**[JAVA COLLECTION](https://classroom.google.com/w/Njc0NDQwMTUzOTQ4/tc/Njc0NDQ3MDQ0NzIz)**

**4. Set**

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

Set là một trong những giao diện (interface) trong Java Collection Framework. Dưới đây là một số đặc điểm chính của Set interface:

1. **Tập hợp duy nhất**:
   * Set không cho phép các phần tử trùng lặp. Mỗi phần tử trong một tập hợp là duy nhất.
2. **Không xác định thứ tự**:
   * Các phần tử trong Set không có thứ tự nhất định (trừ LinkedHashSet, nó bảo toàn thứ tự chèn). Điều này có nghĩa là khi bạn lặp qua các phần tử trong một Set, bạn không thể dự đoán thứ tự mà các phần tử sẽ được trả về.
3. **Các phương thức chính**:
   * add(E e): Thêm một phần tử vào tập hợp.
   * remove(Object o): Xóa một phần tử khỏi tập hợp.
   * contains(Object o): Kiểm tra xem tập hợp có chứa một phần tử cụ thể hay không.
   * size(): Trả về số lượng phần tử trong tập hợp.
   * isEmpty(): Kiểm tra xem tập hợp có rỗng hay không.
4. **Các loại Set**:
   * **HashSet**: Tập hợp sử dụng bảng băm (hash table) để lưu trữ các phần tử. Nó không bảo toàn thứ tự chèn.
   * **LinkedHashSet**: Tập hợp duy trì thứ tự chèn của các phần tử.
   * **TreeSet**: Tập hợp tự động sắp xếp các phần tử theo thứ tự tự nhiên hoặc theo một Comparator được cung cấp.
5. **Hỗ trợ cho phép null**:
   * Set cho phép một giá trị null, nhưng chỉ một giá trị null duy nhất.
6. **Tính tương tác**:
   * Set có thể tương tác với các cấu trúc dữ liệu khác và cho phép các phép toán tập hợp như giao, hợp, và hiệu thông qua các phương thức như retainAll(Collection<?> c), addAll(Collection<? extends E> c), và removeAll(Collection<?> c).
7. **Không đồng bộ**:
   * Các thực thể của Set không đồng bộ, nghĩa là nếu bạn cần sử dụng Set trong môi trường đa luồng, bạn phải đồng bộ hóa nó thủ công hoặc sử dụng Collections.synchronizedSet(Set<T> s).

# **2. Nêu ra các đặc điểm HashSet?**

HashSet là một trong những lớp thực thi của giao diện Set trong Java Collection Framework. Dưới đây là một số đặc điểm chính của HashSet:

1. **Không cho phép phần tử trùng lặp**:
   * HashSet không lưu trữ các phần tử trùng lặp. Mỗi phần tử trong HashSet là duy nhất.
2. **Không bảo toàn thứ tự**:
   * Các phần tử trong HashSet không có thứ tự xác định. Điều này có nghĩa là bạn không thể dự đoán thứ tự của các phần tử khi lặp qua chúng.
3. **Tốc độ truy cập nhanh**:
   * HashSet sử dụng bảng băm để lưu trữ các phần tử, do đó thời gian truy cập (thêm, xóa, và kiểm tra phần tử) trung bình là O(1).
4. **Cho phép một giá trị null**:
   * HashSet cho phép một giá trị null. Tuy nhiên, chỉ có thể có một phần tử null duy nhất trong tập hợp.
5. **Không đồng bộ**:
   * HashSet không đồng bộ. Nếu bạn muốn sử dụng HashSet trong môi trường đa luồng, bạn cần phải đồng bộ hóa nó thủ công hoặc sử dụng Collections.synchronizedSet(Set<T> s).
6. **Hỗ trợ các thao tác tập hợp**:
   * HashSet hỗ trợ các phép toán tập hợp như giao, hợp, và hiệu thông qua các phương thức như retainAll(Collection<?> c), addAll(Collection<? extends E> c), và removeAll(Collection<?> c).
7. **Khả năng mở rộng**:
   * HashSet có thể tự động mở rộng kích thước khi cần thiết (khi số lượng phần tử vượt quá một ngưỡng nhất định).
8. **Khả năng lưu trữ các đối tượng không nhất thiết phải cùng kiểu**:
   * Vì HashSet kế thừa từ Collection, nó có thể chứa các đối tượng thuộc các kiểu khác nhau (nếu không chỉ định kiểu cụ thể), nhưng điều này không phải là một thực tiễn tốt.
9. **Khi nào nên sử dụng**:
   * HashSet là lựa chọn tốt khi bạn cần một tập hợp các phần tử duy nhất mà không cần quan tâm đến thứ tự và cần hiệu suất cao trong các thao tác thêm, xóa, và kiểm tra sự tồn tại.

