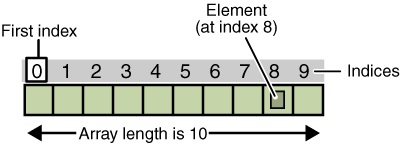
**Mảng**

Mảng (array) là một tập hợp các phần tử có cùng kiểu được lưu trữ gần nhau trong bộ nhớ

Mảng trong java là một đối tượng chứa các phần tử có kiểu dữ liệu giống nhau. Mảng là một cấu trúc dữ liệu nơi lưu trữ các phần tử giống nhau. Với mảng trong java chúng ta chỉ có thể lưu trữ một tập các phần tử có số lượng phần tử cố định.

Mảng trong java lưu các phần tử theo chỉ số, chỉ số của phần tử đầu tiên là 0.



Việc sử dụng **mảng** mang ý nghĩa lưu những giá trị liên quan với nhau. Ví dụ như lưu điểm kiểm tra của 30 học sinh trong lớp, như vậy ta muốn lấy điểm của học sinh nào đó thì chỉ cần viết vị trí của học sinh đó trong danh sách.

**Ưu điểm:**

* Tối ưu code: Gom các phần tử liên quan vào chung một với nhau giúp code gọn gàng hơn.
* Có thể truy cập ngấu nhiên: Do các vị trí ô lưu trữ liên tiếp ta có thể truy cập ngấu nhiên bằng chỉ số phần tử dễ dàng và nhanh chóng.
* Dễ thao tác, quản lý và nâng cấp: Như muốn thay đổi các giá trị theo 1 quy luật thì ta sẽ tận dụng sử dụng những vòng lặp lập trình.

**Nhược điểm:**

* Giới hạn kích thước: Khi sử dụng mảng ta phải khai báo kích thước lưu trữ của mảng và không thể thay đổi kích thước trong lúc chạy.
* Vùng lưu trữ phải liên tiếp: Đây cũng là vừa ưu vừa nhược điểm. Vì yêu cầu các ô nhớ liên tiếp nên phải tốn không gian bộ nhớ, hoặc đủ ô nhớ nhớ nhưng các ô nhớ không tiếp nên không thể khai báo được. Ưu điểm là vì các ô nhớ liên tiếp nhau nên tốc độ truy xuất nhanh.

**Cấu trúc của mảng**

**Mảng một chiều**

**Khai báo:**

Cú pháp khai báo:

**datatype**[] arr1;

**datatype** arr2[];

**datatype** []arr3;

Cú pháp cấp phát bộ nhớ để tạo mảng:

***<tên mảng> = new <* datatype *>[kích cỡ mảng];***

Cú pháp rút gọn hơn:

**<datatype > [] <tên mảng> = new < datatype >[kích cỡ mảng];**

Cú pháp khởi tạo cho mảng:

**< datatype > [] <tên mảng> = {<giá trị>,…}**

**Ví dụ:**

**public** **class** **HelloWorld**{

**public** **static** **void** **main**(String []args){

**char**[] a = {'H', 'o', 'w', 'K','t','e','a','m'};

System.out.print(a);

}

}

Riêng kiểu dữ liệu **char**đặc biệt hơn là có thể in ra toàn bộ giá trị bằng cách print trực tiếp mảng.

**Truy cập phần tử của một mảng**

Bạn truy cập một phần tử mảng bằng cách sử dụng số chỉ mục (index).

