http://www.yegor256.com/2015/04/30/iterating-adapter.html

How to Implement an Iterating Adapter

30 April 2015 modified on 19 May 2015 Yegor Bugayenko

Iterator is one of the fundamental Java interfaces, introduced in Java 1.2. It is supposed to be very simple; however, in my experience, many Java developers don't understand how to implement a custom one, which should iterate a stream of data coming from some other source. In other words, it becomes an **adapter** of another source of data, in the form of an iterator. I hope this example will help.

Let's say we have an object of this class:

```
final class Data {
  byte[] read();
}
```

When we call `read()`, it returns a new array of bytes that were retrieved from somewhere. If there is nothing to retrieve, the array will be empty. Now, we want to create an *adapter* that would consume the bytes and let us iterate them:

```
final class FluentData implements Iterator<Byte> {
  boolean hasNext() { /* ... */ }
  Byte next() { /* ... */ }
  void remove()  { /* ... */ }
}
```

Here is how it should look (it is not thread-safe!):

```
final class FluentData implements Iterator<Byte> {
  private final Data data;
  private final Queue<Byte> buffer = new LinkedList<>();
  public FluentData(final Data dat) {
    this.data = dat;
  }
  public boolean hasNext() {
    if (this.buffer.isEmpty()) {
      for (final byte item : this.data.read()) {
        this.buffer.add(item);
      }
    }
    return !this.buffer.isEmpty();
  }
  public Byte next() {
    if (!this.hasNext()) {
      throw new NoSuchElementException("Nothing left");
    }
    return this.buffer.poll();
  }
  public void remove() {
    throw new UnsupportedOperationException("It is read-only");
  }
}
```

There is no way to make it thread-safe because the iterating process is outside the scope of the iterator. Even if we declare our methods as `synchronized`, this won't guarantee that two threads won't conflict when they both call `hasNext()` and `next()`. So don't bother with it and just document the iterator as not thread-safe, then let its users synchronize one level higher when necessary.