# Cấu trúc dữ liệu trong Java

Các cấu trúc dữ liệu cung cấp bởi các package tiện ích của Java rất mạnh mẽ và thực hiện các tính năng rộng rãi. Những cấu trúc dữ liệu này bao gồm những interface và class.

- Enumeration
- BitSet
- Vector
- Stack
- Dictionary
- Hashtable
- Properties

Tất cả các lớp trên được giới thiệu bởi một framework mới với tên là Collection Framework, được thảo luận ở chương tiếp theo.

Để hiểu sâu hơn các khái niệm được trình bày trong chương này, mời bạn tham khảo loạt bài: **Ví dụ về Cấu trúc dữ liệu (Data Structure) trong Java**.

## Lớp Enumeration trong Java

Interface Enumeration bản thân nó không phải là cấu trúc dữ liệu, nhưng rất quan trong bên trong ngữ cảnh sử dụng các cấu trúc dữ liệu khác. Interface Enumeration định nghĩa để nhận các thành phần kế tiếp từ cấu trúc dữ liệu.

Ví dụ, Enumeration định nghĩa phương thức gọi là nextElement được sử dụng để lấy các thành phần tiếp theo trong cấu trúc dữ liệu chứa nhiều thành phần.

Để tìm hiểu chi tiết về interface này, bạn truy cập link sau: **Enumeration interface trong Java**.

## Lớp BitSet trong Java

Lớp BitSet trong Java triển khai một nhóm các bit hoặc flag mà có thể được thiết lập và xóa một cách riêng rẽ.

Class này rất hữu dụng trong trường hợp bạn muốn lưu trữ một tập các giá trị Boolean và chỉ muốn gắn từng bit các giá trị và thiết lập hoặc xóa nó thích hợp.

Để tìm hiểu chi tiết về class này, bạn truy cập link sau: **Lớp BitSet trong Java**.

# Lớp Vector trong Java

Lớp Vector trong Java là tương tự như các mảng dữ liệu Java truyền thống, ngoại trừ việc có thể tăng lưu trữ cho các thành phần mới.

Giống như mảng, các thành phần trong đối tượng Vector có thể truy cập bởi index.

Một điều tốt về việc sử dụng Vector là bạn không phải lo lắng về việc cài đặt nó cho một kích cỡ cụ thể ngoài việc tạo ra nó, nó có thể tăng và giảm độ lớn khi cần thiết.

Để tìm hiểu chi tiết về class này, bạn truy cập link sau: **Lớp Vector trong Java**.

# Lớp Stack trong Java

Lớp Stack trong Java triển khai một last-in-first-out (LIFO) stack các phần tử.

Bạn có thể nghĩ về stack như một ngăn xếp thẳng đứng các đối tượng, khi bạn thêm một đối tượng mới, bạn lấy nó ở phần đầu các thành phần khác.

Khi bạn lấy một thành phần trên stack, nó lấy từ trên đỉnh xuống. Theo cách nói khác, thành phần cuối cùng mà bạn thêm vào stack sẽ là thành phần đầu tiên khi lấy ra và ngược lại.

Để tìm hiểu chi tiết về class này, bạn truy cập link sau: **Lớp Stack trong Java**.

# Lớp Dictionary trong Java

Lớp Dictionary là một abstract class để định nghĩa cấu trúc dữ liệu cho việc liên kết giữa các key tới value.

Nó thực sự hữu ích trong các trường hợp khi bạn muốn có thể truy cập dữ liệu thông qua một key cụ thể thay vì sử dụng một integer index.

Khi lớp Dictionary là abstract, nó chỉ cung cấp framework cho một cấu trúc dữ liệu so khớp key thay vì một sự triển khai cụ thể.

Để tìm hiểu chi tiết về class này, bạn truy cập link sau: **Lớp Dictionary trong Java**.

# Lớp Hashtable trong Java

Lớp Hashtable cung cấp các ý nghĩa về mặt tổ chức dữ liệu dựa vào cấu trúc mà người dùng định nghĩa key.

Ví dụ, một danh sách địa chỉ bạn có thể lưu trữ và xếp thứ tự dựa và key như zip code hơn là việc sử dụng tên người.

Ý nghĩa đặc trưng của các key liên quan tới hashtable là hoàn toàn phụ thuộc vào hashtable và dữ liệu nó chứa.

Để tìm hiểu chi tiết về class này, bạn truy cập link sau: **Lớp Hashtable trong Java**.

# Lớp Properties trong Java

Lớp properties là lớp con của Hashtable. Nó được sử dụng để duy trì danh sách các giá trị trong đó key là String và value cũng là một String.

Lớp Properties được sử dụng bởi nhiều class khác trong Java. Ví dụ, bạn có một kiểu đối tượng trả về bởi System.getProperties() để lấy về các biến môi trường.

Để tìm hiểu chi tiết về class này, bạn truy cập link sau: **Lớp Properties trong Java**.