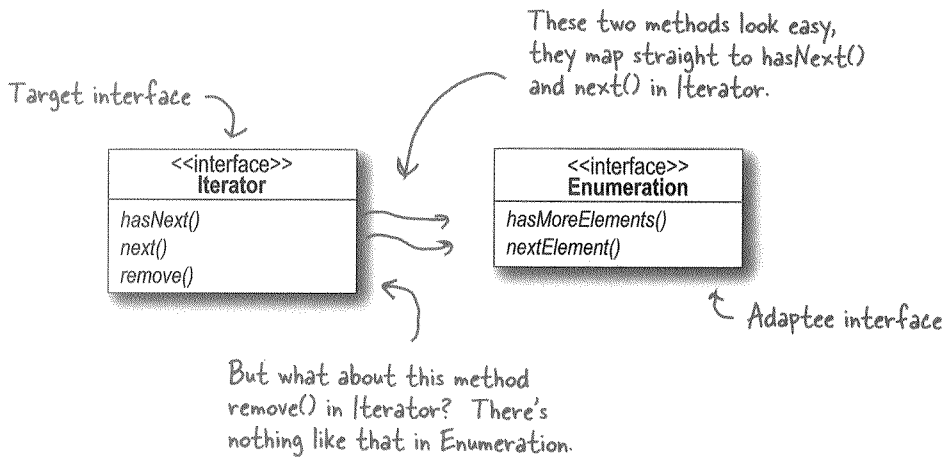# Adapting an Enumeration to an Iterator

First we'll look at the two interfaces to figure out how the methods map from one to the other. In other words, we'll figure out what to call on the adaptee when the client invokes a method on the target.

These two methods look easy, they map straight to hasNext() and next() in Iterator.

Target interface

<<interface>>
Iterator

hasNext()
next()
remove()

<<interface>>
Enumeration

hasMoreElements()
nextElement()

Adaptee interface

But what about this method remove() in Iterator? There's nothing like that in Enumeration.

Designing the Adapter

Here's what the classes should look like: we need an adapter that implements the Target interface and that is composed with an adaptee. The hasNext() and next() methods are going to be straightforward to map from target to adaptee: we just pass them right through. But what do you do about remove()? Think about it for a moment (and we'll deal with it on the next page). For now, here's the class diagram:

Your new code still gets to use Iterators, even if there's really an Enumeration underneath.

<<interface>>
Iterator

hasNext()
next()
remove()

We're making the Enumerations in your old code look like Iterators for your new code.

A class implementing the Enumeration interface is the adaptee.

EnumerationIterator is the adapter.

EnumerationIterator

hasNext()
next()
remove()

<<interface>>
Enumeration

hasMoreElements()
nextElement()