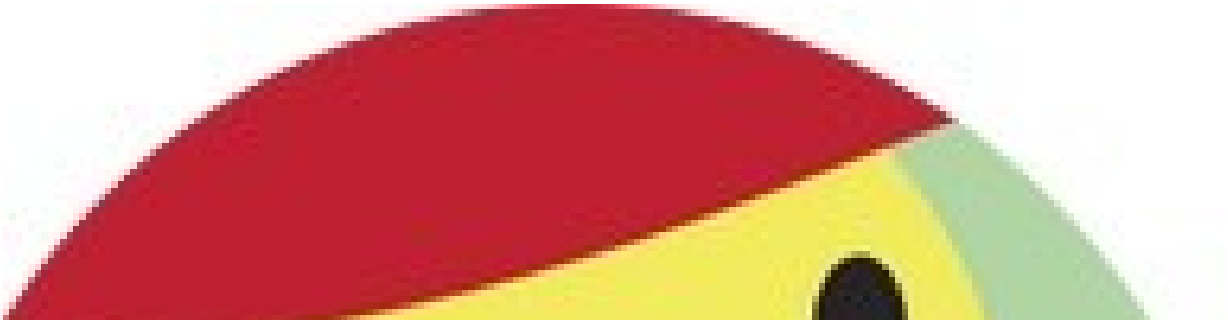
Community

yegor256.com

13 Comments • Created a month ago



How to Implement an Iterating Adapter

In my experience, very few people understand how to implement an iterating adapter in Java; this article should give a practical example.

(yegor256.com)

13 Comments

KillAByte_Garcia • a month ago

About thread-safe, you said: '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().' Do you mean that 'this.data.read()' can be called from two or more threads at the same time getting different parts of the main data-stream regardless of the fact that the method has been declared as syncronized?

Thanks for advance

Yegor Bugayenko author ↗ KillAByte_Garcia • a month ago

I mean that we don't know in what specific order will that threads call next() method of the iterator. So, there is no need to synchronize anything inside the iterator, it's better to let the client handle the synchronization.

KillAByte_Garcia ↗ Yegor Bugayenko • a month ago

Thanks a lot Yegor, it is much more clear now.

• Reply • Share ›

Oliver Doepner • a month ago

If speed matters, I would work directly with the byte arrays, an incrementing array index and comparison with array length. That is much faster than copying byte by byte first into the linked list and then polling them out.

• Reply • Share ›

David Raab ➚ Oliver Doepner • a month ago

I don't knew Java enough, but is that array even needed? Wether in C#, Perl, JavaScript or in any language that i have programmed so far i would never create a new list and fill it. It even nullifies the whole purpose of the Iterator. An Iterator just lazy loads data and only holds the current element in memory. If someone creates a completely new list then an Iterator makes not much sense to me.

That also saying. When i create an iterator i absolutely always would do what you described. just accessing that array with an index. For example in C# (well C# is far easy, you just need to use "yield" in a (static) method to get an IEnumerable (Iterator in C#) object)

C#:

```
static IEnumerable<byte> IterateByteArray(byte[] bytes) {
    foreach ( var byte in bytes ) {
        yield return byte;
    }
}
```

JavaScript:

```
function IterateByteArray(bytes) {
    var index = 0;
    return function() {
        if ( index >= bytes.length ) {
            return undefined;
        }
        return bytes[index++];
    }
}
```

None of this need to create a new list. Does not make sense to me to do that. That means is that a design-flaw of Java, or just a bad implementation of Yegor? Usually a iterator is not really so much slower. Well it always has a little bit more overhead, but it can also be faster in some circumstances. But often, the performance never really matter so much.

• Reply • Share ›

cfoley • a month ago

This will be OK if the client of your iterator uses it in the normal way: one call to next() in a loop. However, if the client decided to make several calls to next in a row then the iterator might fail if the buffer becomes empty. I'll agree that using the iterator in that way would be a little odd but you should still be able to navigate through the entire stream without ever calling hasNext().

Here is my suggested fix:

```
final class FluentData implements Iterator<Byte> {

  private Data data;
  private Queue<Byte> buffer = new LinkedList<>();

  public FluentData(Data data) {
    this.data = data;
  }

  public boolean hasNext() {
    refillBufferWhenEmpty();
    return !buffer.isEmpty();
  }

  public Byte next() {
    refillBufferWhenEmpty();
    return buffer.remove();
  }

  private void refillBufferWhenEmpty() {
    if (buffer.isEmpty())
      for (byte b : data.read())
        buffer.add(b);
  }

  public void remove() {
    throw new UnsupportedOperationException("FluentData is read-only");
  }

}
```

• Reply • Share ›

alexey semenyuk ➚ cfoley • a month ago

It's wrong. It isn't appropriate for an Iterator contract. The Next method should throw NoSuchElementException, when iteration has no more elements.

• Reply • Share ›

cfoley ➚ alexey semenyuk • a month ago

Thanks for the suggestion. It already throws the exception by calling the

remove() method on Queue, which also throws a NoSuchElementException when empty. Yegor's original implementation used poll() which returns null when it is empty which is why he needed to throw the exception himself.

https://docs.oracle.com/javase...

If I was using this in a real project, I would be developing by TDD and one of the tests would explicitly check for this exception. Do you think it would be better to add code to the method to make it more explicit?

▲ | ▼ • Reply • Share ›

alexey semenyuk ➚ cfoley • a month ago

I think that would be better to throw NoSuchElementException with a custom exception message like in an Yegor's implementation, this exception with a custom message is more specific and says more than NoSuchElementException from the Queue class.

▲ | ▼ • Reply • Share ›

cfoley ➚ alexey semenyuk • a month ago

That is a fair comment. Would you write a unit test for the message and another for which class is at the top of the stack trace?

▲ | ▼ • Reply • Share ›

alexey semenyuk • a month ago

You made a little mistake, your class called FluentData, but a constructor called DataIterator.

▲ | ▼ • Reply • Share ›

Yegor Bugayenko author ➚ alexey semenyuk • a month ago

Thanks, fixed.

1 ▲ | ▼ • Reply • Share ›

cfoley ➚ Yegor Bugayenko • a month ago

I see you also fixed the problem I mentioned. It's different from my suggestion and I like using the hasNext() as part of the solution.

▲ | ▼ • Reply • Share ›

More from yegor256.com

yegor256.com recently published

How to Avoid a Software Outsourcing Disaster

8 Comments Recommend

yegor256.com recently published

My Favorite Software Books - Yegor Bugayenko

13 Comments Recommend

yegor256.com recently published

A Few Thoughts on Unit Test Scaffolding

11 Comments Recommend