Ví dụ:

|  |  |
| --- | --- |
|  | public class TruyCapArray1 {      public static void main(String[] args) {          String[] cars = { "Honda", "BMW", "Ford", "Mazda" };          System.out.println(cars[0]);      }  } |

Kết quả:

Honda

**Thay đổi một phần tử trong mảng Java**

Để thay đổi giá trị của một phần tử cụ thể, hãy sử dụng số chỉ mục:

Ví dụ:

package vn.viettuts.array;

public class TruyCapArray2 {

    public static void main(String[] args) {

        String[] cars = { "Honda", "BMW", "Ford", "Mazda" };

        // thay đổi phần tử đầu tiên của mảng cars

        cars[0] = "Morning";

        // hiển thị phần tử đầu tiên của mảng cars

        System.out.println("Phần tử đầu tiên: " + cars[0]);

    }

}

Kết quả:

Phần tử đầu tiên: Morning

**Độ dài mảng trong Java**

Để biết có bao nhiêu phần tử một mảng, sử dụng thuộc tính **length**:

Ví dụ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | package vn.viettuts.array;    public class DodaiArray1 {      public static void main(String[] args) {          String[] cars = { "Honda", "BMW", "Ford", "Mazda" };          System.out.println("Độ dài của mảng cars là: " + cars.length);      }  } |

Kết quả:

4

**Duyệt các phần tử của mảng trong Java**

Có 2 các để duyệt các phần tử của mảng:

* Sử dụng vòng lặp for
* Sử dụng foreach

**Sử dụng vòng lặp for**

Bạn có thể lặp qua các phần tử mảng bằng vòng lặp for và sử dụng thuộc tính length để chỉ định số lần vòng lặp sẽ chạy.

Ví dụ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | package vn.viettuts.array;    public class DuyetArray1 {      public static void main(String[] args) {          String[] cars = { "Honda", "BMW", "Ford", "Mazda" };          for (int i = 0; i < cars.length; i++) {              System.out.println(cars[i]);          }      }  } |

Kết quả:

Honda

BMW

Ford

Mazda

**Sử dụng foreach**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | package vn.viettuts.array;    public class DuyetArray2 {      public static void main(String[] args) {          String[] cars = { "Honda", "BMW", "Ford", "Mazda" };          for (String **car**: cars) {              System.out.println(**car**);          }      }  } |

Kết quả:

Honda

BMW

Ford

Mazda

**So sánh for với foreach**

Nếu bạn so sánh vòng lặp for và vòng lặp foreach, bạn sẽ thấy rằng phương thức foreach dễ viết hơn, nó không yêu cầu bộ đếm (sử dụng thuộc tính length) và nó dễ đọc hơn.

**Sắp xếp mảng**

Có nhiều phương thức mảng có sẵn, ví dụ Sort(), sắp xếp một mảng theo thứ tự bảng chữ cái hoặc theo thứ tự tăng dần, ví dụ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package vn.viettuts.array;    import java.util.Arrays;    public class DuyetArray2 {      public static void main(String[] args) {          String[] cars = { "Honda", "BMW", "Ford", "Mazda" };          // sap xep mangr cars theo thu tu tang dan          Arrays.sort(cars);          System.out.println("Mảng cars sau khi được sắp xếp:");          for (String car : cars) {              System.out.println(car);          }      }  } |

Kết quả:

Mảng cars sau khi được sắp xếp:

BMW

Ford

Honda

Mazda

**Các cách khác để tạo một mảng trong Java**

Trong Java, có nhiều cách khác nhau để tạo một mảng:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // tạo một mảng có độ dài 4, thêm các phần tử sau tạo  String[] cars1 = new String[4];    // tạo một mảng không cần chỉ định số phần tử cụ thể  String[] cars2 = new String[] { "Honda", "BMW", "Ford", "Mazda" };    // tạo một mảng không cần chỉ định số phần tử cụ thể  // và không cần dùng từ khóa new  String[] cars3 = { "Honda", "BMW", "Ford", "Mazda" }; |

Bạn có thể chọn bất kỳ cách khởi tạo nào. Trong hướng dẫn của chúng tôi, chúng tôi sẽ thường sử dụng tùy chọn cuối cùng, vì nó nhanh hơn và dễ đọc hơn.

