## 

|  |
| --- |
| **HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  **KHOA CÔNG NGHỆ THÔNG TIN 1**  **Học Viện Công Nghệ Bưu Chính Viễn Thông (phía Nam) | School Reviews**  **BÀI TẬP LỚN**  **MÔN HỌC: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**  **Chủ đề: Giới thiệu thư viện Java Collection**  **NHÓM BTL: 8 – NHÓM: 02**  **Thành viên: Phạm Đức Lương Nguyễn Văn Minh**  **Lã Mạnh Tú Phan Ngọc Tuấn**  **Giảng viên: NGUYỄN MẠNH SƠN**  **Hà Nội năm 2020** |

**MỤC LỤC**

**I-Tổng quan……………………………………………………………………...3**

**1-Giới thiệu……………………………………………………………......3**

**1.1 Framework là gì?.......................................................................3**

**1.2 Tại sạo lại có Java Collection…………………………………3**

**1.3 Mục đích thiết kế………………………………………………3**

**1.4 Định nghĩa……………………………………………………..4**

**2- Các thành phân của Java Collection…………………………………4**

**3- Hệ thống phân cấp……………………………………………………5**

**II-Các thành phần chính ……………………………………………………….7**

### 1- Iterable interface………………………………………………………7

**2-Collection Interface…………………………………………………….7**

**3- List Interface…………………………………………………………...7**

**4- Queue Interface………………………………………………………..9**

**5- Set Interface……………………………………………………………11**

**6- Map Interface………………………………………………………….12**

**III- Cách lựa chọn khi sử dụng ………………………………………………15**

**1-Tại sao lại dùng Java Collection……………………………………..15**

**2-Lựa chọn khi sử dụng…………………………………………………15**

**I-Tổng quan**

**1-Giới thiệu**

Bất kì lập trình viên nào đã từng làm việc với Java hay Android có lẽ đều biết tới ArrayList – một class cực kì dễ dùng và tiện dụng. Nhưng có lẽ không nhiều người biết rằng ArrayList chỉ là một trong số rất nhiều class thuộc bộ thư viện Java Collection Framework của Java – một bộ thư viện với rất nhiều class mạnh mẽ giúp bạn đơn giản hóa các thao tác khi làm việc với tập hợp và đồ thị.

## **1.1-Framework là gì?**

**Framework**là các đoạn code đã được viết sẵn, cấu thành nên một bộ khung và các thư viện lập trình được đóng gói

Các **framework** giống như là chúng ta có khung nhà được làm sẵn nền móng cơ bản, bạn chỉ cần vào xây dựng và nội thất theo ý mình.

**1.2-Tại sạo lại có Java Collection**

**Collection** ra đời là để khắc phục những hạn chế, nhược điểm khi sử dụng mảng để lập trình.

Mảng có kích cỡ và số chiều cố định. Khó khăn cho việc mở rộng mảng

Các phần tử được đặt và tham chiếu một cách liên tiếp nhau trong bộ nhớ. Khó khăn cho việc thao tác với phần tử trong mảng .

**1.3-Mục đích thiết kế**

Collection framework được thiết kế với mục đích như sau:

1. hiệu năng cao. Sự triển khai cho các tập hợp cơ bản (các mảng động, linked list, tree và hashtable) được sử dụng với hiệu quả cao.
2. cho phép các kiểu tập hợp khác nhau làm việc theo một cách tương tự như nhau với độ phân hóa ở mức cao.
3. Kế thừa và tìm hiểu với các tập hợp dễ dàng.

**1.4-Định nghĩa**

**Java Collections là gì ?**

**Một Java Collections Framework là một tập hợp các lớp (class) và các interface dùng để hỗ trợ việc thao tác trên tập các đối tượng**

**Collections là một tập các lớp dùng để lưu trữ danh sách và có khả năng tự co dãn khi danh sách đó thay đổi**, ví dụ như khi chúng ta thêm, sửa, xóa, chèn phần tử trong danh sách đó

Ngoài ra, Collections còn được dùng để lưu trữ, truy xuất, tương tác với dữ liệu và truyền dữ liệu giữa các phương thức với nhau

**Một đặc điểm rất quan trong là khi sử dụng Collections đó là chúng ta không cần phải khai báo trước số lượng phần tử**. Chính đặc điểm này đã khắc phục được hạn chế về kích thước khi khai báo mảng trong Java.

