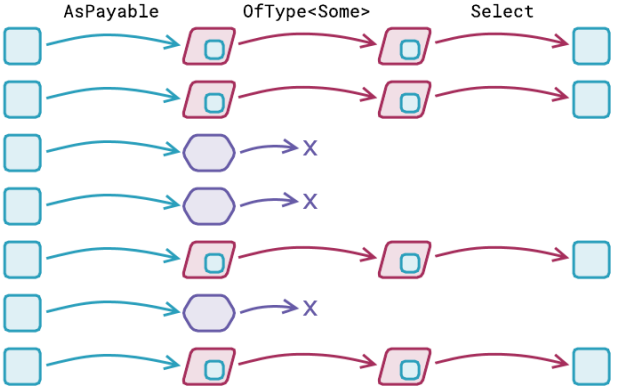
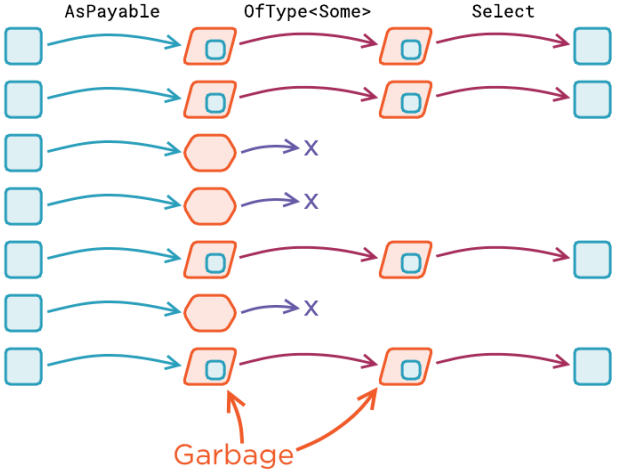
Treating Sequences as Immutable Objects

Zoran Horvat
CEO AT CODING HELMET
@zoranh75 http://csharpmentor.com

Understanding Trade-off with Sequences



Understanding Trade-off with Sequences



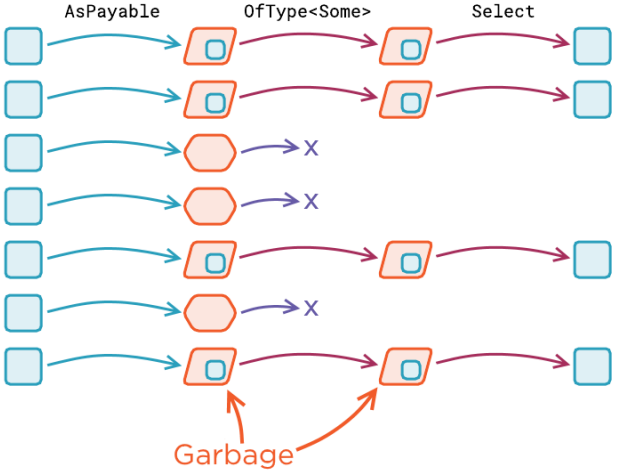
Trick #1:
Check if there is anything to do

```
f(seq) =>
  NotGonnaChange(seq)
    ? seq
    : Work(seq);
```

Return same reference if possible

But checking will cost, too!

Understanding Trade-off with Sequences



Trick #1:
Check if there is anything to do

```
f(seq) =>
  NotGonnaChange(seq)
    ? seq
    : Work(seq);
```

Trick #2:
Do nothing

```
seq.Select(UsefulWork);
```

Short code

Intention-revealing

Readable Sequence Mapping

```
public static IEnumerable<IMoney> PayableAt(
  this IEnumerable<IMoney> moneys, DateTime at) =>
  moneys.SelectOptional(money => money.PayableAt(at));
```

The sequence

Readable Sequence Mapping

```
public static IEnumerable<IMoney> PayableAt(
  this IEnumerable<IMoney> moneys, DateTime at) =>
  moneys.SelectOptional(money => money.PayableAt(at));
```

