Iterator

Design Patterns

February 6th 2025

Stefan Kruik
Max Pijnacker
Donovan Tjokrodimedjo
Merlijn Bijenveld

Tables of contents

What is the Iterator pattern?

- The **Iterator Pattern** is one of the behavioral design patterns
- The pattern provides a way to access elements of a collection **sequentially** without exposing its underlying structure
- Instead of working with the list directly, the Iterator tells you what the next item is
- It is part of the **Java Iterator** interface in the java.util package

What does the Iterator pattern do?

- Provides a way to access elements in an **aggregate**
  - Aggregate can be any iterable object, like a database model
  - As long as there is a Iterator class for it that **implements** it. Like `StudentIterator`
- Abstracts the way you loop through the elements
  - **Generic** way of iterating through the collection
  - Instead of `forEach` or `for i`, you use `while` with `iterator.hasNext()` as stop-condition
  - Provides an uniform & clean way of looping
- Support for multiple iterators
  - Allows for **multiple** states

How does the Iterator pattern work?

- Iterator class implements the Iterator interface: `StudentIterator implements Iterator<Student>`
- Your model/class implements Aggregate interface: `Teacher implements Aggregate<Student>`
  - Aggregate interface holds the `createIterator()` method
- Implement and **override** its methods
  - `hasNext()`, `next()`
- Instead of using **conventional** loops, use the Iterator
  - Create a new instance of the iterator
  - Within the `while hasNext()` loop call `next()`

Examples of the Iterator pattern in code

- Live demonstration of an Iterator implementation

Sources

- https://www.geeksforgeeks.org/iterator-pattern/
  - Base implementation for normal classes
- https://www.digitalocean.com/community/tutorials/iterator-design-pattern-java
  - Iterate even over Enum types