Interview Questions
Collection Framework - 1

1. Iterator Interface

1. How does Java's Iterator prevent concurrent modification?
2. What is the difference between fail-fast and fail-safe iterators? Give examples.
3. How does ConcurrentModificationException occur? Provide a code example that causes this exception.
4. Can we modify a collection while iterating? If yes, how? If no, why not?
5. Why doesn’t Iterator have a hasPrevious() method like ListIterator?
6. How does the forEachRemaining() method in Iterator work?
7. Can we use an Iterator to iterate over a Map? If yes, how?
8. Why does Iterator.remove() method not work on all collections?
9. What is a weakly consistent iterator? How is it different from a fail-fast iterator?
10. Implement a custom iterator for a linked list-like data structure.

2. Collection Interface

1. Why is the Collection interface generic?
2. How does retainAll(), removeAll(), and containsAll() work internally?
3. What is the difference between Collection.remove(Object o) and Iterator.remove()?
4. Can a Collection store heterogeneous objects? If yes, what are the implications?
5. What is the best way to remove multiple elements from a collection?
6. How can we make a Collection read-only?
7. How does Collections.synchronizedCollection() work?
8. What is the difference between unmodifiableCollection() and synchronizedCollection()?
9. How does contains(Object o) work internally for different collections?
10. Implement a custom collection class that supports iteration using Iterator.

3. ArrayList

1. How does ArrayList internally resize itself? Explain with memory allocation details.
2. What happens when you try to access an index beyond the size of an ArrayList?
3. Why does ArrayList allow null elements, but HashSet does not?
4. How does ArrayList.remove(Object o) work internally?
5. How can we efficiently remove multiple elements from an ArrayList?
6. Why is ArrayList not suitable for frequent insertions and deletions?
7. How does subList() work in ArrayList? Can modifications in subList affect the original list?
8. What is the difference between ensureCapacity() and trimToSize() in ArrayList?
9. How can we create an ArrayList that prevents duplicate elements?
10. Implement a dynamic array class similar to ArrayList with auto-resizing logic.

4. Comparable

1. How does compareTo() work in Comparable? Explain with an example.
2. Why should compareTo() be consistent with equals()? What happens if it’s not?
3. What happens if compareTo() is implemented incorrectly?

4. Can a class implement both Comparable and Comparator? Provide a practical example.
5. How does TreeSet use Comparable for sorting elements?
6. Can we change the sorting order defined in Comparable? If yes, how?
7. What are the risks of using Comparable for sorting in large datasets?
8. Why do wrapper classes like Integer and String implement Comparable?
9. How can we sort a list of custom objects in descending order using Comparable?
10. Write a custom implementation of the Comparable interface for a Student class.

5. Comparator

1. What is the difference between compareTo() in Comparable and compare() in Comparator?
2. How do you implement sorting based on multiple fields using Comparator?
3. Why is Comparator preferred over Comparable for sorting objects dynamically?
4. How does Comparator.comparing() work in Java 8?
5. How can we use method references in Comparator? Provide an example.
6. How does thenComparing() work in Comparator?
7. What is the default behavior of Collections.sort() when using a Comparator?
8. How does a custom Comparator affect sorting performance?
9. What happens if Comparator returns inconsistent results?
10. Write a comparator to sort a list of employees by salary, then by name, then by age.