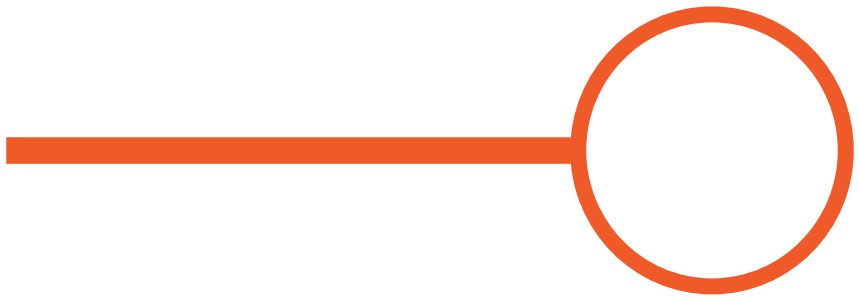
`IEnumerable<T>`

Some List Interfaces



Some Dictionary Interfaces

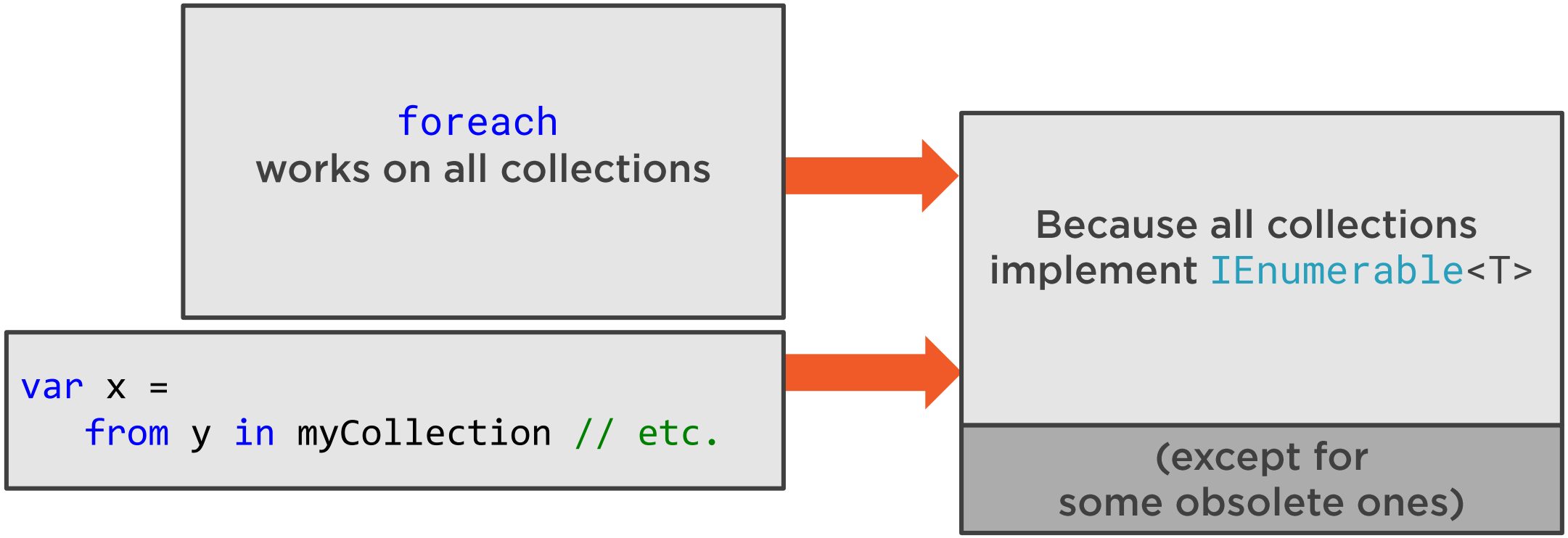




Collection interfaces

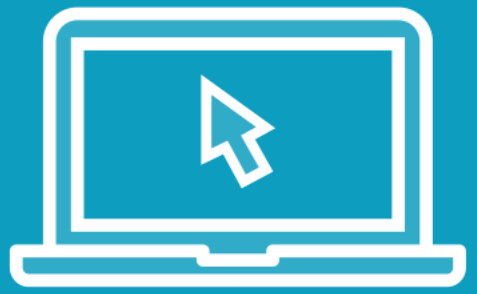
- Interfaces for list and dictionary
- Read-only and writable flavors
- All collections expose IEnumerable<T>

foreach


foreach
works on all collections
var x =
from y in myCollection // etc.
Because all collections
implement IEnumerable<T>
(except for
some obsolete ones)