**2-Các thành phân của Java Collection**

Gói Thư viện Java Collection bao gồm:

* **Interface:** Đây là các kiểu dữ liệu abstract mà biểu diễn collection. Interface cho phép collection được thao tác một cách độc lập theo phép biểu diễn của chúng.

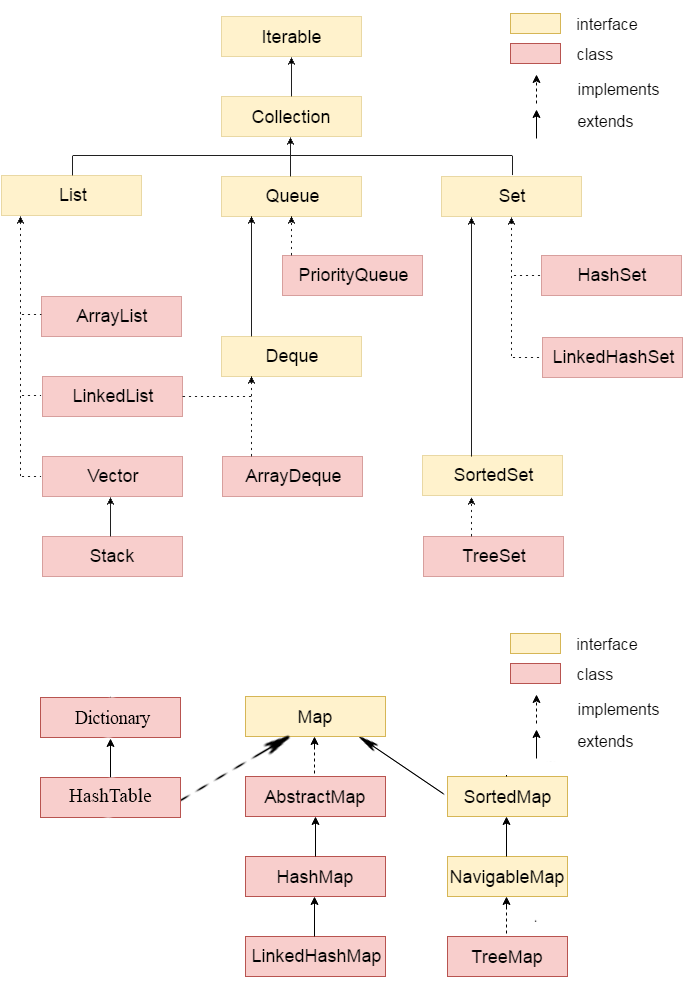
Trong ngôn ngữ hướng đối tượng, các interface nói chung cấu tạo nên một hierarchy- hệ thống cấp bậc.

* **Sự triển khai, ví dụ như các Class:** Đây là sự triển khai cụ thể của collection interface. Về bản chất, chúng là những cấu trúc dữ liệu có thể tái sử dụng.
* **Thuật toán:**Đây là các phương thức thực hiện các trình tính toán hữu ích, như tìm kiếm và xếp thứ tự phân loại, trên các đối tượng mà triển khai collection interface. Các thuật toán được xem như là đa hình: đó là, cùng một phương thức có thể được sử dụng trên nhiều sự triển khai khác nhau của collection interface thích hợp.

Ngoài ra, framework định nghĩa một số map interfaces và class. Map lưu giữ các cặp key/value. Mặc dù các map không là collections về khái niệm, nhưng chúng hoàn toàn tương thích với collection.

**3- Hệ thống phân cấp**

Hệ thống phân cấp của chúng được mô tả tổng quát như bên dưới:

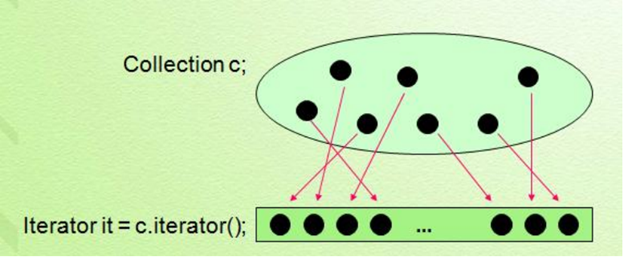


**II-Các thành phần chính**

### 1- Iterable interface

Cấp cao nhất của Collection là **Iterable interface**, nó chứa dữ liệu thành viên **Iterator interface**.

Giao tiếp Iterator cung cấp phương tiện để lặp đi lặp lại các thành phần từ đầu đến cuối của một collection.



