# Câu 1:

a. Khái quát chức năng, nhiệm vụ của tập hợp LinkedHashSet:

LinkedHashSet là một tập hợp con của HashSet và kế thừa các tính năng của HashSet. LinkedHashSet lưu trữ các phần tử dưới dạng một tập hợp không trùng lặp các phần tử, có thể sắp xếp và có thể truy cập nhanh các phần tử của tập hợp.

b. Trong Java, lớp LinkedHashSet là một lớp triển khai của giao diện Set, và giao diện Set được kế thừa từ giao diện Collection. Do đó, cấu trúc phân cấp của LinkedHashSet sẽ là:

Object -> Interface Collection -> Interface Set -> LinkedHashSet

Trong cây phân cấp này, LinkedHashSet kế thừa tất cả các thuộc tính và phương thức từ các lớp cha của nó và triển khai chúng cho phù hợp với việc lưu trữ một tập hợp các phần tử duy nhất theo thứ tự chèn của chúng.

c. Đặc điểm nổi bật của LinkedHashSet:

Đặc điểm nổi bật của LinkedHashSet là nó kết hợp tính năng giữ nguyên thứ tự chèn và tính duy nhất của các phần tử, tương tự như HashSet. Điều này có nghĩa là, khi chèn một phần tử vào LinkedHashSet, nó sẽ được thêm vào cuối danh sách và nếu phần tử đó đã tồn tại thì nó sẽ không được thêm vào lại. Đồng thời, các phần tử được truy cập trong LinkedHashSet sẽ được trả về theo thứ tự chèn của chúng.

Tính năng giữ nguyên thứ tự chèn của LinkedHashSet là rất hữu ích trong các tình huống cần phải lưu trữ và duyệt các phần tử theo đúng thứ tự chèn của chúng, chẳng hạn như lưu trữ các thao tác của người dùng trên một ứng dụng, hoặc lưu trữ các tên file trong một thư mục theo thứ tự chèn.

Câu 2: Khai báo lớp LinkedHashSet:

Trong Java, có hai cách để khai báo đối tượng LinkedHashSet:

Cách 1: Sử dụng hàm tạo không tham số để tạo đối tượng LinkedHashSet rỗng:

Cú pháp khai báo:

LinkedHashSet<E> linkedHashSet = new LinkedHashSet<>();

Trong đó E là kiểu dữ liệu của các phần tử trong tập hợp.Trong ví dụ trên, đối tượng set được khởi tạo mà không có phần tử nào trong tập hợp.

Cách 2: Sử dụng hàm tạo có tham số của lớp LinkedHashSet để tạo một LinkedHashSet từ một tập hợp khác, chẳng hạn như:

Set<Integer> hashSet = new HashSet<>();

hashSet.add(1);

hashSet.add(2);

hashSet.add(3);

LinkedHashSet<Integer> linkedHashSet = new LinkedHashSet<>(hashSet);

Câu 3: Các phép toán thường dùng trong LinkedHashSet:

a. Thêm phần tử:

Dùng phương thức boolean add(Object element); // triển khai từ interface Set

Ví dụ: linkedHashSet.add(x);

b. Lấy phần tử:

LinkedHashSet không có phương thức lấy phần tử theo chỉ số. Ta có thể duyệt qua các phần tử bằng Iterator hoặc for-each loop.

Sử dụng vòng lặp foreach để lấy tất cả các phần tử trong tập hợp:

for (E element : linkedHashSet) {

// Code làm gì đó với phần tử lấy ra

}

Sử dụng phương thức iterator() để lấy tất cả các phần tử trong tập hợp:

Iterator<E> iterator = linkedHashSet.iterator();

while (iterator.hasNext()) {

E element = iterator.next();

// Code làm gì đó với phần tử lấy ra

}

c. Xóa phần tử:

Sử dụng phương thức boolean remove(Object element) triển khai từ interface Set.

Ví dụ: linkedHashSet.remove(element);

d. Sửa phần tử:

LinkedHashSet không hỗ trợ phương thức để sửa phần tử. Muốn sửa phần tử, ta phải xóa phần tử đó ra khỏi tập hợp và thêm phần tử mới vào tập hợp.

Ngoài ra ta có thể sao chép tập hợp đó vào một danh sách (List) trước khi thực hiện sửa đổi. Sau đó, bạn sẽ tạo lại một LinkedHashSet mới từ danh sách đã được sửa đổi để cập nhật tập hợp ban đầu.

e. Chuyển đổi tập hợp LinkedHashSet đang tìm hiểu sang tập hợp khác:

Chuyển đổi sang HashSet:

Set<E> hashSet = new HashSet<>(linkedHashSet);

Chuyển đổi sang TreeSet:

Set<E> treeSet = new TreeSet<>(linkedHashSet);

f. Tìm phần tử lớn nhất/ nhỏ nhất/ sắp xếp các phần tử trong tập hợp:

Để tìm phần tử lớn nhất hoặc nhỏ nhất và sắp xếp trong LinkedHashSet, ta có thể sử dụng lớp tiện ích Collections của Java. Đầu tiên, ta cần chuyển LinkedHashSet thành List, sau đó sử dụng phương thức Collections.max() hoặc Collections.min() để tìm giá trị lớn nhất hoặc nhỏ nhất và Collections.sort() để sắp xếp.

Ví dụ tìm max:

LinkedHashSet<Integer> set = new LinkedHashSet<>();

// thêm các phần tử vào set

List<Integer> list = new ArrayList<>(set);

int max = Collections.max(list);

System.out.println("Giá trị lớn nhất trong tập hợp là: " + max);

Tương tự, để tìm giá trị nhỏ nhất, bạn chỉ cần thay Collections.max() bằng Collections.min().

Ví dụ về sắp xếp:

LinkedHashSet<Integer> set = new LinkedHashSet<>();

// thêm các phần tử vào set

List<Integer> list = new ArrayList<>(set);

Collections.sort(list);

System.out.println("Các phần tử đã được sắp xếp: " + list);

g. So sánh ý nghĩa trong áp dụng tập hợp xử lý các bài toán trong thực tế: LinkedHashSet được sử dụng khi ta muốn lưu trữ các phần tử theo thứ tự chèn, đồng thời loại bỏ các phần tử trùng lặp. Ví dụ, trong việc lưu trữ lịch sử truy cập trang web của người dùng, LinkedHashSet được sử dụng để lưu trữ các URL theo thứ tự truy cập và loại bỏ các URL trùng lặp.