Map Interface

1. Nêu ra các đặc điểm **Map Interface**

Map là một interface trong Java Collections Framework, dùng để lưu trữ dữ liệu dưới dạng các cặp **khóa-giá trị** (key-value). Nó không kế thừa từ interface Collection và do đó, nó không tuân thủ các quy tắc của Collection như các interface khác (List, Set, Queue).

**Đặc điểm của Map Interface**

* **Lưu trữ cặp khóa-giá trị**: Mỗi phần tử trong Map là một cặp bao gồm một khóa (key) và một giá trị (value) tương ứng.
* **Không cho phép khóa trùng lặp**: Mỗi khóa trong Map phải là duy nhất. Nếu bạn cố gắng thêm một cặp có khóa đã tồn tại, giá trị cũ sẽ bị ghi đè.
* **Không có thứ tự cụ thể**: Thứ tự của các cặp khóa-giá trị trong Map thường không được đảm bảo, trừ một số trường hợp đặc biệt của các lớp triển khai.
* **Truy cập nhanh**: Map cung cấp các phương thức hiệu quả để tìm kiếm, thêm, và xóa giá trị dựa trên khóa của chúng.

2. Kể ra các **class**triển khai từ **Map Interface**

Có ba lớp chính triển khai từ Map Interface, mỗi lớp có cấu trúc dữ liệu và mục đích sử dụng riêng:

**HashMap**: Lớp này sử dụng một bảng băm (hash table) để lưu trữ các cặp khóa-giá trị.

**LinkedHashMap**: Lớp này là sự kết hợp giữa bảng băm và danh sách liên kết, duy trì thứ tự chèn.

**TreeMap**: Lớp này sử dụng cấu trúc cây tìm kiếm nhị phân cân bằng (red-black tree) để lưu trữ các cặp khóa-giá trị.

3. Phân biệt rõ trường hợp sử dụng của từng **class**đó

**HashMap** là lớp triển khai Map được sử dụng phổ biến nhất. Nó cung cấp hiệu suất cao nhất cho các thao tác cơ bản.

* **Cấu trúc dữ liệu**: Bảng băm.
* **Ưu điểm**:
* **Hiệu suất cao**: Các thao tác thêm, xóa và tìm kiếm (get, put, remove) có độ phức tạp thời gian trung bình là O(1).
* **Nhược điểm**:
* **Không duy trì thứ tự**: Thứ tự của các cặp không được đảm bảo. Khi duyệt qua HashMap, thứ tự có thể thay đổi.

**Trường hợp sử dụng**: Dùng khi cần lưu trữ dữ liệu dưới dạng cặp khóa-giá trị và yêu cầu hiệu suất cao, không quan tâm đến thứ tự của chúng. Ví dụ: lưu trữ thông tin của người dùng (khóa là userId, giá trị là User object), tra cứu mã sản phẩm và tên sản phẩm.

**LinkedHashMap** kết hợp tốc độ của HashMap với khả năng duy trì thứ tự.

* **Cấu trúc dữ liệu**: Bảng băm + danh sách liên kết kép.
* **Ưu điểm**:
* **Duy trì thứ tự chèn**: Các phần tử được duyệt theo thứ tự mà chúng được thêm vào.
* **Hiệu suất tốt**: Các thao tác vẫn có độ phức tạp O(1) tương tự HashMap.
* **Nhược điểm**:
* **Tốn bộ nhớ hơn**: Do phải duy trì thêm danh sách liên kết, nó tốn nhiều bộ nhớ hơn HashMap.

**Trường hợp sử dụng**: Dùng khi cần hiệu suất nhanh như HashMap nhưng vẫn muốn duy trì thứ tự chèn. Ví dụ: lưu trữ lịch sử các trang đã truy cập trong một phiên làm việc, nơi thứ tự rất quan trọng.

**TreeMap** lưu trữ các cặp khóa-giá trị theo một thứ tự đã sắp xếp.

* **Cấu trúc dữ liệu**: Cây tìm kiếm nhị phân cân bằng (red-black tree).
* **Ưu điểm**:
* **Tự động sắp xếp**: Các khóa được sắp xếp theo thứ tự tự nhiên của chúng hoặc theo một Comparator tùy chỉnh.
* **Nhược điểm**:
* **Hiệu suất thấp hơn**: Các thao tác thêm, xóa và tìm kiếm có độ phức tạp thời gian là O(logn), chậm hơn so với HashMap và LinkedHashMap.

**Trường hợp sử dụng**: Dùng khi cần một Map mà các khóa của nó phải luôn được sắp xếp. Ví dụ: lưu trữ danh sách các điểm số theo thứ tự từ thấp đến cao, hoặc lưu trữ dữ liệu theo thời gian (timestamps) để dễ dàng duyệt theo thứ tự thời gian.