**Hệ thống phân cấp tiếp theo dẫn đầu bởi 2 interface Collection và Map**

**2-Collection Interface**

**Định nghĩa:**Là 1 interface, nó cung cấp các phương thức hay dùng để thao tác trên các dạng và cấu trúc dữ liệu của nhóm Collection.

Là interface nền móng để từ đó xây dung lên cả bộ thư viện Java Collections Framework.

Collection interface được kế thừa từ Iterable Interface nên chúng ta dễ dàng duyệt qua từng phần tử thông qua việc sử dụng Iterator.

Có 3 nhánh con trong nhóm Collection:***List***, **Queue**, **Set**

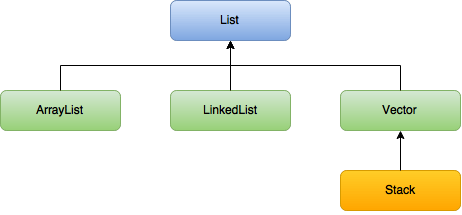
**3- List Interface**

**List:**là một collection có thứ tự (đôi khi còn được gọi là một chuỗi). List có thể chứa các phần tử trùng lặp. Thường có quyền kiểm soát chính xác vị trí các phần tử được chèn vào và có thể truy cập chúng bằng chỉ số (vị trí của chúng).

List Interface có một số đặc điểm như sau:

1. Các phần tử có thể được chèn hoặc được truy cập thông qua vị trí của chúng trong danh sách, bởi sử dụng chỉ mục xây dựng bắt đầu từ 0.
2. Một list có thể chứa nhiều bản sao phần tử.

Vì được kế thừa từ Collection interface nên nó có đầy đủ các phương thức của Collection Interface.

****

Một số class thực thi List Interface thường sử dụng:

* **ArrayList**: là 1 class dạng list được implement dựa trên mảng có kích thước thay đổi được.

Có thể chứa các phần tử trùng lặp

Duy trì thứ tự chèn

Không đồng bộ

Truy các cập phần tử nhanh, thêm xóa chậm

Không sử dụng cho các kiểu dữ liệu nguyên thủy

* **LinkedList**: là một class dạng list hoạt động trên cơ sở của cấu trúc dữ liệu danh sách liên kết đôi (double-linked list)

Có thể chứa các phần tử trùng lặp.

Duy trì thứ tự chèn.

Không đồng bộ.

Truy xuất phần tử chậm hơn so với ArrayList.

Việc thêm, xóa dữ liệu nhanh hơn vì không cần dịch chuyển các phần tử.

Nên sử dụng LinkedList trong trường hợp cần nhiều thao tác thêm xóa phần tử.

* **Vector**: là 1 class thực thi giao diện List Interface, có cách thực lưu trữ như mảng tuy nhiên có kích thước thay đổi được, khá là tương tự với ArrayList, tuy nhiên điểm khác biệt là Vector là synchronized, hay là đồng bộ, có thể hoạt động đa luồng mà không cần gọi synchronize một cách tường minh
* **Stack:** cũng là 1 class dạng list, Stack có cách hoạt động dựa trên cơ sở của cấu trúc dữ liệu ngăn xếp (stack) với kiểu vào ra LIFO (last-in-first-out hay vào sau ra trước) nổi tiếng.

**4- Queue Interface**

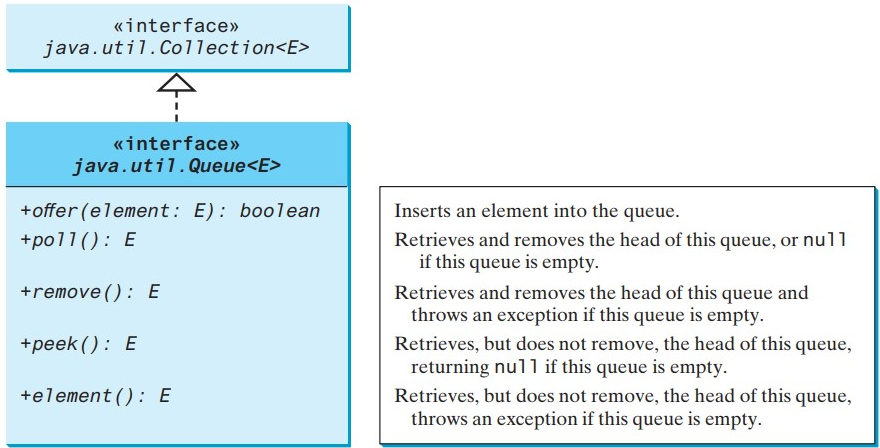
Queue (hàng đợi) là kiểu dữ liệu vào ra FIFO (first-in-first-out hay vào trước ra trước). Bên cạnh các thao tác cơ bản của collection, Queue cung cấp các thao tác bổ sung như chèn, lấy ra và kiểm tra.

