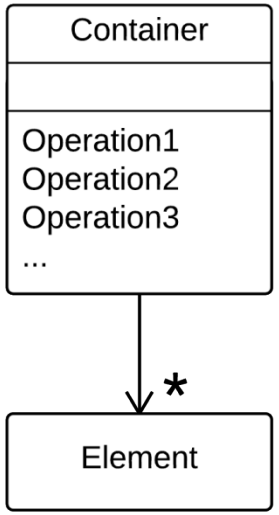
Iterator Pattern and Sequences

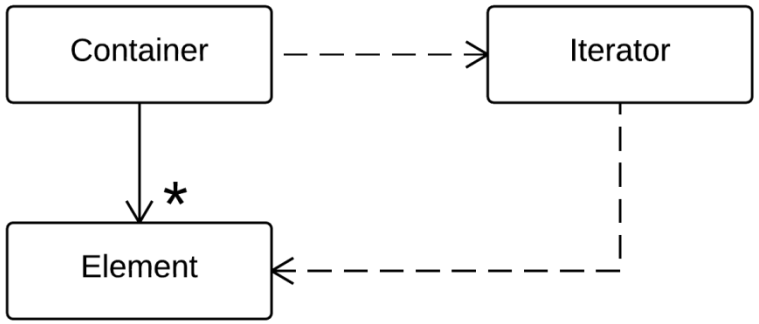
Zoran Horvat
CTO at InterVenture GmbH

@zoranh75 www.codinghelmet.com

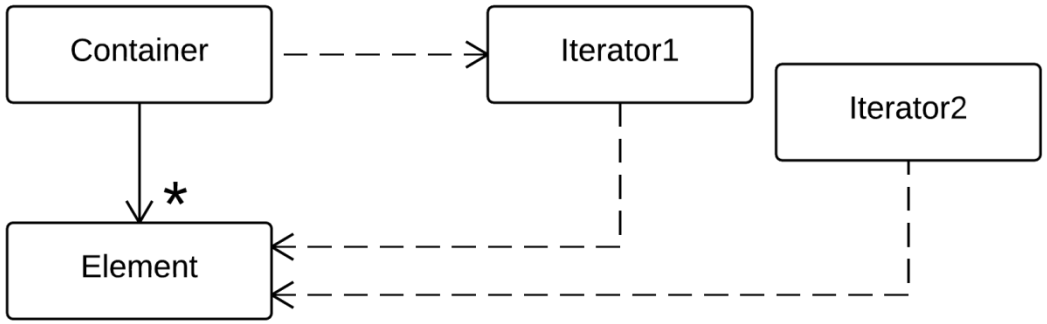
Iterator Design Pattern



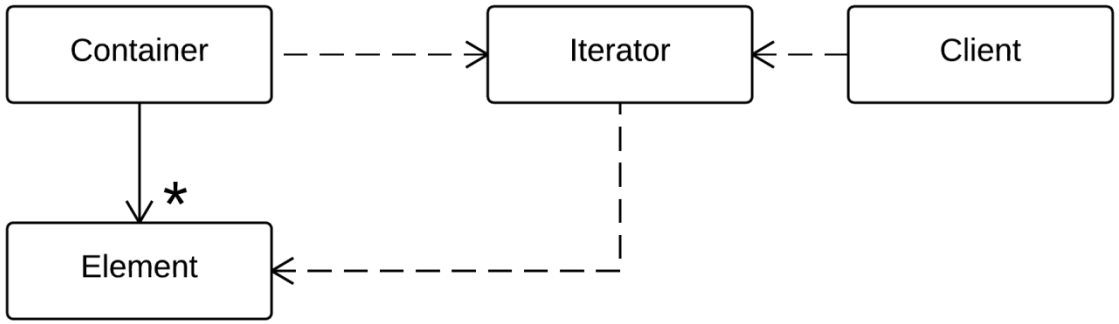
Iterator Design Pattern



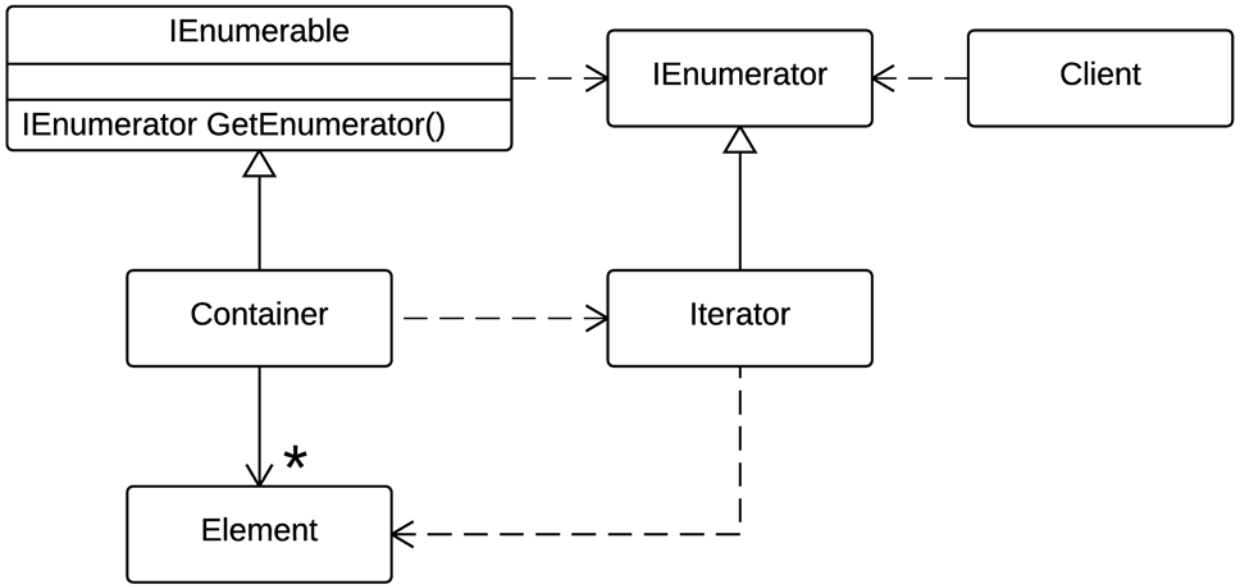
Iterator Design Pattern



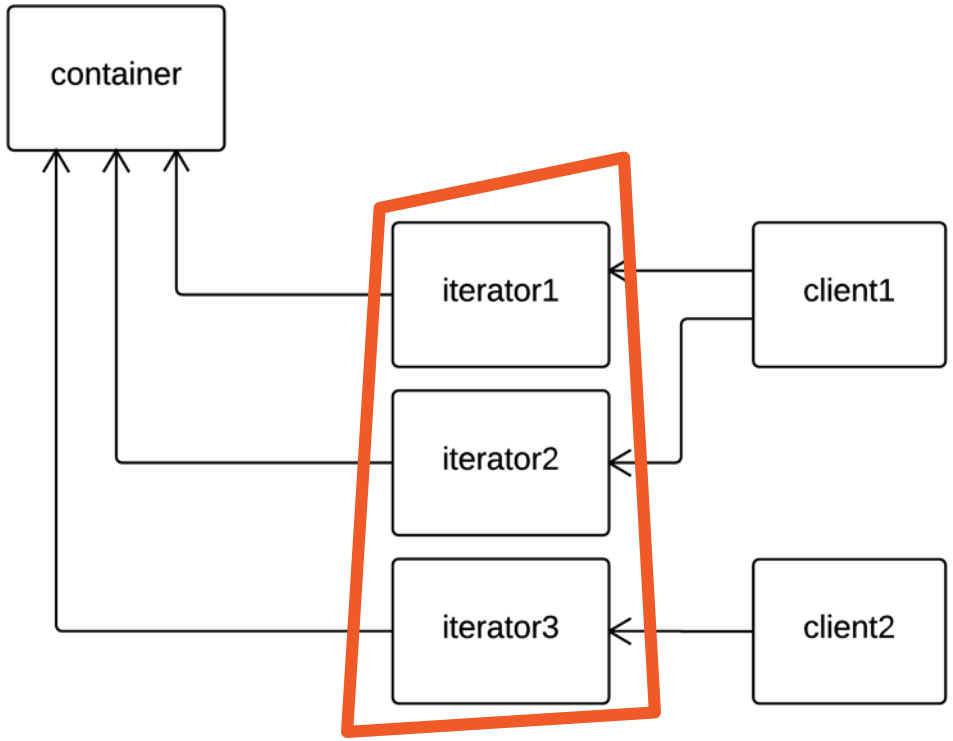
Iterator Design Pattern



Iterator Design Pattern

