The ***Iterator Pattern*** provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation. It also places the task of traversal on the iterator object, not on the aggregate, which simplifies the aggregate interface and implementation, and places the responsibility where it should be.

ConcreteIterator

hasNext()

next()

remove()

ConcreteAggregate

createAggregate()

<<interface>>

Iterator

hasNext()

next()

remove()

<<interface>>

Aggregate

createIterator()

Client

* An Iterator allows access to an aggregate’s elements without exposing its internal structure.
* An iterator takes the job of iterating over an aggregate and encapsulates it in another object.
* When using an iterator, we relieve the aggregate of the responsibility of supporting operations for traversing its data.
* An iterator provides a common interface for traversing the items of an aggregate, allowing you to use polymorphism when writing code that makes use of the items of the aggregate.