# Task4:Collection Framework and Generic

## I.Collection Framework

### 1. ****Interfaces (Giao diện)****

+Các giao diện trong Collection Framework định nghĩa các phương thức mà các lớp cài đặt chúng cần phải cung cấp. Các giao diện chính bao gồm:

+Collection: Giao diện cơ sở cho tất cả các kiểu tập hợp. Nó định nghĩa các phương thức cơ bản như add(), remove(), size(), và clear().

+List: Kế thừa từ Collection, đại diện cho một tập hợp các phần tử có thứ tự. Các phương thức bao gồm add(), remove(), get(), set(), và indexOf(). Các lớp cài đặt chính là ArrayList, LinkedList, và Vector.

+Set: Kế thừa từ Collection, đại diện cho một tập hợp không có các phần tử trùng lặp. Các lớp cài đặt chính là HashSet, LinkedHashSet, và TreeSet.

+Queue: Kế thừa từ Collection, đại diện cho một cấu trúc hàng đợi. Các lớp cài đặt chính là LinkedList, PriorityQueue, và ArrayDeque.

+Deque: Kế thừa từ Queue, đại diện cho một hàng đợi hai đầu. Các lớp cài đặt chính là ArrayDeque và LinkedList.

+Map: Không kế thừa từ Collection, nhưng vẫn là một phần của Collection Framework. Nó lưu trữ các cặp khóa-giá trị. Các lớp cài đặt chính là HashMap, LinkedHashMap, và TreeMap.

### 2. Classes (Lớp)

Các lớp trong Collection Framework thực hiện các giao diện và cung cấp các chức năng cụ thể:

+ArrayList: Cài đặt giao diện List, sử dụng mảng để lưu trữ các phần tử. Hỗ trợ truy cập nhanh và các phép toán như thêm, xóa phần tử.

+LinkedList: Cài đặt giao diện List và Deque, sử dụng danh sách liên kết. Hỗ trợ các thao tác chèn và xóa hiệu quả hơn ArrayList.

+HashSet: Cài đặt giao diện Set, sử dụng bảng băm để lưu trữ phần tử. Hỗ trợ tìm kiếm nhanh và không cho phép phần tử trùng lặp.

+LinkedHashSet: Cài đặt giao diện Set, tương tự như HashSet, nhưng duy trì thứ tự chèn của các phần tử.

+TreeSet: Cài đặt giao diện Set, lưu trữ phần tử theo thứ tự tự nhiên hoặc theo một Comparator đã cung cấp.

+HashMap: Cài đặt giao diện Map, sử dụng bảng băm để lưu trữ các cặp khóa-giá trị. Cho phép truy cập nhanh theo khóa.

+LinkedHashMap: Cài đặt giao diện Map, duy trì thứ tự chèn của các mục khóa-giá trị.

+TreeMap: Cài đặt giao diện Map, lưu trữ các mục khóa-giá trị theo thứ tự tự nhiên của khóa hoặc theo một Comparator đã cung cấp.

### 3. ****Key Methods (Các phương thức chính)****

Một số phương thức quan trọng trong các giao diện và lớp của Collection Framework:

+add(E e): Thêm phần tử vào tập hợp.

+remove(Object o): Xóa phần tử khỏi tập hợp.

+contains(Object o): Kiểm tra xem tập hợp có chứa phần tử không.

+size(): Trả về số lượng phần tử trong tập hợp.

+isEmpty(): Kiểm tra xem tập hợp có rỗng không.

+clear(): Xóa tất cả các phần tử trong tập hợp.

### 4. ****Iterators (Bộ lặp)****

Các bộ lặp cho phép bạn duyệt qua các phần tử trong một tập hợp. Các phương thức chính của Iterator bao gồm:

+hasNext(): Kiểm tra xem còn phần tử nào nữa không.

+next(): Lấy phần tử tiếp theo.

+remove(): Xóa phần tử hiện tại (tùy chọn).

### 5. ****Collections Utility Class****

java.util.Collections là một lớp tiện ích với các phương thức tĩnh để làm việc với các tập hợp. Ví dụ:

+sort(List<T> list): Sắp xếp một danh sách.

+reverse(List<?> list): Đảo ngược thứ tự các phần tử trong danh sách.

+shuffle(List<?> list): Xáo trộn các phần tử trong danh sách.

## II.Gneric

**Generics**: Tham số hóa kiểu dữ liệu trong Java, cho phép bạn viết các lớp, giao diện và phương thức với các loại dữ liệu chưa xác định khi biên dịch.

**Ưu điểm**:

+ Tạo cấu trúc dữ liệu tổng quát.

+ Tính linh hoạt và khả năng tái sử dụng mã.

+ Đảm bảo tính an toàn về kiểu trong thời gian biên dịch (phát hiện lỗi về kiểu dữ liệu ngay khi chương trình được biên dịch).

+ Không ảnh hưởng đến hiệu suất.

1. **Generic class**

+ Một lớp có thể được khai báo với một kiểu tham số, ví dụ:

public class GenericClass<T> {

private T value;

public void setValue(T value) {

this.value = value;

}

public T getValue() {

return value;

}

}

1. **Generic method**

public static <T> void printDataType(T inp) { System.out.println(inp.getClass().getSimpleName()); }

1. **Giới Hạn Kiểu**

**+hạn kiểu** (Bounded Types) cho phép bạn chỉ định giới hạn cho các kiểu dữ liệu mà lớp hoặc phương thức generic có thể chấp nhận.

public class BoundedTypeClass<T extends Number> {

private T number;

public void setNumber(T number) {

this.number = number;

}

public T getNumber() {

return number;

} }

1. **Ký Tự Đại Diện**

+ **Ký tự đại diện** (Wildcards) cho phép làm việc với các kiểu không xác định.

+ Đại diện cho bất kỳ kiểu nào:

public static void unbounded(List<?> list) {

for (Object element : list) {

System.out.println(element);

}

}

+ Đại diện cho kiểu T hoặc kiểu con của T:

public static void upperBounded(List<? extends Number> list) {

for (Number num : list) {

System.out.println(num);

}

}