Array

**1. Hiểu biết về kiểu mảng (Array) trong Java**

**a. Định nghĩa:**

* **Mảng** là một **cấu trúc dữ liệu** trong Java, cho phép lưu trữ **nhiều phần tử có cùng kiểu dữ liệu** trong một biến duy nhất. Mảng trong Java có **kích thước cố định**, nghĩa là khi mảng đã được khởi tạo, bạn không thể thay đổi kích thước của nó.
* Mảng có **chỉ số bắt đầu từ 0**, nghĩa là phần tử đầu tiên trong mảng có chỉ số là 0, và phần tử cuối cùng có chỉ số là n-1 với n là số phần tử trong mảng.

**b. Cú pháp khai báo mảng:**

Mảng trong Java có thể khai báo cho các kiểu dữ liệu nguyên thủy (int, float, double, char,...) hoặc các kiểu đối tượng (String, Object,...).

1. **Khai báo và khởi tạo mảng**:

int[] array = new int[5]; // Mảng chứa 5 phần tử kiểu int, tất cả giá trị ban đầu là 0

**2. Khai báo và khởi tạo đồng thời:**

int[] array = {1, 2, 3, 4, 5}; // Mảng chứa 5 phần tử với giá trị cụ thể

**3. Truy cập và cập nhật giá trị phần tử mảng**:

array[0] = 10; // Gán giá trị 10 cho phần tử đầu tiên của mảng

int x = array[0]; // Truy cập phần tử đầu tiên của mảng

**c. Các loại mảng:**

* **Mảng một chiều (1D array)**: Mảng chỉ có một dòng các phần tử. Ví dụ: int[] array = {1, 2, 3};
* **Mảng hai chiều (2D array)**: Mảng chứa các phần tử dạng mảng con, thường được coi như bảng hoặc ma trận.

**d. Kích thước của mảng:**

Mảng trong Java có **kích thước cố định** khi được khởi tạo, nghĩa là số phần tử trong mảng không thể thay đổi sau khi mảng đã được tạo. Bạn có thể truy cập kích thước mảng thông qua thuộc tính .length.

**2. Ưu và nhược điểm của kiểu mảng trong Java**

**a. Ưu điểm:**

1. **Truy cập nhanh**:
   * Mảng cung cấp khả năng truy cập **ngẫu nhiên** (random access) đến các phần tử. Bạn có thể truy cập một phần tử bất kỳ trong mảng thông qua chỉ số (index), vì vậy việc truy xuất và thay đổi giá trị của phần tử rất nhanh.
   * Ví dụ: array[3] sẽ trả về phần tử thứ 4 trong mảng, và quá trình này thực hiện trong **O(1)** thời gian.
2. **Dễ sử dụng**:
   * Mảng trong Java dễ khai báo và sử dụng, giúp quản lý nhiều biến có cùng kiểu dữ liệu dễ dàng hơn.
3. **Lưu trữ liên tục**:
   * Các phần tử của mảng được lưu trữ liên tục trong bộ nhớ, điều này giúp tối ưu hóa việc truy cập dữ liệu và cải thiện hiệu năng khi làm việc với khối lượng dữ liệu lớn.
4. **Sử dụng ít bộ nhớ**:
   * Mảng không yêu cầu thêm không gian bộ nhớ cho các cấu trúc phụ trợ như các cấu trúc dữ liệu động khác (ví dụ: danh sách liên kết - linked list).

**b. Nhược điểm:**

1. **Kích thước cố định**:
   * Kích thước của mảng phải được xác định khi khởi tạo và không thể thay đổi. Điều này dẫn đến việc bạn có thể phải cấp phát dư thừa bộ nhớ nếu không biết trước kích thước chính xác hoặc không thể mở rộng mảng khi cần thêm phần tử.
   * Ví dụ, nếu bạn cần thêm phần tử vào một mảng đã đầy, bạn phải tạo một mảng mới có kích thước lớn hơn và sao chép dữ liệu từ mảng cũ.
2. **Không kiểm tra ranh giới**:
   * Java không kiểm soát chặt chẽ việc truy cập ngoài giới hạn của mảng (index out of bounds), ngoại trừ việc ném ngoại lệ ArrayIndexOutOfBoundsException. Nếu bạn cố gắng truy cập một chỉ số không hợp lệ, chương trình sẽ bị dừng do lỗi.

int[] array = {1, 2, 3};

System.out.println(array[5]); // Sẽ ném ra ArrayIndexOutOfBoundsException

3 **Thiếu tính linh hoạt**:

* Mảng thiếu tính năng linh hoạt như các cấu trúc dữ liệu khác (ví dụ: ArrayList, LinkedList) khi thao tác với dữ liệu động, như thêm, xóa phần tử hoặc thay đổi kích thước mảng.

4 **Không thể lưu trữ các loại dữ liệu khác nhau**:

* Mảng chỉ có thể lưu trữ các phần tử **cùng loại dữ liệu**. Nếu bạn cần lưu trữ các phần tử khác loại, bạn sẽ cần sử dụng các cấu trúc dữ liệu khác như ArrayList hoặc Object.