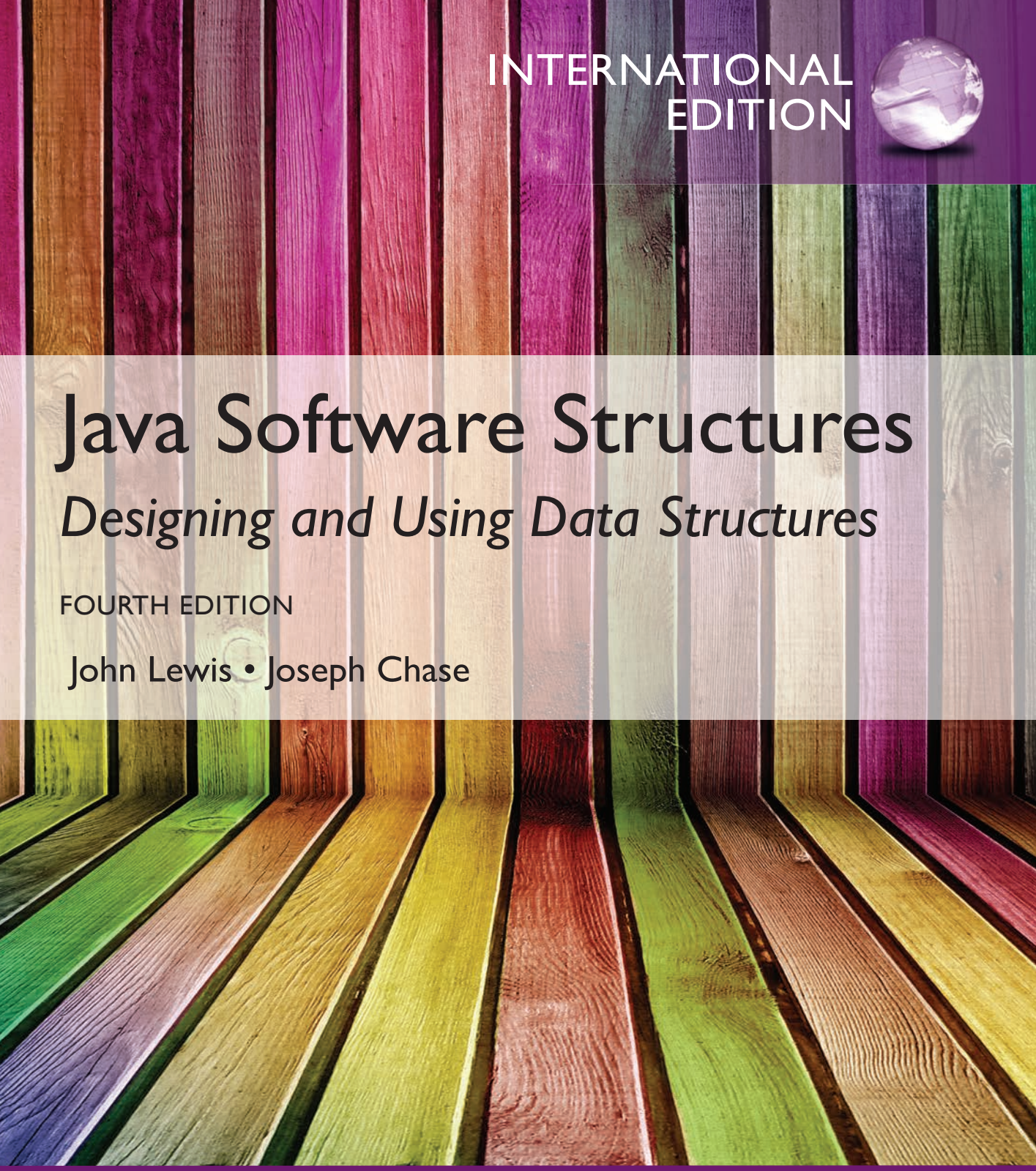

Java Software Structures

Designing and Using Data Structures

FOURTH EDITION

John Lewis • Joseph Chase

ALWAYS LEARNING

PEARSON

Java™ Software Structures

DESIGNING AND USING DATA STRUCTURES

4TH EDITION

LISTING 7.4 *continued*

```
     * @param modCount the current modification count for the ArrayList
     */
    public ArrayListIterator()
    {
        iteratorModCount = modCount;
        current = 0;
    }

    /**
     * Returns true if this iterator has at least one more element
     * to deliver in the iteration.
     *
     * @return  true if this iterator has at least one more element to deliver
     *          in the iteration
     * @throws  ConcurrentModificationException if the collection has changed
     *          while the iterator is in use
     */
    public boolean hasNext() throws ConcurrentModificationException
    {
        if (iteratorModCount != modCount)
            throw new ConcurrentModificationException();

        return (current < rear);
    }

    /**
     * Returns the next element in the iteration. If there are no
     * more elements in this iteration, a NoSuchElementException is
     * thrown.
     *
     * @return  the next element in the iteration
     * @throws  NoSuchElementException if an element not found exception occurs
     * @throws  ConcurrentModificationException if the collection has changed
     */
    public T next() throws ConcurrentModificationException
    {
        if (!hasNext())
            throw new NoSuchElementException();

        current++;

        return list[current - 1];
    }
```

LISTING 7.4 *continued*

```
    /**
     * The remove operation is not supported in this collection.
     *
     * @throws UnsupportedOperationException if the remove method is called
     */
    public void remove() throws UnsupportedOperationException
    {
        throw new UnsupportedOperationException();
    }
}
```

The `next` method returns the next element in the iteration and increments the `current` index value. If the `next` method is invoked and there are no elements left to process, then a `NoSuchElementException` is thrown.

