Array & ArrayList

Nội dung

- Giới thiệu Array & ArrayList
- Khái niệm về Array
- Cú pháp, đặc điểm và cách sử dụng
- Khái niệm về ArrayList
- Cú pháp, đặc điểm và cách sử dụng
- So sánh và kết luận

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Giới thiệu

- Trong chương này, chúng ta sẽ được học về 2 loại cấu trúc dữ liệu kiểu tập hợp đó là Array và ArrayList
- Các cấu trúc dữ liệu này sẽ giúp ích cho chúng ta rất nhiều khi cần lưu trữ và quản lý một tập hợp các dữ liệu hay các đối tượng khác nhau trong chương trình

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

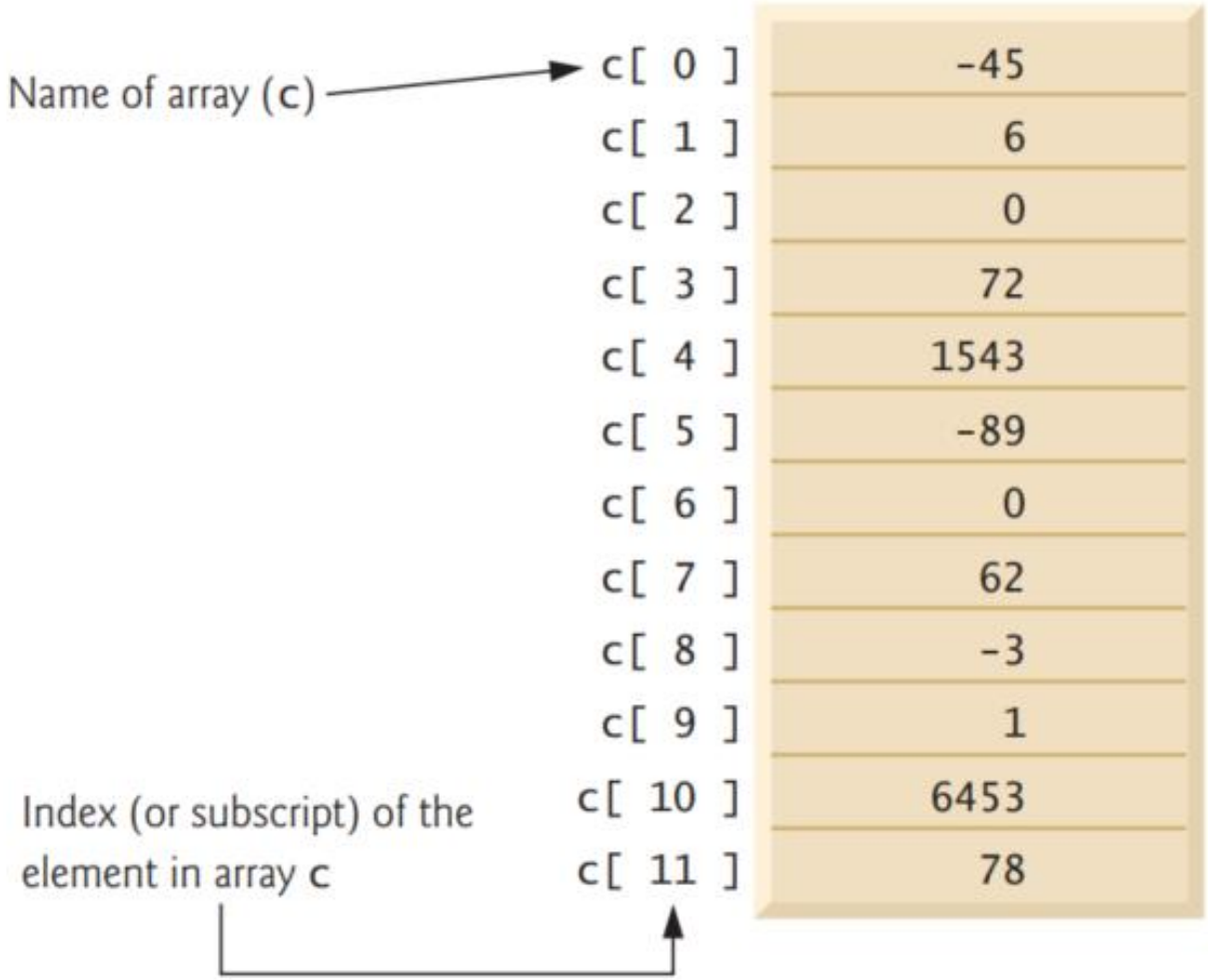
Array

- Array – Mảng là một cấu trúc dữ liệu đơn giản trong ngôn ngữ lập trình
- Mảng được dùng để lưu trữ các đối tượng dữ liệu cùng kiểu (data type), cùng loại và cùng mục đích sử dụng
- Mảng có đặc điểm là khi được khai báo thì nó sẽ được cấp phát một vùng các ô nhớ một cách tuần tự và liên tục trong bộ nhớ của máy tính (Internal Memory - RAM)