Making Code Efficient with `IEnumerable<T>`

Demo



Method that returns a collection

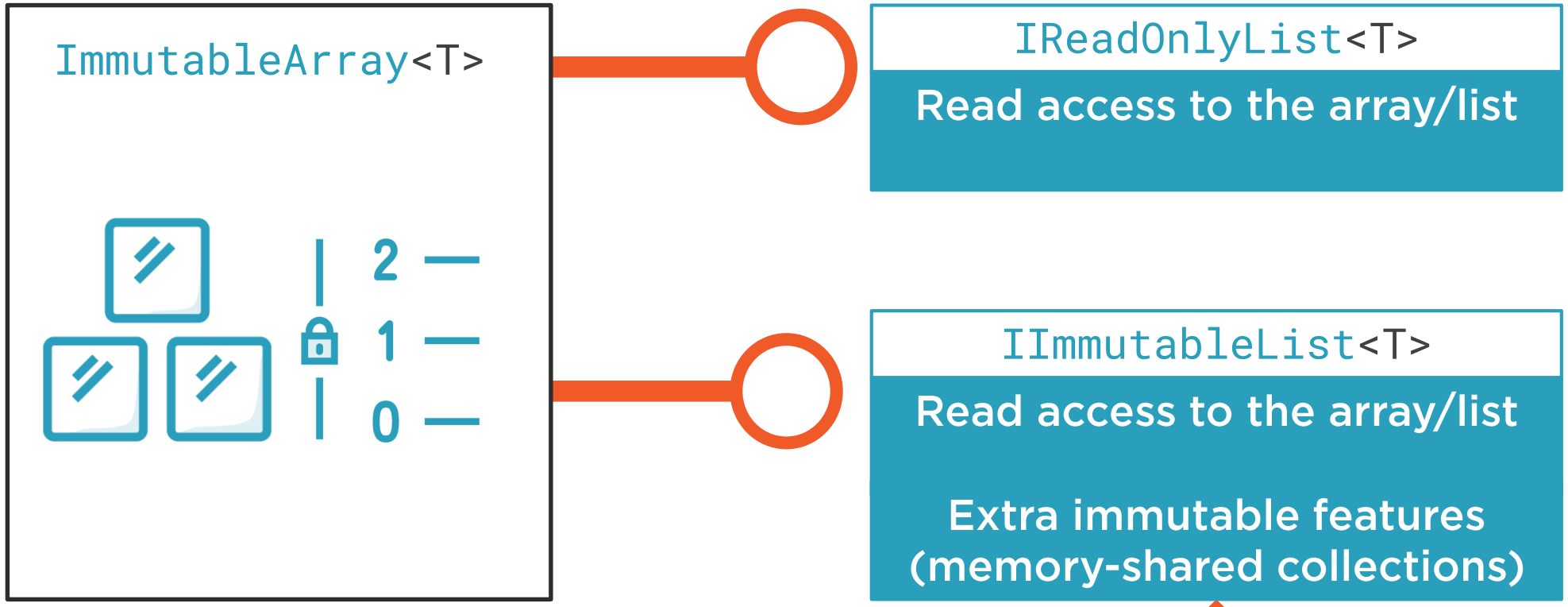
- Avoid building this collection
- But requires changing the return type to `IEnumerable<T>`

Decoupling Code with Interfaces

Demo

A collection property

- Declare the type as an interface
- Decouples declaration from instantiation type



Arrays and lists share the same interfaces

Decoupling code

Makes parts of the code less dependent on other parts of the code.

Decoupling types with interfaces makes code easier to modify and test

Module Summary

Interfaces

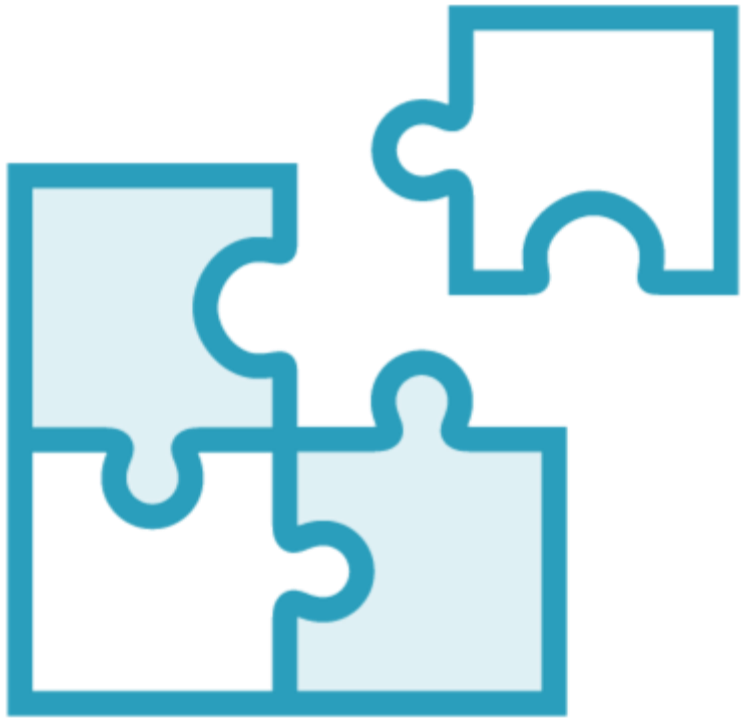
- Collections implement `IEnumerable<T>`
  - Allows enumeration with `foreach`, LINQ, etc.
- `IEnumerable<T>` sometimes gives performance benefits
  - Usually with LINQ
- Interfaces can decouple your code
  - Easier to maintain

Course Summary

Advanced C# Collections

- Lists, dictionaries, linked lists, stacks, queues, sets
- Collection scalability: O(n) notation
  - O(1) is the most scalable
- Comparers and equality comparers
- Concurrent collections
- Read-only and immutable collections

A Challenge



Thanks for watching!