Một **Map** là một cấu trúc dữ liệu trong Java được sử dụng để lưu trữ các cặp key-value. Dưới đây là một số đặc điểm quan trọng của **Map** trong Java:

1. **Cặp Key-Value**: **Map** lưu trữ dữ liệu dưới dạng các cặp key-value, trong đó mỗi key tương ứng với một value. Mỗi key trong **Map** là duy nhất và không thể trùng lặp.
2. **Không theo thứ tự (Unordered)**: Trong các phiên bản trước của Java, **HashMap** không đảm bảo thứ tự của các cặp key-value. Tuy nhiên, từ Java 8, **LinkedHashMap** và **TreeMap** đảm bảo thứ tự của các cặp key-value dựa trên các tiêu chí khác nhau.
3. **Khóa (Key) duy nhất**: Mỗi key trong một **Map** là duy nhất. Điều này có nghĩa là không thể có hai key giống nhau trong một **Map**.
4. **Giá trị (Value) có thể là bất kỳ loại dữ liệu nào**: Giá trị của mỗi cặp key-value có thể là bất kỳ loại dữ liệu nào, bao gồm cả các đối tượng tùy chỉnh.
5. **Có thể chứa giá trị null**: Một **Map** có thể chứa giá trị null nhưng chỉ có một key null duy nhất.
6. **Performance**: Hiệu suất truy xuất và thêm các phần tử vào **Map** thường là rất nhanh đối với các cài đặt hiệu quả như **HashMap**.
7. **Iterating (Lặp qua các phần tử)**: **Map** cung cấp các phương thức để lặp qua các phần tử của nó, như **forEach()**, **keySet()**, **entrySet()**.
8. **Không thread-safe**: Đa số các cài đặt của **Map** không đảm bảo tính an toàn khi sử dụng đồng thời trong môi trường đa luồng mà không có các biện pháp đồng bộ hóa bên ngoài.
9. **Các cài đặt phổ biến**: Java cung cấp một số cài đặt **Map**, bao gồm **HashMap**, **LinkedHashMap**, **TreeMap**, **Hashtable**, và **ConcurrentHashMap**.
10. **Các phương thức cơ bản**: Một số phương thức cơ bản của **Map** bao gồm **put()**, **get()**, **containsKey()**, **containsValue()**, **remove()**, và **size()**.

**List (Interface) extends Collection**

**ArrayList**

**Set (Interface)**

-Set không cho phép thêm phần tử trùng lặp, không có khái niệm về chỉ số

-Dùng hashSet (default) để triển khai interface

-HashSet không đảm bảo thứ tự (được lưu trữ dựa trên giá trị Hash). Đây là có hiệu suất cao nhất trong các loại Set.

-LinkedHashSet đảm bảo thứ tự

-TreeSet tăng dần theo giá trị

Các phương thức add (thêm), remove (xóa), contains (kiểm tra tồn tại), isEmpty (kiểm tra trống), addAll (truyền tất cả list vào)

* **Vậy khi nào dùng set: dùng khi chúng ta không muốn lưu trữ trùng và không quan tâm thứ tự**