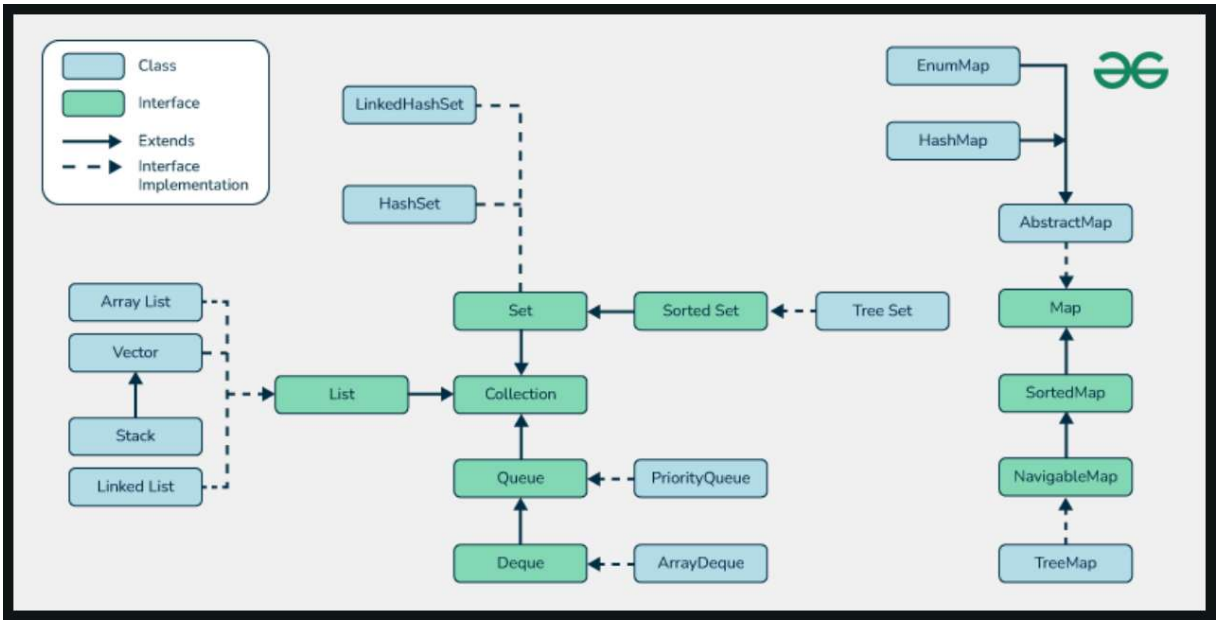
Cập nhật tháng 8 năm 2024

[Bài đọc] Giới thiệu Collection

1. Khái niệm về Collection

- **Collection** trong Java là một framework cung cấp một kiến trúc chuẩn để lưu trữ và thao tác trên nhóm các đối tượng (objects).
- Nó cung cấp các lớp và giao diện mạnh mẽ để làm việc với các tập hợp dữ liệu như danh sách (lists), tập hợp (sets), hàng đợi (queues), và bản đồ (maps).



2. Đặc điểm của Collection

- **Tính linh hoạt:** Hỗ trợ nhiều kiểu dữ liệu khác nhau.
- **Tự động mở rộng:** Hầu hết các lớp trong Collection có khả năng tự mở rộng khi thêm phần tử mới.
- **Xử lý hiệu quả:** Cung cấp các phương thức để thêm, sửa, xóa và tìm kiếm một cách tối ưu.
- **Sử dụng lại mã:** Collection được thiết kế để tái sử dụng trong nhiều ứng dụng khác nhau.
- **Generic Support:** Hỗ trợ kiểu tham số hóa (Generics) để đảm bảo an toàn kiểu tại thời điểm biên dịch.

3. **Các Interface chính trong Collection Framework**

Dưới đây là các giao diện cốt lõi mà Collection Framework cung cấp:

Interface	**Ý nghĩa**
Collection	Interface gốc của tất cả tập hợp
List	Tập hợp các phần tử có thứ tự, có thể chứa phần tử trùng lặp.
Set	Tập hợp các phần tử không trùng lặp, không có thứ tự cụ thể.
Queue	Cung cấp các thao tác FIFO (First In First Out).
Deque	Cung cấp các thao tác FIFO và LIFO (Last In First Out).
Map	Tập hợp các cặp key-value, không chứa các key trùng lặp.

