

```

package utils;

import java.util.NoSuchElementException;

public interface Iterator<E> {
    boolean hasNext();
    E next() throws NoSuchElementException;
    void remove() throws IllegalStateException;
}

```

This code block defines the `iterator()` method, which returns a new `ArrayIterator` object to provide iterator functionality. By calling this method, you can obtain an iterator object for iterating over the elements in the `ArrayList`.

```

public class ArrayIterator implements Iterator<E> {
    private int index; // 当前位置

    public ArrayIterator() {
        index = 0;
    }

    public boolean hasNext() {
        return index < size();
    }

    public E next() throws NoSuchElementException {
        if (!hasNext()) {
            throw new NoSuchElementException();
        }
        E result = get(index);
        index++;
        return result;
    }

    public void remove() throws IllegalStateException {
        if (index == 0) {
            throw new IllegalStateException();
        }
    }
}

```

This code block defines an inner class `ArrayIterator` that implements the `Iterator<E>` interface to provide iterator functionality for the `ArrayList`. It includes a private field `index` to track the current iteration position, and a constructor that initializes `index` to 0. `ArrayIterator` implements the `hasNext()` method to check if there is a next element, the `next()` method to retrieve the next element and move the pointer to the next position, and the `remove()` method to remove the previously accessed element. These methods allow users to iterate over the elements in the

ArrayList using the ArrayIterator and perform corresponding operations.

```
private class LinkedIterator implements Iterator<E> {
    private Node<E> current = head;

    @Override
    public boolean hasNext() {
        return current != null;
    }

    @Override
    public E next() {
        if (!hasNext()) {
            throw new NoSuchElementException();
        }
        E item = current.item;
        current = current.next;
        return item;
    }

    @Override
    public void remove() {
        // Implement if needed
        throw new UnsupportedOperationException();
    }
}
```

This code block shows the implementation of the iterator() method in the LinkedList class, which returns a new LinkedIterator object. The LinkedIterator class is an inner class within LinkedList that implements the Iterator<E> interface. It contains a current field to keep track of the current node during iteration.

The LinkedIterator class provides the following methods:

hasNext(): Returns true if there are more elements in the linked list.

next(): Returns the next element in the linked list and moves the iterator to the next position.

remove(): This method is not implemented and throws an UnsupportedOperationException. This indicates that removal during iteration is not supported.

By implementing the iterator() method and the LinkedIterator class, the LinkedList class now supports iteration over its elements using the enhanced for loop or explicitly with an iterator.


Result:

Finished after 0.411 seconds

Runs: 6/6

Errors: 0

Failures: 0

>  JUNIT05QQListIteratorTest [Runner: JUnit 5] (0.132 s)

⬇

⬆

✖

📄

🔒

🔄

🔍

📄

⌵

⋮