**Collection Framework trong Java**

Java Collection Framework cung cấp một kiến trúc để lưu trữ và quản lý các nhóm đối tượng. Nó bao gồm nhiều giao diện như List, Set, Map và các lớp triển khai của chúng.

**1. List**

List là một giao diện trong Java, đại diện cho một tập hợp có thứ tự, cho phép chứa các phần tử trùng lặp. Thứ tự các phần tử được bảo toàn khi chèn vào.

**Một số lớp triển khai của List:**

* **ArrayList**: Là một mảng động, kích thước có thể thay đổi. Truy cập phần tử theo chỉ số nhanh, nhưng thêm/xóa phần tử ở giữa danh sách thì chậm.

List<String> list = new ArrayList<>();

list.add("Apple");

list.add("Banana");

list.add("Orange");

**Ví dụ:**

ArrayList<Integer> numbers = new ArrayList<>();

numbers.add(10); // Thêm phần tử vào danh sách

numbers.add(20);

System.out.println(numbers.get(1)); // Truy cập phần tử tại vị trí 1 (giá trị là 20)

**2. Set**

Set là một giao diện đại diện cho một tập hợp không chứa phần tử trùng lặp. Không đảm bảo thứ tự phần tử.

**Một số lớp triển khai của Set:**

* **HashSet**: Là một Set không cho phép phần tử trùng lặp và không bảo toàn thứ tự của các phần tử.

Set<String> set = new HashSet<>();

set.add("Apple");

set.add("Banana");

set.add("Apple"); // Phần tử trùng lặp sẽ không được thêm vào

**Ví dụ:**

HashSet<String> fruits = new HashSet<>();

fruits.add("Apple");

fruits.add("Banana");

fruits.add("Orange");

System.out.println(fruits.contains("Banana")); // Kiểm tra nếu "Banana" có trong tập hợp

**3. Map**

Map là một giao diện ánh xạ (mapping) giữa các khóa (key) và giá trị (value). Mỗi khóa chỉ ánh xạ tới một giá trị duy nhất. Các khóa không được trùng lặp, nhưng các giá trị có thể trùng lặp.

**Một số lớp triển khai của Map:**

* **HashMap**: Lưu trữ các cặp khóa-giá trị mà không bảo toàn thứ tự. Truy cập khóa nhanh nhưng không cho phép khóa trùng lặp.

Map<String, Integer> map = new HashMap<>();

map.put("Apple", 1);

map.put("Banana", 2);

**Ví dụ:**

HashMap<String, Integer> productPrices = new HashMap<>();

productPrices.put("Laptop", 1500);

productPrices.put("Phone", 800);

System.out.println(productPrices.get("Laptop")); // Trả về giá trị tương ứng với khóa "Laptop"

**Các Lớp Triển Khai Cụ Thể**

**1. ArrayList**

Là một lớp triển khai của List. ArrayList cung cấp một mảng động cho phép thay đổi kích thước.

**Cú pháp:**

ArrayList<String> list = new ArrayList<>();

Các phương thức phổ biến:

* add(E e): Thêm một phần tử vào danh sách.
* get(int index): Trả về phần tử tại vị trí chỉ định.
* remove(int index): Xóa phần tử tại vị trí chỉ định.
* size(): Trả về số phần tử trong danh sách.

**Ví dụ:**

ArrayList<String> animals = new ArrayList<>();

animals.add("Cat");

animals.add("Dog");

System.out.println(animals.size()); // Kết quả: 2

**2. HashSet**

Là một lớp triển khai của Set, HashSet sử dụng bảng băm (hash table) để lưu trữ các phần tử.

**Cú pháp:**

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

Các phương thức phổ biến:

* add(E e): Thêm phần tử vào Set.
* contains(Object o): Kiểm tra xem phần tử có tồn tại trong Set không.
* remove(Object o): Xóa phần tử khỏi Set.

**Ví dụ:**

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

set.add(1);

set.add(2);

set.add(2); // Phần tử trùng lặp sẽ bị bỏ qua

System.out.println(set.size()); // Kết quả: 2

**3. HashMap**

Là một lớp triển khai của Map, HashMap lưu trữ các cặp khóa-giá trị. Mỗi khóa chỉ ánh xạ đến một giá trị duy nhất.

**Cú pháp:**

HashMap<String, Integer> map = new HashMap<>();

Các phương thức phổ biến:

* put(K key, V value): Thêm hoặc cập nhật cặp khóa-giá trị.
* get(Object key): Trả về giá trị tương ứng với khóa.
* remove(Object key): Xóa cặp khóa-giá trị theo khóa.

**Ví dụ:**

HashMap<String, String> countries = new HashMap<>();

countries.put("USA", "Washington");

countries.put("France", "Paris");

System.out.println(countries.get("USA")); // Kết quả: "Washington"

**Generics trong Java**

Generics cho phép định nghĩa các lớp, phương thức có thể hoạt động với nhiều kiểu dữ liệu khác nhau mà không cần phải định nghĩa lại cho từng kiểu cụ thể. Điều này giúp tăng tính tái sử dụng mã và giảm lỗi.

**1. Tạo Lớp Sử Dụng Generics**

Khi khai báo lớp Generics, ta sử dụng các ký tự như T, E, K, V để biểu diễn các kiểu dữ liệu tổng quát.

**Cú pháp:**

class Box<T> {

private T value;

public void set(T value) {

this.value = value;

}

public T get() {

return value;

}

}

**Ví dụ:**

Box<Integer> intBox = new Box<>();

intBox.set(123);

System.out.println(intBox.get()); // Kết quả: 123

Box<String> strBox = new Box<>();

strBox.set("Hello");

System.out.println(strBox.get()); // Kết quả: "Hello"

**2. Tạo Phương Thức Sử Dụng Generics**

Phương thức Generics cho phép truyền các đối số của nhiều kiểu dữ liệu khác nhau.

**Cú pháp:**

public <T> void printArray(T[] array) {

for (T element : array) {

System.out.println(element);

}

}

**Ví dụ:**

Integer[] intArray = {1, 2, 3};

String[] strArray = {"Apple", "Banana"};

printArray(intArray); // In ra các số nguyên

printArray(strArray); // In ra các chuỗi

Trên đây là tổng quan về Collection Framework và Generics trong Java.