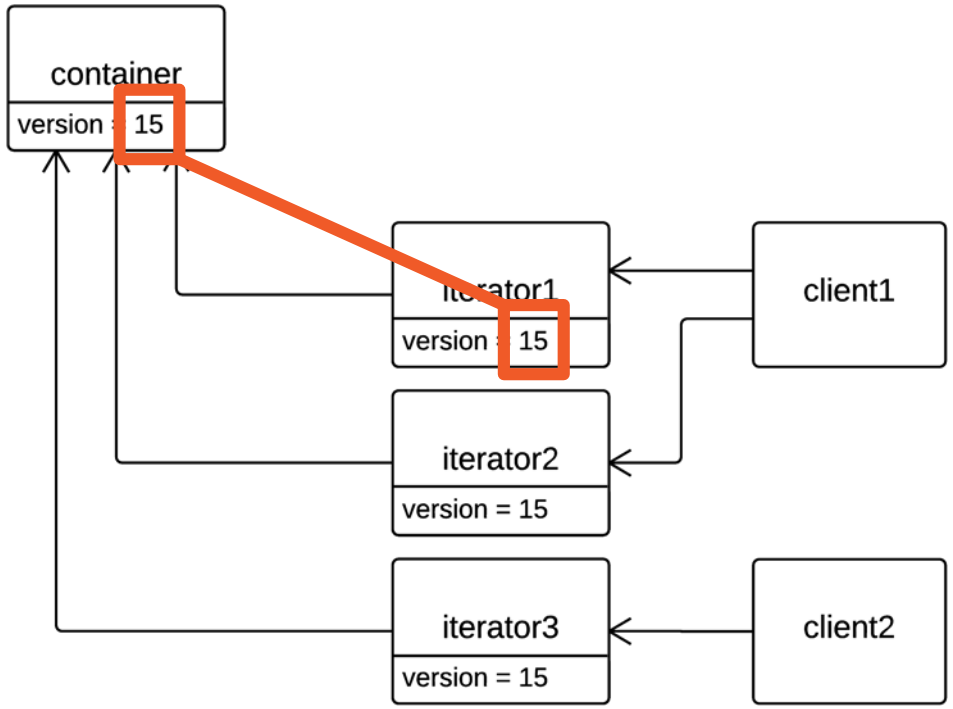


Simultaneous Iterations

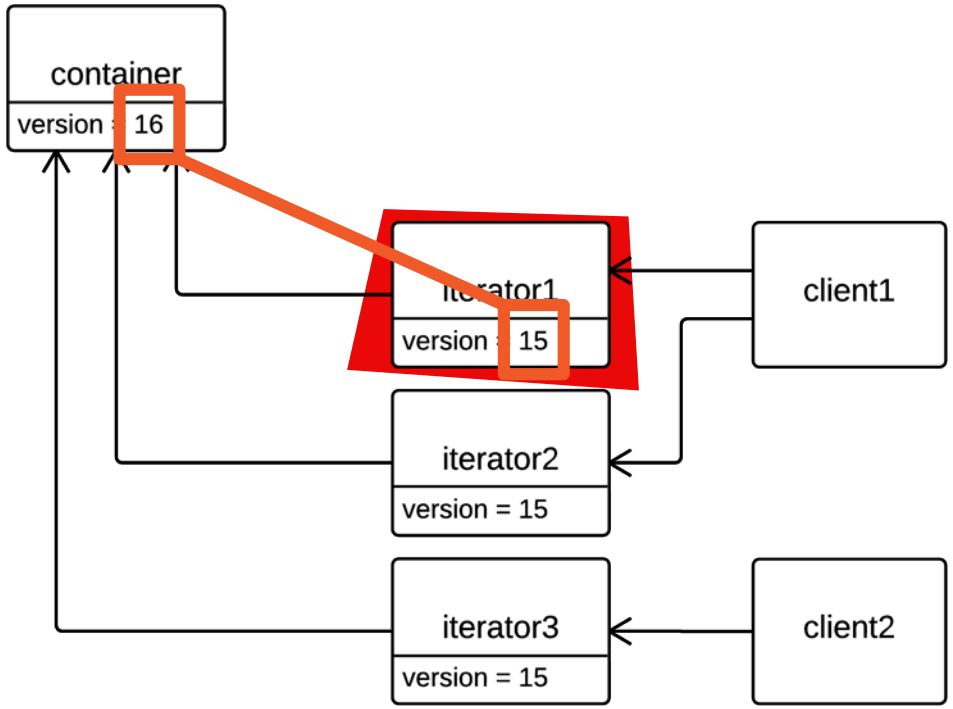


Iterators must be mutually independent

Simultaneous Iterations



Simultaneous Iterations



Summary

Iterator design pattern in .NET

- IEnumerable – container role
- IEnumerator – iterator role

yield keyword

- Produces sequences without having the underlying collection

Sequences

- Decouple collection from iteration
- If we don't use a concrete collection, then we can save a lot of memory
- The code may also become shorter and easier to understand

Summary

Infinite sequences

- When there is no underlying collection, sequence may run forever
- C# lets us process long sequences without using much memory

Collections and optional data

- No need to use null reference to indicate a missing object
- Use an empty collection instead
- Option<T> functional type
  - Sequence with zero or one element

In the following module:
Option<T> functional type