

# Java Interview Question and Answers

## ➤ Topic – Iterator:

### Iterator Interview Questions with Answers

---

#### 1. What are Iterator or cursor in Java?

Answer: An iterator in Java is a special type of object that provides sequential (one by one) access to the elements of a collection object. It is introduced in Java 1.2 Collections Framework.

We use iterator in the Collections Framework to retrieve elements sequentially (one by one). It is called a universal Iterator or cursor.

#### 2. What are the types of Iterators in Java?

Answer: There are four types of iterators or cursors available in Java. They are as:

- Enumeration
- Iterator
- ListIterator
- Spliterator

#### 3. What is Iterable Interface in Java?

Answer: The Collection interface extends Iterable interface that is present at the top of the collection hierarchy. The iterable interface is present in `java.lang.Iterable` package .

It provides a uniform way to retrieve the elements one by one from a collection object.

---

#### 4. The iterable interface provides which method in Java?

Answer: The iterable interface provides just one method named `iterator()` which returns an instance of Iterator and provides access one by one to the elements in the collection object.

## 5. How to create Iterator object in Java?

Answer: We create Iterator object by calling `iterator()` method which is present in the `Iterable` interface. The general syntax for creating Iterator object is as follows:

a) `Iterator itr = c.iterator();` // c is any collection object.

b) `Iterator<Type> itr = c.iterator();` // Generic type.

For example:

```
Iterator<String> itr = c.iterator();
```

```
Iterator<Integer> itr = c.iterator();
```

## 6. How Java Iterator works internally?

Answer: Go to this tutorial to get the best answer with diagram: [Iteraor in Java](#)

## 7. What are Iterator methods provided by Iterator interface in Java?

Answer: The Iterator interface provides three methods in Java. They are as:

- (a) **public boolean hasNext()**: Return true if the iteration has more elements to traverse (iterate) in the forward direction.
- (b) **public Object next()**: Return next element in the collection. It will throw `NoSuchElementException` when the iteration is complete.
- (c) **public void remove()**: Removes the last or the most recent element returned by the iterator.

## 8. What are the advantages of iterating elements of collections using iterator?

Answer: Java Iterator provides the following advantages. They are:

- An iterator can be used with any collection classes.
- We can perform both read and remove operations.
- It acts as a universal cursor for collection API.

## 9. What are the limitation of Iterator in Java?

Answer: Iterator has the following limitations or drawbacks. They are:

- By using Iterator, we can iterate elements of collection only in the forwarding direction. We cannot iterate in the backward direction. Hence, these are called single-direction cursors.
- We can perform either read operation or remove operation.
- We cannot perform the replacement of new objects.

---

## 10. What is fail-fast iterator?

Answer: When an Iterator throws a ConcurrentModification exception while iterating a collection, it is called fail-fast iterator in Java.

It throws a ConcurrentModification exception under either of the following conditions:

- (a) Fail Fast iterator throws a ConcurrentModificationException if the underlying collection is structurally modified while iterating over it. Structural modification is an operation that adds or deletes one or more elements in the collection.
- (b) In a multi-threading environment, if one thread is attempting to modify a collection while another thread is iterating over it.
- (c) When fail-fast iterator uses an original collection to traverse over elements of collection.

Some examples of fail-fast iterators are HashMap, ArrayList, Vector, HashSet, etc.

---

## 11. What is fail-safe Iterator?

Answer: When an Iterator doesn't throw any exception while modifying the collection during iteration, it is called fail-safe iterator.

Fail-safe iterator makes the copy of the original collection to traverse over the elements. Thus, the original collection remains structurally unchanged.

Some examples of fail-safe iterators are ConcurrentHashMap, CopyOnWriteArrayList, etc.

---

## 12. What is the difference Fail-fast Iterator and Fail-safe Iterator in Java?

Answer: The fundamental difference between Fail-fast and Fail-safe Iterator is that the Fail-safe does not throw any `ConcurrentModificationException` in modifying the collection object during the iteration.

While fail-fast Iterator throws an exception in such scenarios. This is because fail-safe iterator works on a clone of original collection.

There are also several other differences between them based on different parameters. Let's know them:

### (a) **Exception:**

- Fail-fast iterator throws a `ConcurrentModificationException` in modifying the collection object during the iteration.
- While fail-safe iterator does not throw any kind of exception.

### (b) **Clone of collection object:**

- Fail-fast iterator does not create a clone of collection object during iteration process. It works on the original collection.
- While fail-safe iterator creates a copy or clone of collection object during the iteration process.

### (c) **Memory utilization:**

- Fail-fast requires low memory during the process.
- While fail-safe requires more memory during the process.


### (d) **Modification:**

- Fail-fast iterator does not allow modification during iteration.
- While fail-safe iterator allows modification during the iteration process.

### (e) **Performance:**

- Fail-fast iterator is fast.
- While fail-safe iterator is slightly slower than Fail Fast.

