Iterator

Karoly Nyisztor
DEVELOPER
@knyisztor   www.leakka.com

Overview

Motivation

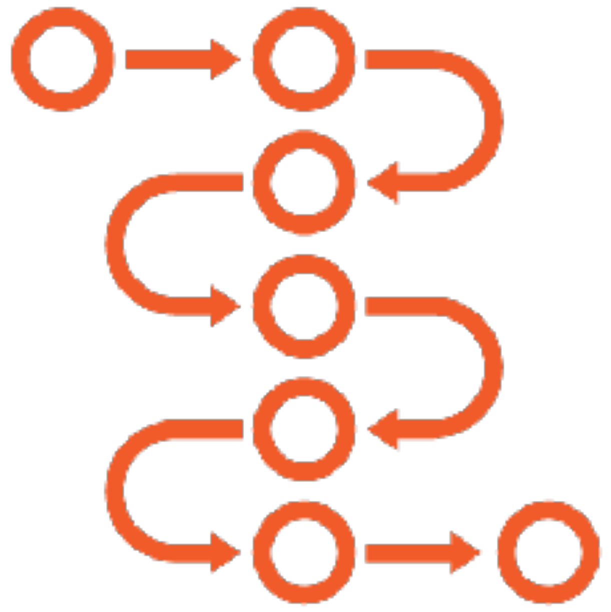
StackTraversal demo

- Add traversal capability to a custom collection

Custom Iterator demo

- Implement a custom iterator from scratch

Motivation



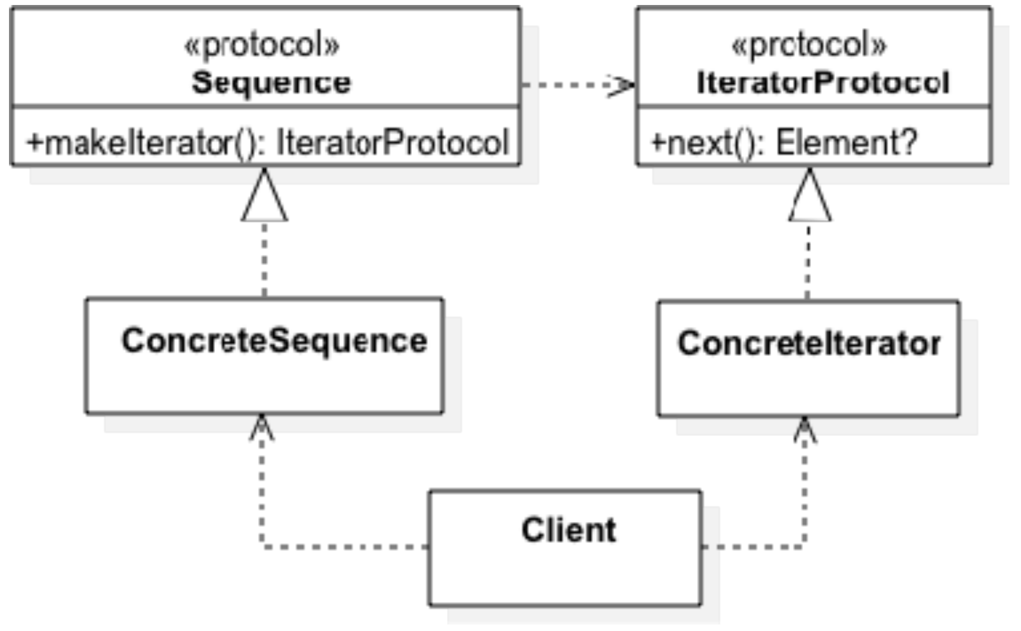
Traverse a container of objects

- Access elements sequentially
- Provide a uniform interface for iterating
- Avoid exposing container details

Iterator

Provides sequential access to the elements of a container without exposing its underlying details.

Iterator Design in Swift



Sequence protocol

- Defines requirements for sequential access to its elements

IteratorProtocol

- Provides a unified interface for accessing the elements of a sequence one at a time

```swift
let items = [1, 3, 7, 11]

for item in items {
    print(item)
}
```

◀ for - in loop

```swift
var iterator = items.makeIterator()

while let item = iterator.next() {
    print(item)
}
```

◀ behind the scenes

Demo

StackTraversal demo

- Add traversal capability to a custom collection

Demo

Custom Iterator demo

- Implement a custom iterator from scratch

Summary

The Iterator design pattern:

- Provides sequential access to the elements of a container
- Does not expose the internal structure of the traversed object
- Keeps track of the traversed elements

Pitfalls

- Consider the performance impact of your Iterator implementation