# **3. Nêu ra các đặc điểm LinkedHashSet?**

LinkedHashSet là một lớp trong Java Collection Framework kế thừa từ HashSet và thực hiện giao diện Set. Dưới đây là một số đặc điểm chính của LinkedHashSet:

1. **Bảo toàn thứ tự chèn**:
   * LinkedHashSet duy trì thứ tự chèn của các phần tử. Điều này có nghĩa là khi bạn lặp qua LinkedHashSet, các phần tử sẽ được trả về theo thứ tự mà chúng đã được thêm vào.
2. **Không cho phép phần tử trùng lặp**:
   * Giống như HashSet, LinkedHashSet không cho phép các phần tử trùng lặp. Mỗi phần tử trong LinkedHashSet là duy nhất.
3. **Tốc độ truy cập nhanh**:
   * LinkedHashSet sử dụng một bảng băm để lưu trữ các phần tử, do đó thời gian truy cập (thêm, xóa, và kiểm tra phần tử) trung bình là O(1). Tuy nhiên, vì nó duy trì danh sách liên kết để bảo toàn thứ tự, nên việc lưu trữ có thể tốn nhiều bộ nhớ hơn so với HashSet.
4. **Cho phép một giá trị null**:
   * LinkedHashSet cho phép một giá trị null, giống như HashSet. Tuy nhiên, chỉ có thể có một phần tử null duy nhất trong tập hợp.
5. **Không đồng bộ**:
   * LinkedHashSet không đồng bộ. Nếu bạn muốn sử dụng nó trong môi trường đa luồng, bạn cần phải đồng bộ hóa nó thủ công hoặc sử dụng Collections.synchronizedSet(Set<T> s).
6. **Hỗ trợ các thao tác tập hợp**:
   * LinkedHashSet hỗ trợ các phép toán tập hợp như giao, hợp, và hiệu thông qua các phương thức như retainAll(Collection<?> c), addAll(Collection<? extends E> c), và removeAll(Collection<?> c).
7. **Khả năng mở rộng**:
   * Giống như HashSet, LinkedHashSet cũng có thể tự động mở rộng kích thước khi cần thiết.
8. **Hiệu suất tốt cho các tình huống cần duy trì thứ tự**:
   * LinkedHashSet là lựa chọn lý tưởng khi bạn cần một tập hợp duy nhất mà cần duy trì thứ tự của các phần tử trong quá trình thêm và xóa.

# **4. Nêu ra các đặc điểm TreeSet?**

TreeSet là một lớp trong Java Collection Framework, thực hiện giao diện Set và kế thừa từ AbstractSet. Dưới đây là một số đặc điểm chính của TreeSet:

1. **Sắp xếp tự động**:
   * TreeSet tự động sắp xếp các phần tử theo thứ tự tự nhiên của chúng hoặc theo một Comparator được cung cấp tại thời điểm khởi tạo. Điều này có nghĩa là khi bạn lặp qua các phần tử trong TreeSet, chúng sẽ được trả về theo thứ tự đã sắp xếp.
2. **Không cho phép phần tử trùng lặp**:
   * Giống như các lớp Set khác, TreeSet không cho phép các phần tử trùng lặp. Mỗi phần tử trong TreeSet phải là duy nhất.
3. **Chạy chậm hơn so với HashSet**:
   * Thời gian truy cập (thêm, xóa, và kiểm tra phần tử) trong TreeSet là O(log n) do việc sử dụng cây đỏ-đen (red-black tree) để tổ chức các phần tử, chậm hơn so với HashSet, mà thời gian truy cập trung bình là O(1).
4. **Không cho phép giá trị null**:
   * TreeSet không cho phép phần tử null. Nếu bạn cố gắng thêm giá trị null, sẽ phát sinh NullPointerException.
5. **Hỗ trợ các thao tác tập hợp**:
   * TreeSet hỗ trợ các phép toán tập hợp như giao, hợp, và hiệu thông qua các phương thức như retainAll(Collection<?> c), addAll(Collection<? extends E> c), và removeAll(Collection<?> c).
6. **Tính tương tác với các cấu trúc dữ liệu khác**:
   * TreeSet có thể dễ dàng chuyển đổi sang các cấu trúc dữ liệu khác như ArrayList hoặc HashSet.
7. **Có thể truy cập các phần tử lớn nhất và nhỏ nhất**:
   * TreeSet cung cấp các phương thức như first() và last() để truy cập phần tử nhỏ nhất và lớn nhất trong tập hợp.
8. **Không đồng bộ**:
   * TreeSet không đồng bộ. Nếu bạn cần sử dụng TreeSet trong môi trường đa luồng, bạn cần phải đồng bộ hóa nó thủ công hoặc sử dụng Collections.synchronizedSortedSet(Set<T> s).
9. **Khả năng mở rộng**:
   * TreeSet có thể tự động mở rộng kích thước khi cần thiết.