The sequence of money

Readable Sequence Mapping

```
public static IEnumerable<IMoney> PayableAt(
  this IEnumerable<IMoney> moneys, DateTime at) =>
  moneys.SelectOptional(money => money.PayableAt(at));
```

The sequence of money objects payable at

Readable Sequence Mapping

```
public static IEnumerable<IMoney> PayableAt(
  this IEnumerable<IMoney> moneys, DateTime at) =>
  moneys.SelectOptional(money => money.PayableAt(at));
```

The sequence of money objects payable at specified time

Readable Sequence Mapping

```
public static IEnumerable<IMoney> PayableAt(
  this IEnumerable<IMoney> moneys, DateTime at) =>
  moneys.SelectOptional(money => money.PayableAt(at));
```

The sequence of money objects payable at specified time is obtained

Readable Sequence Mapping

```
public static IEnumerable<IMoney> PayableAt(
  this IEnumerable<IMoney> moneys, DateTime at) =>
  moneys.SelectOptional(money => money.PayableAt(at));
```

The sequence of money objects payable at specified time is obtained by selecting

Readable Sequence Mapping

```
public static IEnumerable<IMoney> PayableAt(
  this IEnumerable<IMoney> moneys, DateTime at) =>
  moneys.SelectOptional(money => money.PayableAt(at));
```

The sequence of money objects payable at specified time is obtained by selecting optional

Readable Sequence Mapping

```
public static IEnumerable<IMoney> PayableAt(
  this IEnumerable<IMoney> moneys, DateTime at) =>
  moneys.SelectOptional(money => money.PayableAt(at));
```

The sequence of money objects payable at specified time is obtained by selecting optional money

Readable Sequence Mapping

```
public static IEnumerable<IMoney> PayableAt(
  this IEnumerable<IMoney> moneys, DateTime at) =>
  moneys.SelectOptional(money => money.PayableAt(at));
```

The sequence of money objects payable at specified time is obtained by selecting optional money objects payable at

Readable Sequence Mapping

```
public static IEnumerable<IMoney> PayableAt(
  this IEnumerable<IMoney> moneys, DateTime at) =>
  moneys.SelectOptional(money => money.PayableAt(at));
```

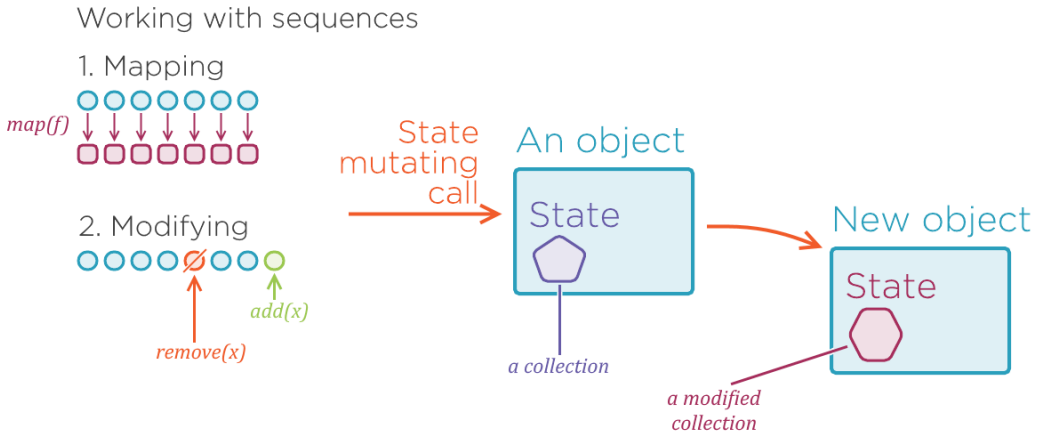
The sequence of money objects payable at specified time is obtained by selecting optional money objects payable at specified time

Readable Sequence Mapping

```
public static IEnumerable<IMoney> PayableAt(
  this IEnumerable<IMoney> moneys, DateTime at) =>
  moneys.SelectOptional(money => money.PayableAt(at));
```