Một trường hợp khác, nếu bạn khai báo một mảng và khởi tạo nó sau, bạn phải sử dụng từ khóa new:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | // khai báo một mảng  String[] cars;    // thêm giá trị, sử dụng từ khóa new  cars = new String[] { "Honda", "BMW", "Ford" };    // nếu không dùng new sẽ bị lỗi  cars = {"Honda","BMW","Ford"}; |

**Truyền mảng vào phương thức trong java**

Chúng ta có thể truyền mảng vào phương thức, điều này giúp tái sử dụng code logic để xử lý mảng bất kỳ. Ví dụ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | public class TestArray2 {      static void min(int arr[]) {          int min = arr[0];          for (int i = 1; i < arr.length; i++)              if (min > arr[i]) {                  min = arr[i];              }          System.out.println(min);      }        public static void main(String args[]) {          int a[] = { 33, 3, 4, 5 };          min(a);// truyền mảng tới phương thức      }  } |

Kết quả:

3

**Sao chép một mảng trong java**

Chúng ta có thể sao chép một mảng tới mảng khác bởi phương thức arraycopy của lớp System. Cú pháp của phương thức arraycopy:

|  |  |
| --- | --- |
| 1  2  3 | public static void arraycopy(  Object src, int srcPos,Object dest, int destPos, int length  ) |

Ví dụ về sao chép mảng trong java:

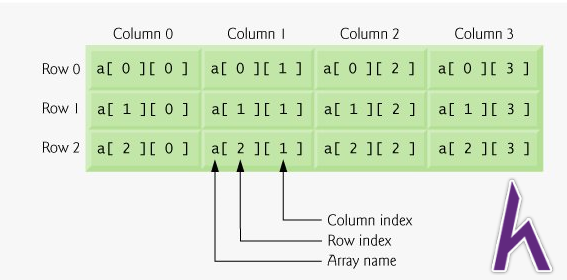
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | public class TestArrayCopyDemo {      public static void main(String[] args) {          char[] copyFrom = { 'd', 'e', 'c', 'a', 'f',                  'f', 'e', 'i', 'n', 'a', 't', 'e', 'd' };          char[] copyTo = new char[7];            System.arraycopy(copyFrom, 2, copyTo, 0, 7);          System.out.println(new String(copyTo));      }  } |

Kết quả:

caffein

**Mảng hai chiều và đa chiều trong Java**

Cũng như mảng một chiều dùng để lưu các giá trị có nét tương đồng. Thì mảng đa chiều chỉ là **tăng số chiều lưu trữ nhiều chiều hơn**, hay còn gọi là ma trận. Thường thường ta hay sử dụng mảng 2 chiều. Trong ma trận 2 chiều, ta hay gọi chiều thứ 1 là hàng, còn chiều thứ 2 là cột.



Cú pháp khai báo:

**<kiểu dữ liệu> [][] <tên mảng> = new <kiểu dữ liệu>[kích cỡ hàng] [kích cỡ cột];**

Cú pháp khởi tạo cho mảng:

**<kiểu dữ liệu> [][] <tên mảng> = {{các giá trị hàng 1}, {các giá trị hàng 2},… {các giá trị hàng n}}**

**Ví dụ**: Tạo ma trận 3 hàng 2 cột và in các giá trị ra theo hàng và cột.

**public** **class** **HelloWorld**{

**public** **static** **void** **main**(String []args){

**int**[][] a = {{1,2,3},{4,5,6},{7,8,9}};

**for** (**int** i=0; i<3; i++){

**for**(**int** j=0; j<3; j++){

System.out.print(a[i][j]+" ");

}

System.out.println();

}

}

}

**Khai báo mảng đa chiều trong java**

|  |  |
| --- | --- |
| 1  2  3  4 | dataType[][] arr;  dataType [][]arr;  dataType arr[][];  dataType []arr[]; |

Ví dụ về khởi tạo và gán giá trị cho mảng đa chiều trong java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | int[][] arr=new int[3][3]; // 3 hàng 3 cột  arr[0][0]=1;  arr[0][1]=2;  arr[0][2]=3;  arr[1][0]=4;  arr[1][1]=5;  arr[1][2]=6;  arr[2][0]=7;  arr[2][1]=8;  arr[2][2]=9; |