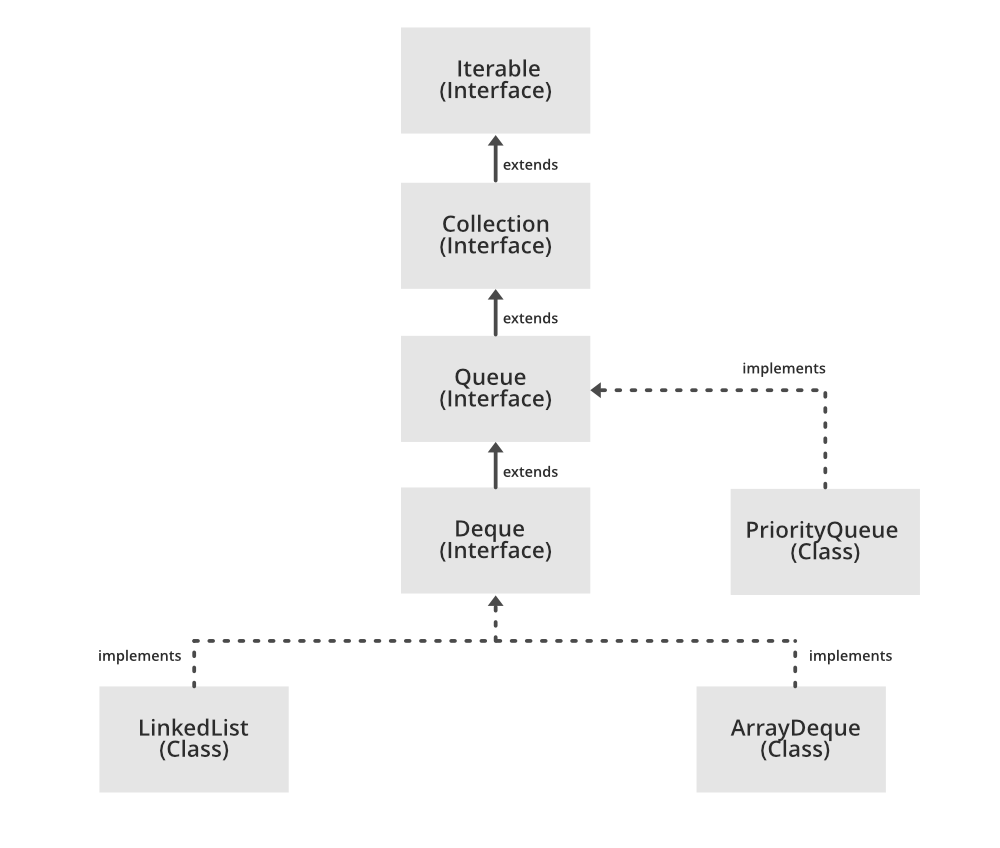
Queue có các đặc điểm cơ bản sau:

1. Là tập hợp cho phép các phần tử trùng lặp.
2. Không cho phép phần tử null.



**Và cũng giống List Interface , Queue Interface cũng kế thừa và mang đầy đủ phương thức từ Collection Interface. Cũng với 1 số phương thức riêng khác :**





**Hệ thống cấp bậc của queue**

 Một số class về Queue thường sử dụng:

**LinkedList**: chính là LinkedList mình đã nói ở phần List

LinkedList là một hàng đợi khá chuẩn. Nhưng nhớ rằng LinkedList thi hành cả 2 interface List và Queue.

Nếu bạn coi nó như một hàng đợi thì hãy sử dụng cách thức truy cập vào phần tử của hàng đợi.

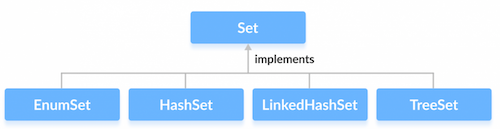
**PriorityQueue**: là 1 dạng queue mà trong đó các phần tử trong queue sẽ được sắp xếp.