### (f) **Examples:**

- The fail-fast iterator returned by classes are `HashMap`, `ArrayList`, `Vector`, `HashSet`, etc.
  - The fail-safe iterator returned by classes are `CopyOnWriteArrayList`, `ConcurrentHashMap`, etc.
- 

### 13. Explain different ways to iterate over collection in Java.

Answer: There are many ways to iterate over elements of a collection in Java. Following are the most common methods.

**(a) Using enhanced For loop:** The following code is an example.

```
Collection<String> collection = Arrays.asList("AA", "BB", "CC", "DD", "EE");

for(String s : collection) {

    System.out.println(s);

}
```

**(b) Using Iterator method:** The following code is an example.

```
Collection<String> collection = Arrays.asList("One", "Two", "Three", "Four",
"Five");

Iterator<String> itr = collection.iterator();

while(itr.hasNext()) {

    System.out.println(itr.next());

}
```

**(c) Using Simple For loop:** A sample code is as:

```
List<String> list = Arrays.asList("One", "Two", "Three", "Four");

for( int i = 0; i < list.size(); i++ )

{

    System.out.println(list.get(i));

}
```

**(d) Using forEach method:** An example code is as:

```
Collection<String> collection = Arrays.asList("Apple", "Orange", "Banana",
"Guava");

collection.forEach(s -> System.out.println(s));
```

This is recently introduced in Java 8. It can be invoked on any Iterable and takes one argument implementing the functional interface `java.util.function.Consumer`.

---

**14. Suppose that you are iterating over a collection using Iterator. Can you remove an element from the collection using its Iterator? If yes, how to?**

Answer: Iterator interface provides a method named `remove()` that removes the last element that was returned by the `next` method.

The following example code will remove "Two" from the List using Iterator.

```
List list = new ArrayList();

list.add("One");

list.add("Two");

list.add("Three");

Iterator iterator = list.iterator();

while(iterator.hasNext())
{
    String str = iterator.next();

    if(str.equals("Two"))

        iterator.remove();
}
```

**15. Difference between Enumeration and Iterator.**

Answer: Both are useful in retrieving elements from a collection. But the major difference between Enumeration and Iterator is regarding functionality.

- (a) By using an enumeration, we can perform only read access, but using an Iterator, we can perform both read and remove operation.
- (b) Iterator is fail-fast while enumeration is not fail-fast.
- (c) Iterator is slower than enumeration.
- 

### 16. What is ListIterator in Java?

Answer: ListIterator is the most powerful iterator or cursor. It is a bi-directional cursor that retrieve the elements from a collection object in both forward and reverse directions. It is an interface.

### 17. How to create ListIterator object in Java?

Answer: We can create a ListIterator object by calling listIterator() method of the List interface. The general syntax for creating ListIterator object is as:

```
ListIterator<Type> litr = l.listIterator(); // l is any list object and Type is type of objects being iterated.
```

For example:

```
ListIterator<String> litr = l.listIterator();
```

### 18. How Java ListIterator works internally?

Answer: Refer to this tutorial for better answer: [ListIterator in Java](#)

---

### 19. What are the advantages of ListIterator in Java?

Answer: ListIterator provides several advantages. They are:

- ListIterator supports many operations, such as read, remove, replacement, and the addition of new objects.
- Using ListIterator, we can perform Iteration in both forward and backward directions.
- Methods of ListIterator are easy to use.

### 20. What are the limitations of ListIterator in Java?

Answer: ListIterator is the most powerful cursor, but it still has some limitations. They are as:

- ListIterator is applicable only for the list implemented class objects. Therefore, it is not a universal Java cursor.
  - It is not applicable to whole collection API.
- 

## **21. What is the fundamental difference between Iterator and ListIterator in Java?**

Answer: The major difference between Iterator vs ListIterator in Java is as:

- (a) Iterator is applicable to the whole Collection API. While ListIterator is only applicable for List implemented classes such as ArrayList, CopyOnWriteArrayList, LinkedList, Stack, Vector, etc.
  - (b) It is a Universal Iterator. While ListIterator is not a Universal Iterator.
  - (c) Iterator supports only forward direction iteration. While ListIterator supports both forward and backward direction iterations.
  - (d) It supports only read and delete operations. While ListIterator supports all the operations such as read, remove, replacement, and the addition of the new elements.
  - (e) It is known as a uni-directional iterator. While ListIterator is also known as bi-directional iterator.
- 

## **22. Can we modify the collection structure while iterating using for-each loop?**

Answer: While executing a for-each loop, we cannot modify the collection.

## **23. Why is Iterator considered more thread safe?**

Answer: Iterator is considered being more thread safe because it throws exception if we modify the collection while iterating.

## **24. Can we modify the collection structure while iterating using an Iterator?**

Answer: It is not permissible for one thread to modify the underlying collection structure while another thread is iterating over it.

## **25. How to avoid ConcurrentModificationException while iterating a collection?**



Answer: To avoid ConcurrentModificationException while iterating a collection, use concurrent collection class such as CopyOnWriteArrayList instead of ArrayList.

---

In this tutorial, we have listed the top 25 **Iterator interview questions in Java** with the best possible answers. Hope that you will have understood all the answers of the above Iterator questions and practiced them. All the best!!!

