# Đặc điểm của Map Interface

1. **Không chứa các phần tử trùng lặp**: Một Map không thể chứa các khóa trùng lặp, mỗi khóa có thể liên kết với một giá trị duy nhất.
2. **Cặp key-value**: Map lưu trữ dữ liệu dưới dạng các cặp key (khóa) và value (giá trị). Khóa có thể được sử dụng để truy xuất giá trị tương ứng.
3. **Không tuần tự**: Thứ tự của các cặp key-value không được đảm bảo (trừ một số trường hợp cụ thể như LinkedHashMap hoặc TreeMap).
4. **Các phương thức cơ bản**:

* put(K key, V value): Thêm hoặc cập nhật giá trị tương ứng với khóa.
* get(Object key): Lấy giá trị dựa trên khóa.
* remove(Object key): Xóa phần tử dựa trên khóa.
* containsKey(Object key): Kiểm tra sự tồn tại của khóa.
* containsValue(Object value): Kiểm tra sự tồn tại của giá trị.
* keySet(): Trả về tập hợp các khóa.
* values(): Trả về tập hợp các giá trị.

# Các class triển khai từ Map Interface

1. **HashMap**: Lưu trữ các phần tử không theo thứ tự cụ thể và cho phép null cho cả khóa và giá trị.
2. **LinkedHashMap**: Lưu trữ các phần tử theo thứ tự chèn vào và cho phép null cho cả khóa và giá trị.
3. **TreeMap**: Lưu trữ các phần tử theo thứ tự sắp xếp tự nhiên của khóa hoặc dựa trên một Comparator tùy chỉnh. Không cho phép null cho khóa.
4. **Hashtable**: Đồng bộ (synchronized), không cho phép null cho cả khóa và giá trị.
5. **ConcurrentHashMap**: Phiên bản đồng bộ (thread-safe) của HashMap, tối ưu hiệu suất trong môi trường đa luồng.

# Phân biệt và trường hợp sử dụng từng class

1. **HashMap**:

* **Đặc điểm**: Không đảm bảo thứ tự các phần tử, tốc độ truy xuất nhanh (O(1)) khi sử dụng các phương thức như get() và put().
* **Sử dụng khi**: Cần một cấu trúc dữ liệu nhanh chóng để lưu trữ và truy xuất các cặp key-value, và không cần quan tâm đến thứ tự các phần tử.

1. **LinkedHashMap**:

* **Đặc điểm:** Duy trì thứ tự các phần tử dựa trên thứ tự chèn vào. Tốc độ truy xuất cũng tương đối nhanh (O(1)).
* **Sử dụng khi**: Cần lưu trữ các phần tử theo thứ tự chèn, ví dụ như khi muốn duy trì thứ tự của các lần truy cập trước đó.

1. **TreeMap**:

* **Đặc điểm:** Các phần tử được sắp xếp theo thứ tự tự nhiên (hoặc theo Comparator cung cấp). Thời gian truy xuất lâu hơn (O(log n)) so với HashMap do phải duy trì cấu trúc cây.
* **Sử dụng khi**: Cần lưu trữ và truy xuất các phần tử theo thứ tự sắp xếp. Ví dụ, khi cần thao tác trên dữ liệu theo thứ tự tăng dần hoặc giảm dần của khóa.

1. **Hashtable**:

* **Đặc điểm:** Đồng bộ hóa (synchronized) cho nên thích hợp cho môi trường đa luồng. Không cho phép null cho cả khóa và giá trị.
* **Sử dụng khi**: Cần một cấu trúc dữ liệu thread-safe để quản lý các cặp key-value và không quan tâm đến thứ tự.

1. **ConcurrentHashMap**:

* **Đặc điểm:** Đồng bộ hóa, nhưng hiệu suất tốt hơn Hashtable vì chỉ khóa cục bộ (partial locking). Cho phép nhiều thread truy xuất cùng lúc mà không ảnh hưởng đến hiệu năng.
* **Sử dụng khi**: Cần một Map thread-safe cho các ứng dụng đa luồng, nhưng muốn tối ưu hiệu năng hơn so với Hashtable..