**ArrayDeque**: là 1 dạng deque (queue 2 chiều) được implement dựa trên mảng

**5- Set Interface**

Set (tập hợp) là kiểu dữ liệu mà bên trong nó mỗi phần tử chỉ xuất hiện duy nhất một lần (tương tự như tập hợp trong toán học vậy)

Set Interface cung cấp các phương thức để tương tác với set. Set Interface được kế thừa từ Collection Interface nên nó cũng có đầy đủ các phương thức của Collection Interface

****

Set được triển khai bởi Hashset, LinkedHashset, Treeset hoặc EnumSet.

* **HashSet** lưu trữ các phần tử của nó trong bảng băm, là cách thực hiện tốt nhất, tuy nhiên nó không đảm bảo về thứ tự các phần tử được chèn vào.
* **TreeSet** lưu trữ các phần tử của nó trong một cây, sắp xếp các phần tử của nó dựa trên các giá trị của chúng, về cơ bản là chậm hơn HashSet.
* **LinkedHashSet** được triển khai dưới dạng bảng băm với có cấu trúc dữ liệu danh sách liên kết, sắp xếp các phần tử của nó dựa trên thứ tự chúng được chèn vào tập hợp (thứ tự chèn).
* **EnumSet** là một cài đặt chuyên biệt để sử dụng với các kiểu enum.

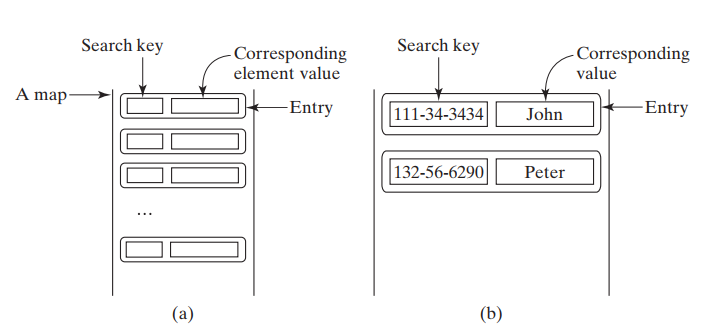
**6- Map Interface**

Map (đồ thị/ánh xạ) là kiểu dữ liệu cho phép ta quản lý dữ liệu theo dạng cặp key-value, trong đó key là duy nhất và tương ứng với 1 key là một giá trị value.

Map được sử dụng trong trường hợp chúng ta muốn truy xuất, cập nhật hoặc tìm kiếm phần tử thông qua khóa của phần tử đó. Ví dụ:

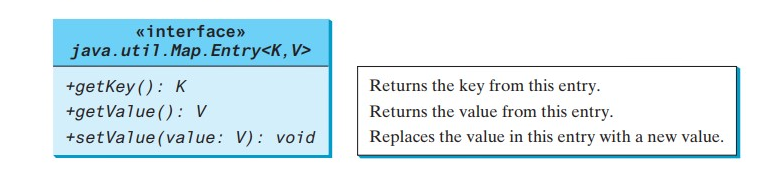
• Một Map bao gồm thông tin của người quản lý và nhân viên trong một công ty. Mỗi một người quản lý (key) sẽ liên kết với danh sách các nhân viên mà người đó quản lý (value).

• Một Map bao gồm thông tin của một lớp học và các sinh viên có trong lớp đó. Mỗi một lớp (key) sẽ liên kết với danh sách các sinh viên của lớp đó (value)



## **- Map.Entry Interface**

Entry là 1 interface con của Map. Được sử dụng để lấy ra giá trị các key-value trong map.



Map Interface cung cấp cho ta các phương thức để tương tác với kiểu dữ liệu như vậy. Không giống như các interface ở trên, Map Interface không kế thừa từ Collection Interface mà đây là 1 interface độc lập với các phương thức của riêng mình

Java Map - javatpoint

Hệ thống cấp bậc của Map

**TreeMap**: là class thực thi giao diện Map Interface với dạng cây đỏ đen (Red-Black tree) trong đó các key đã được sắp xếp. Class này cho phép thời gian thêm, sửa, xóa và tìm kiếm 1 phần tử trong Map là tương đương nhau và đều là O(log(n))

* Nó chứa các key duy nhất.
* Nó KHÔNG cho phép bất kỳ key nào là null và nhưng có thể có nhiều giá trị null.
* Nó duy trì các phần tử được thêm vào theo thứ tự key tăng dần.