In this implementation of the iterator, the `remove` operation is not supported (remember, it's considered optional). If this method is called, then an `UnsupportedOperationException` is thrown.

7.4 Implementing Iterators: With Links

Similarly, an iterator for a collection using links can also be defined. Like the `ArrayListIterator` class, the `LinkedListIterator` class (see Listing 7.5) is implemented as a private inner class. The `LinkedList` outer class maintains its own `modCount` that must stay in synch with the iterator's stored value.

In this iterator, though, the value of `current` is a pointer to a `LinearNode` instead of an integer index value. The `hasNext` method, therefore, simply confirms that `current` is pointing to a valid node. The `next` method returns the element at the current node and moves the `current` reference to the next node. As with our `ArrayListIterator`, the remove method is not supported.

LISTING 7.5

```
/**
 * LinkedListIterator represents an iterator for a linked list of linear nodes.
 */
private class LinkedListIterator implements Iterator<T>
{
```

LISTING 7.5 *continued*

```
private int iteratorModCount; // the number of elements in the collection
private LinearNode<T> current; // the current position

/**
 * Sets up this iterator using the specified items.
 *
 * @param collection the collection the iterator will move over
 * @param size       the integer size of the collection
 */
public LinkedListIterator()
{
    current = head;
    iteratorModCount = modCount;
}

/**
 * Returns true if this iterator has at least one more element
 * to deliver in the iteration.
 *
 * @return  true if this iterator has at least one more element to deliver
 *          in the iteration
 * @throws  ConcurrentModificationException if the collection has changed
 *          while the iterator is in use
 */
public boolean hasNext() throws ConcurrentModificationException
{
    if (iteratorModCount != modCount)
        throw new ConcurrentModificationException();

    return (current != null);
}

/**
 * Returns the next element in the iteration. If there are no
 * more elements in this iteration, a NoSuchElementException is
 * thrown.
 *
 * @return the next element in the iteration
 * @throws NoSuchElementException if the iterator is empty
 */
public T next() throws ConcurrentModificationException
{
    if (!hasNext())
        throw new NoSuchElementException();
```

LISTING 7.5 *continued*

```
        T result = current.getElement();
        current = current.getNext();
        return result;
    }

    /**
     * The remove operation is not supported.
     *
     * @throws UnsupportedOperationException if the remove operation is called
     */
    public void remove() throws UnsupportedOperationException
    {
        throw new UnsupportedOperationException();
    }
}
```

Summary of Key Concepts

- An iterator is an object that provides a way to access each element in a collection in turn.
- A collection is often defined as `Iterable`, which means it provides an `Iterator` when needed.
- The optional `remove` method of an iterator makes it possible to remove an element without having to traverse the collection again.
- Most iterators are fail-fast and will throw an exception if the collection is modified while an iterator is active.
- You should make no assumptions about the order in which an iterator delivers elements unless it is explicitly stated.
- An iterator class is often implemented as an inner class of the collection to which it belongs.
- An iterator checks the modification count to ensure that it stays consistent with the modification count from the collection when it was created.

Summary of Terms

iterator An object that allows the user to acquire and use each element in the collection one at a time.

fail-fast An iterator that throws an exception if its collection is modified in any way except through the iterator itself.

Self-Review Questions

SR 7.1 What is an iterator?

SR 7.2 What does the `Iterable` interface represent?

SR 7.3 What does the `Iterator` interface represent?

SR 7.4 What is the relationship between a for-each loop and iterators?

SR 7.5 Why might you need to use an explicit iterator instead of a for-each loop?

SR 7.6 What does it mean for an iterator to be fail-fast?

SR 7.7 How is the fail-fast characteristic implemented?

Exercises

EX 7.1 Write a for-each loop that prints all elements in a collection of `Item` objects called `inventory`. What is required for that loop to work?

EX 7.2 Write a while loop that uses an explicit iterator to accomplish the same thing as Exercise 7.1.

EX 7.3 Write a for-each loop that calls the `addBonus` method on each `SalaryAccount` object in a collection called `payroll`. What is required for that loop to work?

EX 7.4 Write a while loop that uses an explicit iterator to accomplish the same thing as Exercise 7.3.

Answers to Self-Review Questions

SRA 7.1 An iterator is an object that is used to process each element in a collection one at a time.

SRA 7.2 The `Iterable` interface is implemented by a collection to formally commit to providing an iterator when it is needed.

SRA 7.3 The `Iterator` interface is implemented by an interface and provides methods for checking for, accessing, and removing elements.

SRA 7.4 A for-each loop can be used only with collections that implement the `Iterable` interface. It is a syntactic simplification that can also be accomplished using an iterator explicitly.

SRA 7.5 You may need to use an explicit iterator rather than a for-each loop if you don't plan on processing all elements in a collection or if you may use the iterator's `remove` method.

SRA 7.6 A fail-fast iterator will fail quickly and cleanly if the underlying collection has been modified by something other than the iterator itself.

SRA 7.7 An iterator notes the modification count of the collection when it is created and, on subsequent operations, makes sure that that value hasn't changed. If it has, the iterator throws a `ConcurrentModificationException`.