How does the Iterator interface work?

The `Iterator` interface provides a standard way to traverse a collection of objects one by one. It acts like a cursor that moves through the collection.[1] [2]

It has three primary methods:

1. `hasNext()`: Returns `true` if there are more elements to visit.
2. `next()`: Returns the next element and advances the cursor. Throws `NoSuchElementException` if no elements remain.
3. `remove()`: Removes the last element returned by the iterator from the underlying collection. This is the **only safe way** to modify a collection while iterating over it (to avoid `ConcurrentModificationException`).[3] [4]

Usage Example:

```
List<String> list = new ArrayList<>(List.of("A", "B", "C"));
Iterator<String> it = list.iterator();

while (it.hasNext()) {
    String item = it.next();
    if (item.equals("B")) {
        it.remove(); // Safely removes "B"
    }
}
```

Difference between Iterator and ListIterator

Feature	Iterator	ListIterator
Scope	Works with **all** Collections (`List`, `Set`, `Queue`).	Works **only** with `List` implementations (`ArrayList`, `LinkedList`). [5]
Direction	**Forward only** (`next()`).	**Bi-directional** (`next()` and `previous()`). [5]
Operations	Can only **read** and **remove**.	Can **read**, **remove**, **add**, and **replace** (`set()`).
Index Access	No index access.	Can get current index (`nextIndex()`, `previousIndex()`).
Start Point	Always starts at the beginning.	Can start iteration from any index (`list.listIterator(int index)`).

Interview Summary:
"I use `Iterator` for standard loops and safe removals across any collection type. I typically switch to `ListIterator` only when I need to traverse a **List backwards** or need to **modify/replace** elements at specific indices during iteration."

⁂

1. https://www.geeksforgeeks.org/java/java-implementing-iterator-and-iterable-interface/
2. https://www.programiz.com/java-programming/iterator
3. https://www.geeksforgeeks.org/java/iterator-interface-in-java/
4. https://www.w3schools.com/java/java_iterator.asp
5. https://jenkov.com/tutorials/java-collections/iterator.html
6. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/429036f4-3bc3-4a7a-8c6f-90246e73b9b9/image.jpg
7. https://course.ccs.neu.edu/cs2510sp18/lecture25.html
8. https://gist.github.com/jnwhiteh/68d095c630dfcaddffd1
9. https://stackoverflow.com/questions/5849154/can-we-write-our-own-iterator-in-java
10. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/24f18c0a-b56e-4862-a619-e2959036a5c2/image.jpg
11. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/97e9ed86-ed11-4035-a18b-1eb05c1bab4e/image.jpg
12. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/87f62423-96f2-4071-9802-8f6699e0ecd8/image.jpg
13. https://www.youtube.com/watch?v=aVoXoZqwoGE
14. https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html