4. **Phân loại Collection**

Collection trong Java có thể được phân loại như sau:

1. **Dựa trên kiểu lưu trữ**:
   - **List**:
     - Dùng để lưu các phần tử có thứ tự, có thể trùng lặp.
     - Các lớp phổ biến:
       - **ArrayList**: Sử dụng mảng động, tốc độ truy cập nhanh nhưng chèn và xóa chậm.
       - **LinkedList**: Sử dụng danh sách liên kết, tốc độ chèn và xóa nhanh hơn nhưng truy cập chậm.
       - **Vector**: Phiên bản đồng bộ của ArrayList.
   - **Set**:
     - Lưu trữ các phần tử không trùng lặp, không có thứ tự rõ ràng.
     - Các lớp phổ biến:
       - **HashSet**: Dựa trên bảng băm, không duy trì thứ tự.
       - **LinkedHashSet**: Duy trì thứ tự chèn vào.
       - **TreeSet**: Sắp xếp các phần tử theo thứ tự tự nhiên hoặc comparator.
2. **Dựa trên cách hoạt động**:
   - **Queue**: Dùng để xử lý dữ liệu theo nguyên tắc FIFO hoặc LIFO.
     - Lớp phổ biến: **PriorityQueue**.
   - **Map**: Lưu trữ dữ liệu theo cặp key-value.
     - Các lớp phổ biến:
       - **HashMap**: Không đảm bảo thứ tự key.
       - **LinkedHashMap**: Duy trì thứ tự chèn **key-value**.
       - **TreeMap**: Sắp xếp các key theo thứ tự tự nhiên hoặc comparator.

5. **So sánh giữa List, Set và Map**

Đặc điểm	List	Set	Map
Dữ liệu trùng lặp	Có thể trùng lặp	Không trùng lặp	Key không trùng lặp, value có thể trùng lặp
Thứ tự	Duy trì thứ tự chèn	Không đảm bảo thứ tự	Duy trì thứ tự tùy thuộc vào lớp triển khai
Cách truy cập	Qua chỉ số (index)	Qua các phần tử	Qua key

6. **Các phương thức cơ bản của Collection**

Một số phương thức thường được sử dụng trong Collection:

- Thêm phần tử:
  - **add(E e):** Thêm một phần tử.
- Xóa phần tử:
  - **remove(Object o):** Xóa phần tử được chỉ định.
- Duyệt phần tử:
  - **iterator():** Trả về một đối tượng Iterator để duyệt qua tập hợp.
- Kiểm tra phần tử:
  - **contains(Object o):** Kiểm tra xem tập hợp có chứa phần tử không.
- Kích thước:
  - **size():** Trả về số phần tử hiện có trong tập hợp.

7. **Ưu điểm của Collection Framework**

- **Tính nhất quán:** Các lớp và giao diện trong Collection Framework tuân theo một cấu trúc chuẩn.
- **Đa dạng:** Hỗ trợ nhiều loại cấu trúc dữ liệu khác nhau.
- **Tái sử dụng mã:** Việc sử dụng các lớp Collection giúp giảm số lượng mã cần viết.

8. **Ví dụ minh họa**

```java
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Map;
import java.util.HashMap;

public class CollectionExample {
    public static void main(String[] args) {
        // List Example
        ArrayList<String> list = new ArrayList<>();
        list.add("Java");
        list.add("Python");
        list.add("Java");
        System.out.println("List: " + list);

        // Set Example
        HashSet<String> set = new HashSet<>();
        set.add("Java");
        set.add("Python");
        set.add("Java");
        System.out.println("Set: " + set);

        // Map Example
        Map<Integer, String> map = new HashMap<>();
        map.put(1, "Java");
        map.put(2, "Python");
        map.put(1, "C++");
        System.out.println("Map: " + map);
    }
}
```

Kết quả:

```
List: [Java, Python, Java]
Set: [Java, Python]
Map: {1=C++, 2=Python}
```

Link tài nguyên đọc thêm: https://www.geeksforgeeks.org/collections-in-java-2/