The sequence of money objects payable at specified time is obtained by selecting optional money objects payable at specified time

Mutating Collections Inside Objects



Introducing Immutable Collections

Common collections

`System.Collections.Generic` namespace

- `List<T>`
- `Dictionary<TKey, TValue>`
- `HashSet<T>`
- `LinkedList<T>`
- `Queue<T>`
- `Stack<T>`
- `SortedSet<T>`

None supports efficient immutable manipulation

Introducing Immutable Collections

Common collections

`System.Collections.Generic` namespace

`List<T>`
`Dictionary<TKey, TValue>`
`HashSet<T>`
`LinkedList<T>`
`Queue<T>`
`Stack<T>`
`SortedSet<T>`

Wallet implementation

```
Wallet newWallet =
  wallet.With(currency);
```

Existing wallet object will not change

Introducing Immutable Collections

Common collections

`System.Collections.Generic` namespace

`List<T>`
`Dictionary<TKey, TValue>`
`HashSet<T>`
`LinkedList<T>`
`Queue<T>`
`Stack<T>`
`SortedSet<T>`

Wallet implementation

```
Wallet newWallet =
  wallet.With(currency);
```

Desired list implementation

```
List newList = list.Add(element);
```

Immutable implementation

Introducing Immutable Collections

Common collections

`System.Collections.Generic` namespace

`List<T>`
`Dictionary<TKey, TValue>`
`HashSet<T>`
`LinkedList<T>`
`Queue<T>`
`Stack<T>`
`SortedSet<T>`

Wallet implementation

```
Wallet newWallet =
  wallet.With(currency);
```

Desired list implementation

```
List newList = list.Add(element);
```

Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```

```
ImmutableList<T> Add(T value);
ImmutableList<T> Clear();
ImmutableList<T> Remove(T value);
...
```

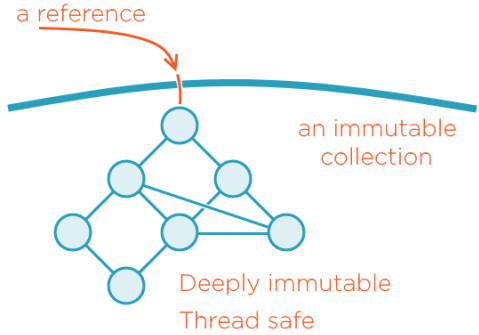
All mutating methods return a new collection of the same kind

Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```

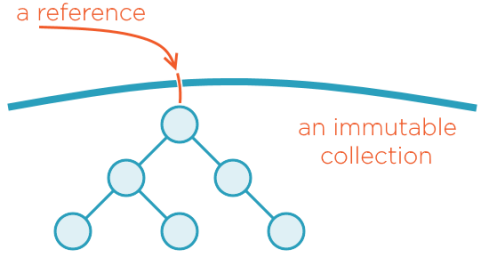


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```



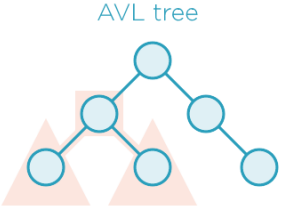
Mostly based on AVL tree
(Adelson, Velsky, Landis)

Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```



Balanced tree

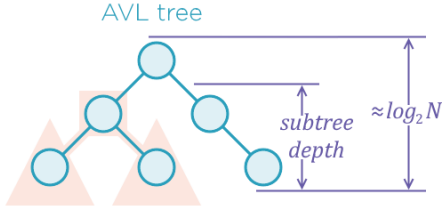
For any node in the tree, **size** of its left and right subtree must be balanced

Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

`ImmutableList<T>`
`ImmutableDictionary<TKey, TValue>`
`ImmutableHashSet<T>`
`ImmutableArray<T>`
`ImmutableQueue<T>`
`ImmutableStack<T>`
`ImmutableSortedSet<T>`
`ImmutableSortedDictionary<TKey, TValue>`