**HashMap**: là class thực thi giao diện Map Interface với các key được lưu trữ dưới dạng bảng băm, cho phép tìm kiếm nhanh O(1).

* Nó chứa các key duy nhất.
* Nó có thể có 1 key là null và nhiều giá trị null.
* Nó duy trì các phần tử KHÔNG theo thứ tự.

**LinkedHashMap**:

* LinkedHashMap lưu trữ dữ liệu dưới dạng cặp key và value.
* Nó chứa các key duy nhất.
* Nó có thể có 1 key là null và nhiều giá trị null.
* Nó duy trì các phần tử theo thứ tự chèn.

**EnumMap**: cũng là 1 Map class nữa, tuy nhiên các key trong Map lại là các enum chứ không phải object như các dạng Map class ở trên.

**WeakHashMap**: tương tự như HashMap tuy nhiên có 1 điểm khác biệt đáng chú ý là các key trong Map chỉ là các Weak reference (hay Weak key), có nghĩa là khi phần tử sẽ bị xóa khi key được giải phóng hay không còn một biến nào tham chiếu đến key nữa.

**III- Cách lựa chọn khi sử dụng**

**1-Tại sao lại dùng Java Collection**

Chúng ta không phải viết code để thực hiện các cấu trúc dữ liệu và thuật toán theo cách thủ công.

Code của chúng ta sẽ hiệu quả hơn nhiều vì Collections Framework được tối ưu hóa cao.

Hơn nữa, Collections Framework cho phép chúng ta sử dụng cấu trúc dữ liệu cụ thể cho một loại dữ liệu cụ thể để tối ưu hóa. Đây là vài ví dụ:

**+** Nếu chúng ta muốn dữ liệu của mình là duy nhất, thì chúng ta có thể sử dụng Set Interface.

**+** Để lưu trữ dữ liệu theo cặp key / value, chúng ta có thể sử dụng Map Inteface.

**+** Class ArrayList (trong List) làm cho các mảng có thể thay đổi kích thước.

**2-Lựa chọn khi sử dụng**

* **List :**

ArrayList trong trường hợp cần lưu trữ và truy cập dữ liệu

LinkedList trong trường hợp cần nhiều thao tác thêm xóa phần tử

Sử dụng vector để duy trì thứ tự của phần tử được thêm vào và cần hoạt động đa luồng

Stack sử dụng trong các trường hợp để giải quyết các vấn đề hoặc bài toán cụ thể đảo ngược xâu kí tự, đổi cơ số, tính giá trị của biểu thức số học.

**-Queue:**

**Vì hoạt động theo kiểu LIFO nên nó sử dụng giải quyết các bài toán và vấn đề cụ thể**

**Riêng với LinkedList** thi hành cả 2 interface List và Queue.

* Nếu bạn coi nó như một hàng đợi thì hãy sử dụng cách thức truy cập vào phần tử của hang đợi.

**-Set :**

Nếu ưu tiên về tốc độ sử dụng, không quan tâm thứ tự HashSet

Nếu muốn một Set theo thứ tự mà các phần tử được insert vào thì dùng LinkedHashSet.

Nếu cần một Set được sắp xếp chọn TreeSet

**-Map:**

Nếu ưu tiên về tốc độ sử dụng, không quan tâm thứ tự HashMap

Nếu muốn một Map theo thứ tự mà các phần tử được insert vào thì dùng LinkedHashSet.

Nếu cần một Map được sắp xếp chọn TreeMap

**TÀI LIỆU THAM KHẢO**

* **Introduction to Java Programming and Data Structures**
* **Collections in Java** [**https://www.javatpoint.com/collections-in-java**](https://www.javatpoint.com/collections-in-java)
* **Collections in Java** [**https://www.geeksforgeeks.org/collections-in-java-2/**](https://www.geeksforgeeks.org/collections-in-java-2/)
* **Java - Collections Framework** [**https://www.tutorialspoint.com/java/java\_collections.htm**](https://www.tutorialspoint.com/java/java_collections.htm)
* **Collection trong java** [**https://gpcoder.com/2493-collection-trong-java/**](https://gpcoder.com/2493-collection-trong-java/)
* **Collection trong java** [**https://viettuts.vn/java-collection**](https://viettuts.vn/java-collection)
* **Java Collections – Interface, List, Queue, Sets in Java With Examples**

[**https://www.edureka.co/blog/java-collections/**](https://www.edureka.co/blog/java-collections/)