Set Interface

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

**Set** là một interface trong Java Collections Framework, đại diện cho một tập hợp các phần tử không có thứ tự và không cho phép các phần tử trùng lặp.

Các đặc điểm chính của Set Interface:

* **Không có thứ tự**: Các phần tử không được lưu trữ theo một chỉ mục cụ thể và thứ tự của chúng có thể không được đảm bảo.
* **Không cho phép trùng lặp**: Nếu cố gắng thêm một phần tử đã tồn tại vào một Set, thao tác này sẽ bị bỏ qua và phương thức add() sẽ trả về false.
* **Không có phương thức truy cập theo chỉ mục**: Do không có thứ tự, Set không có phương thức get(int index) như List.

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

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

* **HashSet**: Lớp này sử dụng một bảng băm (hash table) để lưu trữ các phần tử.
* **LinkedHashSet**: Lớp này là sự kết hợp giữa bảng băm và danh sách liên kết, duy trì thứ tự chèn.
* **TreeSet**: Lớp này sử dụng cấu trúc cây tìm kiếm nhị phân cân bằng (red-black tree) để lưu trữ các phần tử.

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

**HashSet** là lựa chọn phổ biến nhất và hiệu quả nhất khi bạn chỉ cần lưu trữ các phần tử không trùng lặp mà không quan tâm đến thứ tự của chúng.

* **Cấu trúc dữ liệu**: Bảng băm (hash table).
* **Ưu điểm**:
* **Hiệu suất cao**: Các thao tác thêm, xóa và tìm kiếm phần tử có độ phức tạp thời gian trung bình là O(1).
* **Nhược điểm**:
* **Không duy trì thứ tự**: Thứ tự chèn hoặc bất kỳ thứ tự nào khác đều không được đảm bảo.

**Trường hợp sử dụng**: Dùng khi cần kiểm tra sự tồn tại của một phần tử nhanh chóng hoặc loại bỏ các phần tử trùng lặp. Ví dụ: lưu trữ danh sách các từ duy nhất từ một văn bản, hoặc các địa chỉ email duy nhất của người dùng.

**LinkedHashSet** kết hợp tốc độ của HashSet với khả năng duy trì thứ tự chèn.

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

**Trường hợp sử dụng:** Dùng khi cần loại bỏ các phần tử trùng lặp nhưng vẫn muốn duy trì thứ tự thêm vào ban đầu. Ví dụ: lưu trữ danh sách các tag mà người dùng đã thêm vào một bài viết, theo đúng thứ tự họ đã nhập.

**TreeSet** lưu trữ các phần tử theo thứ tự tự nhiên của chúng hoặc theo một Comparator tùy chỉnh.

* **Cấu trúc dữ liệu**: Cây tìm kiếm nhị phân cân bằng (red-black tree).
* **Ưu điểm**:
* **Tự động sắp xếp**: Các phần tử luôn được lưu trữ theo thứ tự tăng dần.
* **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 O(1) của HashSet.

**Trường hợp sử dụng**: Dùng khi cần các phần tử không trùng lặp và muốn chúng luôn được sắp xếp. Ví dụ: lưu trữ danh sách các điểm số duy nhất theo thứ tự tăng dần, hoặc lưu trữ các tên người dùng được sắp xếp theo thứ tự bảng chữ cái.