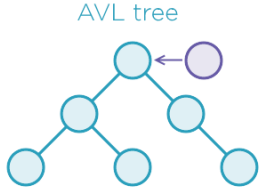
AVL tree
For any node in the AVL tree, **depths** of its left and right subtree must not differ by more than one

Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

`ImmutableList<T>`
`ImmutableDictionary<TKey, TValue>`
`ImmutableHashSet<T>`
`ImmutableArray<T>`
`ImmutableQueue<T>`
`ImmutableStack<T>`
`ImmutableSortedSet<T>`
`ImmutableSortedDictionary<TKey, TValue>`

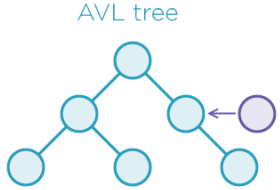


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

`ImmutableList<T>`
`ImmutableDictionary<TKey, TValue>`
`ImmutableHashSet<T>`
`ImmutableArray<T>`
`ImmutableQueue<T>`
`ImmutableStack<T>`
`ImmutableSortedSet<T>`
`ImmutableSortedDictionary<TKey, TValue>`

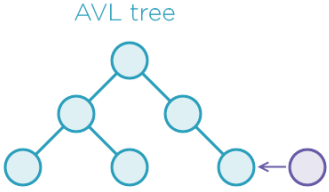


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

`ImmutableList<T>`
`ImmutableDictionary<TKey, TValue>`
`ImmutableHashSet<T>`
`ImmutableArray<T>`
`ImmutableQueue<T>`
`ImmutableStack<T>`
`ImmutableSortedSet<T>`
`ImmutableSortedDictionary<TKey, TValue>`

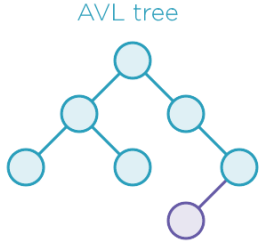


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```

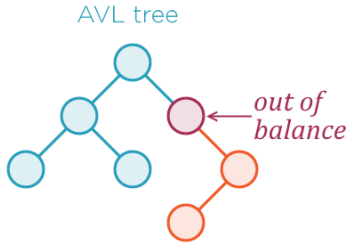


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

`ImmutableList<T>`
`ImmutableDictionary<TKey, TValue>`
`ImmutableHashSet<T>`
`ImmutableArray<T>`
`ImmutableQueue<T>`
`ImmutableStack<T>`
`ImmutableSortedSet<T>`
`ImmutableSortedDictionary<TKey, TValue>`



Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```

Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```

Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

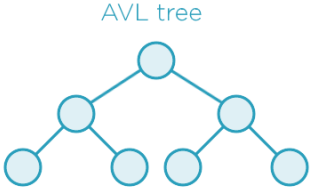
```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```

Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```

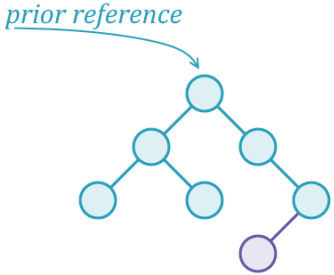


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```

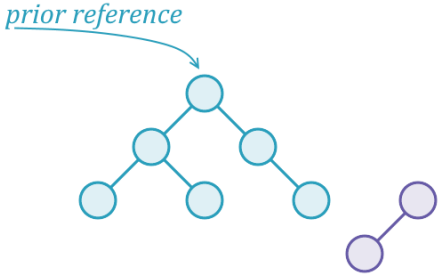


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

- `ImmutableList<T>`
- `ImmutableDictionary<TKey, TValue>`
- `ImmutableHashSet<T>`
- `ImmutableArray<T>`
- `ImmutableQueue<T>`
- `ImmutableStack<T>`
- `ImmutableSortedSet<T>`
- `ImmutableSortedDictionary<TKey, TValue>`

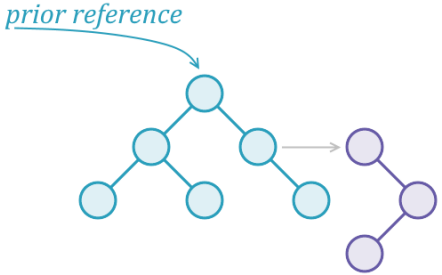


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```

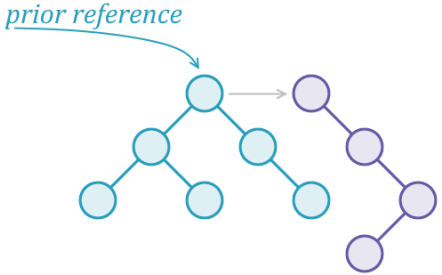


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```

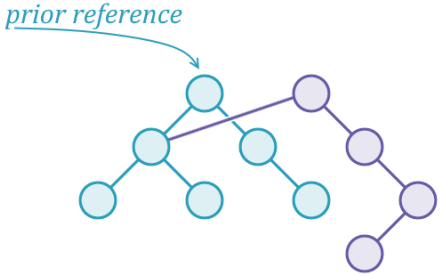


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

- `ImmutableList<T>`
- `ImmutableDictionary<TKey, TValue>`
- `ImmutableHashSet<T>`
- `ImmutableArray<T>`
- `ImmutableQueue<T>`
- `ImmutableStack<T>`
- `ImmutableSortedSet<T>`
- `ImmutableSortedDictionary<TKey, TValue>`

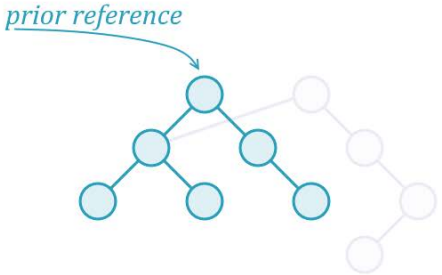


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```

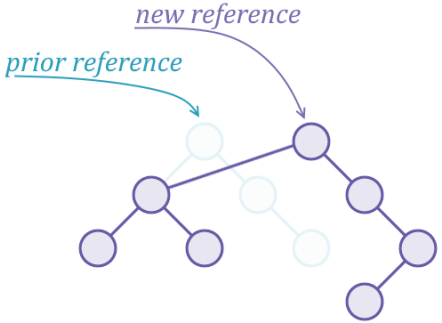


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```

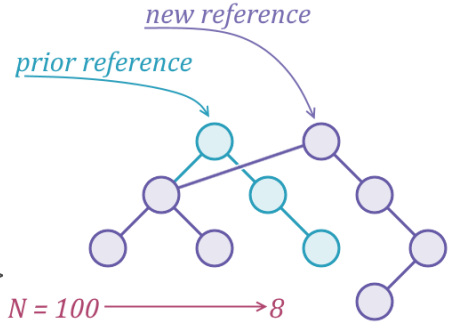


Introducing Immutable Collections

Immutable collections

`System.Collections.Immutable` namespace

```
ImmutableList<T>
ImmutableDictionary<TKey, TValue>
ImmutableHashSet<T>
ImmutableArray<T>
ImmutableQueue<T>
ImmutableStack<T>
ImmutableSortedSet<T>
ImmutableSortedDictionary<TKey, TValue>
```



Summary

Working with collections of objects

- Often reduced to a sequence
- `IEnumerable<T>` usually sufficient

Immutable collections

- `ImmutableList<T>` is most useful
- Corresponds to `IEnumerable<T>`

Summary

Principles of simple design

- Stick to linear sequences: `IEnumerable<T>` and `ImmutableList<T>`
- That leads to simple design

Cost of using immutable collections

- $O(logN)$ time and space complexity in most of them
- Implementation based on AVL tree
- Stack and queue don't need a tree

Next module:
Composing Functions into Larger Behavior