Mảng (Array) trong Java

Thường thì, mảng là một tập hợp các phần tử có kiểu tương tự nhau mà có vị trí ô nhớ liền kề. Mảng trong Java là một đối tượng chứa các phần tử có kiểu dữ liệu giống nhau. Nó là một cấu trúc dữ liệu, tại đó chúng ta có thể lưu trữ các phần tử tương tự nhau. Chúng ta chỉ có thể lưu trữ một tập hợp cố cố định các phần tử trong một mảng trong Java.

Mảng trong Java là dựa trên chỉ mục (index), phần tử đầu tiên của mảng được lưu trữ tại chỉ mục 0.

Chương hướng dẫn này giới thiệu cách khai báo các biến mảng, tạo các mảng, xử lý các mảng bởi sử dụng chỉ mục của các biến, mảng một chiều và mảng đa chiều trong Java.

Để hiểu sâu hơn các khái niệm được trình bày trong chương này, mời bạn tham khảo loạt bài: **Ví dụ về Array trong Java**.

Lợi thế của mảng trong Java

- **Tối ưu hóa code**: từ đó chúng ta có thể thu nhận và sắp xếp dữ liệu một cách dễ dàng.
- **Truy cập ngẫu nhiên**: chúng ta có thể lấy bất cứ dữ liệu nào ở tại bất cứ vị trí chỉ mục nào.

Hạn chế của mảng trong Java

- **Giới hạn kích cỡ**: Chúng ta chỉ có thể lưu trữ kích cỡ cố định số phần tử trong mảng. Nó không tăng kích cỡ của nó tại runtime. Để xử lý vấn đề này, Collection Framework được sử dụng trong Java.

Các kiểu mảng trong Java

Có hai kiểu mảng trong Java, đó là:

- Mảng một chiều
- Mảng đa chiều

Khai báo biến mảng trong Java

Để sử dụng một mảng trong một chương trình, bạn phải khai báo một biến để tham chiếu mảng, và bạn phải xác định kiểu mảng mà biến có thể tham chiếu. Dưới đây là cú pháp để khai báo một biến mảng:

```
Kieu_du_lieu[] Bien_tham_chieu_mang;   // cach uu tien.

hoac

Kieu_du_lieu Bien_tham_chieu_mang[];  //  lam viec nhung khong la cach uu tien.
```

Ghi chú: Kieu_du_lieu[] Bien_tham_chieu_mang được ưa thích hơn. Còn **Kieu_du_lieu Bien_tham_chieu_mang[]** xuất phát từ ngôn ngữ C/C++ và được chấp nhận trong Java.

Ví dụ:

Đoạn code sau là ví dụ minh họa cho cú pháp này:

```
double[] BK49;         // cach uu tien.

hoac

double BK49[];         //  lam viec nhung khong la cach uu tien.
```

Tạo mảng trong Java

Bạn có thể tạo một mảng bởi sử dụng toán tử new với cú pháp sau:

```
Bien_tham_chieu_mang = new Kieu_du_lieu[Kich_co_mang];
```

Lệnh trên thực hiện hai công việc sau:

- Nó tạo một mảng bởi sử dụng new Kieu_du_lieu[Kich_co_mang];
- Nó gán tham chiếu của mảng mới được tạo tới biến Bien_tham_chieu_mang

Khai báo một biến mảng, tạo một mảng, và gán tham chiếu của mảng tới biến có thể được tổ hợp trong một lệnh, như sau:

```
Kieu_du_lieu[] Bien_tham_chieu_mang = new Kieu_du_lieu[Kich_co_mang];
```

Bạn cũng có thể tạo các mảng bởi sử dụng cách sau:

```
Kieu_du_lieu[] Bien_tham_chieu_mang = {giatri0, giatri1, ..., giatriN};
```

Các phần tử mảng được truy cập thông qua **index – chỉ mục**. Chỉ mục của mảng được tính toán từ 0 tới **Bien_tham_chieu_mang.length-1**.

Ví dụ:

Lệnh sau khai báo một biến mảng, BK49, tạo một mảng gồm 10 phần tử với kiểu double và gán tham chiếu tới BK49.

```
double[] BK49 = new double[10];
```

Mảng một chiều trong Java

Bạn theo dõi ví dụ đơn giản sau về mảng một chiều. Ở đây, chúng ta khai báo, thuyết minh, khởi tạo và vọc mảng.