Ví dụ khai báo và khởi tạo mảng đa chiều trong java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | public class TestArray3 {      public static void main(String args[]) {            // khai báo và khởi tạo mảng 2 chiều          int arr[][] = { { 1, 2, 3 }, { 2, 4, 5 }, { 4, 4, 5 } };            // in mảng 2 chiều r màn hình          for (int i = 0; i < 3; i++) {              for (int j = 0; j < 3; j++) {                  System.out.print(arr[i][j] + " ");              }              System.out.println();          }        }  } |

Kết quả:

1 2 3

2 4 5

4 4 5

Ví dụ cộng 2 ma trận (mảng đa chiều) trong Java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | package vn.viettuts.array;    public class CongMaTran {      public static void main(String[] args) {          // tao hai ma tran          int a[][] = { { 1, 3, 4 }, { 3, 4, 5 } };          int b[][] = { { 1, 3, 4 }, { 3, 4, 5 } };            // tao ma tran khac de luu giu ket qua phep cong hai ma tran          int c[][] = new int[2][3];            // cong va in tong hai ma tran          for (int i = 0; i < 2; i++) {              for (int j = 0; j < 3; j++) {                  c[i][j] = a[i][j] + b[i][j];                  System.out.print(c[i][j] + " ");              }              System.out.println();          }      }  } |

Kết quả:

2 6 8

6 8 10

**Lợi thế và hạn chế của mảng trong java**

**Lợi thế**

* **Tối ưu code**: Mảng giúp lấy hoặc sắp xếp dữ liệu dễ dàng.
* **Truy cập ngẫu nhiên**: Chúng ta có thể lấy dữ liệu được lưu trữ ở bất kỳ vị trí nào thông qua chỉ số.

**Hạn chế**

|  |  |
| --- | --- |
| **STT** | **Phương thức và Miêu tả** |
| 1 | **public static int binarySearch(Object[] a, Object key)**  Tìm kiếm mảng của Object (byte, int, double, …) đã cho với giá trị đã xác định bởi sử dụng thuật toán tìm kiếm nhị phân. Mảng này phải được xếp thứ tự trước khi gọi phương thức này. Nó trả về chỉ mục của từ khóa tìm kiếm, nếu nó nằm trong danh sách, nếu không thì, bằng (-(điểm chèn + 1)). |
| 2 | **public static boolean equals(long[] a, long[] a2)**  Trả về true nếu hai mảng long đã cho là cân bằng nhau. Hai mảng này được cho là cân bằng nếu cả hai mảng chứa cùng số lượng phần tử, và tất cả các cặp phần tử tương ứng của hai mảng là cân bằng. Phương thức tương tự có thể được sử dụng bởi tất cả kiểu dữ liệu gốc khác (byte, short, int, …). |
| 3 | **public static void fill(int[] a, int val)**  Gán giá trị int đã cho tới mỗi phần tử của mảng int đã cho. Phương thức tương tự có thể được sử dụng bởi tất cả kiểu dữ liệu gốc khác (byte, short, int, …). |
| 4 | **public static void sort(Object[] a)**  Xếp thứ tự mảng các đối tượng đã cho theo thứ tự tăng dần, theo thứ tự tự nhiên của các phần tử. Phương thức tương tự có thể được sử dụng bởi tất cả kiểu dữ liệu gốc khác (byte, short, int, …). |

* **Giới hạn kích thước**: Chúng ta phải khởi tạo kích thước của mảng trong java trước khi sử dụng. Không thể thay đổi được kích thước của nó lúc runtime. Collection framework được sử dụng trong java để giải quyết vấn đề này.

**Giới thiệu Lớp Array trong Java**

Lớp *java.util.Arrays* chứa nhiều phương thức static đa dạng để xếp thứ tự và tìm kiếm các mảng, so sánh các mảng và điền các phần tử vào mảng.