# 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. Tập hợp kế thừa từ lớp HashSet và implements interface Set.

LinkedHashSet được đánh dấu ở vị trí thứ 7 trong cây phân cấp của Java Collection Framework.

c. Đặc điểm nổi bật của tập hợp LinkedHashSet:

Lưu trữ các phần tử dưới dạng một tập hợp không trùng lặp các phần tử.

Có thể sắp xếp các phần tử theo thứ tự chèn (theo thứ tự phần tử được thêm vào tập hợp).

Có thể truy cập nhanh các phần tử của tập hợp.

Khai báo lớp LinkedHashSet:

Cú pháp khai báo:

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

Trong đó E là kiểu dữ liệu của các phần tử trong tập hợp.

Câu 2: Các phép toán thường dùng trong LinkedHashSet:

a. Thêm phần tử:

linkedHashSet.add(element);

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) {

// do something with element

}

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();

// do something with element

}

c. Xóa phần tử:

Sử dụng phương thức remove().

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.

Không có phương thức đặc biệt để sửa phần tử trong LinkedHashSet. Tuy nhiên, ta có thể xóa phần tử cũ và thêm phần tử mới vào tập hợp để thực hiện việc sửa đổi phần tử.

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:

Tìm phần tử lớn nhất:

E maxElement = Collections.max(linkedHashSet);

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.

Câu 4:

a. Phát biểu bài toán: Cho một danh sách các số nguyên. Hãy viết chương trình loại bỏ các số trùng lặp và giữ lại các số theo thứ tự xuất hiện ban đầu.

b. Thực hiện cài đặt minh họa bài toán điển hình thao tác với các đối tượng nguyên thủy:

import java.util.LinkedHashSet;

public class Main {

public static void main(String[] args) {

// Khai báo và khởi tạo tập hợp

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

// Thêm các phần tử vào tập hợp

set.add(5);

set.add(1);

set.add(3);

set.add(2);

set.add(4);

set.add(5); // thêm phần tử trùng lặp, tập hợp vẫn chỉ có 5 phần tử

// In ra các phần tử trong tập hợp

System.out.println("Các phần tử trong tập hợp là: " + set);

// Lấy phần tử đầu tiên của tập hợp

int first = set.iterator().next();

System.out.println("Phần tử đầu tiên trong tập hợp là: " + first);

// Xóa một phần tử khỏi tập hợp

set.remove(3);

System.out.println("Các phần tử trong tập hợp sau khi xóa 3 là: " + set);

// Sửa một phần tử trong tập hợp

set.remove(4);

set.add(6);

System.out.println("Các phần tử trong tập hợp sau khi sửa 4 thành 6 là: " + set);

// Chuyển đổi tập hợp sang mảng

Integer[] arr = set.toArray(new Integer[0]);

System.out.println("Mảng chứa các phần tử trong tập hợp là: " + Arrays.toString(arr));

// Tìm phần tử lớn nhất và nhỏ nhất trong tập hợp

int min = Collections.min(set);

int max = Collections.max(set);

System.out.println("Phần tử nhỏ nhất trong tập hợp là: " + min);

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

// Sắp xếp tập hợp theo thứ tự giảm dần

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

Collections.sort(list, Collections.reverseOrder());

LinkedHashSet<Integer> sortedSet = new LinkedHashSet<>(list);

System.out.println("Các phần tử trong tập hợp sau khi sắp xếp giảm dần là: " + sortedSet);

}

}

c. Thực hiện Bài toán điển hình thao tác với các đối tượng tự định nghĩa.

public class Student {

private int id;

private String name;

private int age;

private String address;

public Student(int id, String name, int age, String address) {

this.id = id;

this.name = name;

this.age = age;

this.address = address;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

public String getAddress() {

return address;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Student student = (Student) o;

return id == student.id;

}

@Override

public int hashCode() {

return Objects.hash(id);

}

@Override

public String toString() {

return "Student{" +

"id=" + id +

", name='" + name + '\'' +

", age=" + age +

", address='" + address + '\'' +

'}';

}

}

LinkedHashSet<Student> students = new LinkedHashSet<>();

// Thêm các đối tượng Student vào tập hợp

students.add(new Student(1, "John", 18, "123 Main St"));

students.add(new Student(2, "Jane", 20, "456 Park Ave"));

students.add(new Student(3, "Bob", 19, "789 Elm St"));

// In ra tất cả các đối tượng Student trong tập hợp

for (Student student : students) {

System.out.println(student);

}

// Xóa một đối tượng Student khỏi tập hợp

students.remove(new Student(2, "Jane", 20, "456 Park Ave"));