Khi xử lý (chế biến) các phần tử mảng, chúng ta thường sử dụng hoặc vòng lặp for hoặc vòng lặp foreach bởi vì tất cả phần tử trong một mảng là cùng kiểu và kích cỡ mảng đã biết.

```
class Array1{
public static void main(String args[]){


int a[]=new int[5];//phan khai bao va thuyet minh
a[0]=10;//Phan khoi tao
a[1]=20;
a[2]=70;
a[3]=40;
a[4]=50;


//in mang
for(int i=0;i<a.length;i++)//length la thuoc tinh cua mang
System.out.println(a[i]);


}}
```

Chúng ta có thể **khai báo, thuyết minh và khởi tạo** mảng trong Java bởi:

```
int a[]={33,3,4,5};//khai bao, khoi tao va thuyet minh
```

Bạn theo dõi ví dụ sau để in mảng này.

```
class Testarray1{
public static void main(String args[]){

int a[]={33,3,4,5};//khai bao, khoi tao va thuyet minh

//in mang
for(int i=0;i<a.length;i++)//length la thuoc tinh cua mang
System.out.println(a[i]);

}}
```

Truyền mảng tới phương thức trong Java

Bạn có thể truyền mảng tới phương thức để mà bạn có thể tái sử dụng cùng tính logic của phương thức đó trên bất cứ mảng nào. Dưới đây là ví dụ đơn giản để lấy số nhỏ nhất của một mảng bởi sử dụng phương thức.

```
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);//Truyen mang toi phuong thuc

}}
```

Vòng lặp foreach trong Java

JDK 1.5 giới thiệu một vòng lặp for mới, được biết với tên gọi **foreach hoặc enhanced for**, mà cho bạn khả năng "vọc" mảng một cách liên tục mà không cần sử dụng một biến chỉ mục.

Ví dụ:

Code sau hiển thị tất cả phần tử trong mảng BK49:

```
public class TestArray3 {


   public static void main(String[] args) {
      double[] BK49 = {1.9, 2.9, 3.4, 3.5};


      // In tat ca cac phan tu mang
      for (double element: BK49) {
         System.out.println(element);
      }
   }
}
```

Trả về một mảng từ một phương thức trong Java

Một phương thức cũng có thể trả về một mảng. Ví dụ, phương thức dưới đây trả về một mảng đảo ngược của mảng khác.

```
public static int[] reverse(int[] list) {
  int[] result = new int[list.length];


  for (int i = 0, j = result.length - 1; i < list.length; i++, j--) {
    result[j] = list[i];
  }
  return result;
}
```

Mảng đa chiều trong Java

Trong trường hợp này, dữ liệu được lưu trữ trong hàng và cột dựa trên chỉ mục. Cú pháp để khai báo mảng đa chiều trong Java:

```
Kieu_du_lieu[][] Bien_tham_chieu_mang; (hoac)
Kieu_du_lieu [][]Bien_tham_chieu_mang; (hoac)
Kieu_du_lieu Bien_tham_chieu_mang[][]; (hoac)
Kieu_du_lieu []Bien_tham_chieu_mang[];
```

Bạn có thể thuyết minh mảng đa chiều trong Java, giống như sau:

```
int[][] arr=new int[3][3];//3 hang va 3 cot
```

Ví dụ về khởi tạo mảng đa chiều trong Java

```
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 đơn giản sau sẽ khai báo, thuyết minh, khởi tạo và in một mảng hai chiều.

```
class Testarray3{
public static void main(String args[]){


//khai bao va khoi tao mang 2 chieu
int arr[][]={{1,2,3},{2,4,5},{4,4,5}};


//in mang hai chieu
for(int i=0;i<3;i++){
 for(int j=0;j<3;j++){
```

```
   System.out.print(arr[i][j]+" ");
 }
 System.out.println();
}


}}
```

Tên lớp của mảng trong Java là gì?

Trong Java, mảng là một đối tượng. Với đối tượng Array, một lớp ủy nhiệm được tạo có tên có thể thu được bởi phương thức **getClass(), getName()** trên đối tượng đó.

```
class Testarray4{
public static void main(String args[]){


int arr[]={4,4,5};


Class c=arr.getClass();
String name=c.getName();


System.out.println(name);


}}
```

Sao chép một mảng trong Java

Bạn có thể sao chép một mảng này sang mảng khác bởi phương thức arraycopy của lớp System. Cú pháp của phương thức arraycopy như sau:

```
public static void arraycopy(
Object src, int srcPos,Object dest, int destPos, int length
)
```

Bạn theo dõi ví dụ của phương thức arraycopy trong Java để hiểu rõ hơn về cú pháp trên:

```
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));
    }
}
```

Cộng hai ma trận trong Java

Ví dụ đơn giản sau sẽ thực hiện phép cộng hai ma trận trong Java:

```
class Testarray5{
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();//new line
}

}}
```

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.

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, …).