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array

- Ví dụ: khi khai báo 1 mảng c có 12 phần tử thì hệ điều hành sẽ cung cấp cho chúng ta một vùng có 12 ô nhớ liên tục trong bộ nhớ máy tính để lưu trữ giá trị của các phần tử của mảng



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array

- Đặc điểm đầu tiên cần ghi nhớ là: mảng lưu trữ một tập các giá trị/đối tượng cùng kiểu
  - Cùng kiểu "nguyên thủy" (Primitive type). Ví dụ: int, double, boolean, …
  - Cùng kiểu "tham chiếu" (Reference type). Ví dụ: SinhVien, HangHoa, …

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array

- Trong ngôn ngữ lập trình Java, mảng là một phần quan trọng trong lưu trữ dữ liệu (*trên bộ nhớ*) và trong quá trình xử lý thuật toán

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array

- Một số đặc điểm khác cần lưu ý khi làm việc với mảng trong Java
  - Mảng là **objects** – kiểu dữ liệu đối tượng/tham chiếu ➔ không phải kiểu dữ liệu "*nguyên thủy*" (Primitive type) giống như **int** hay **double**. Do đó, khi cần cấp phát một mảng trong chương trình, ta sử dụng từ khóa **new** và cú pháp giống như cú pháp để tạo ra một đối tượng mới

    Ví dụ: `int[] a = new int[12]; // cấp phát 1 mảng a có 12 phần tử`
  - Tên của mảng được xem là một biến tham chiếu chứa địa chỉ của ô nhớ đầu tiên trong vùng ô nhớ được cấp phát cho mảng

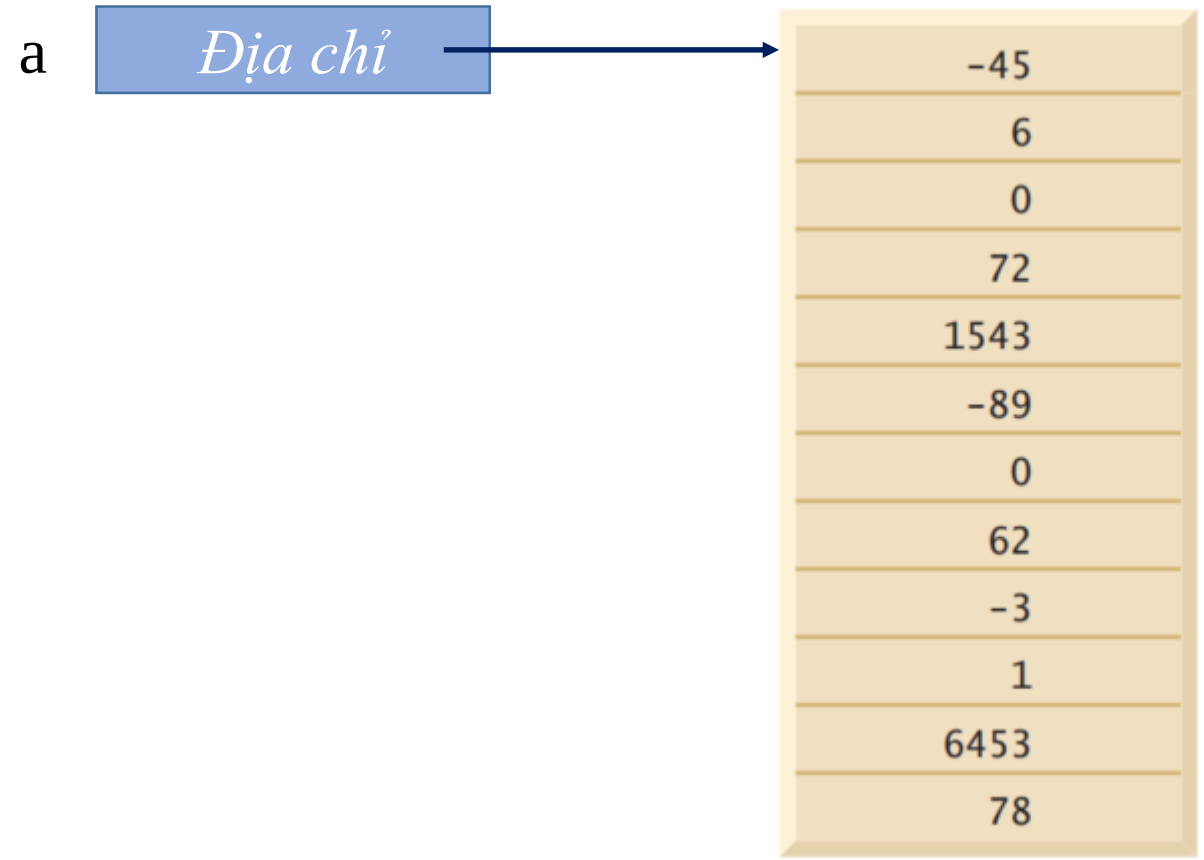
ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array

- Ví dụ: khi ta khai báo

```
int[] a = new int[12];
```

thì a sẽ là một biến chứa địa chỉ của ô nhớ đầu tiên trong vùng ô nhớ được quản lý bởi mảng a



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

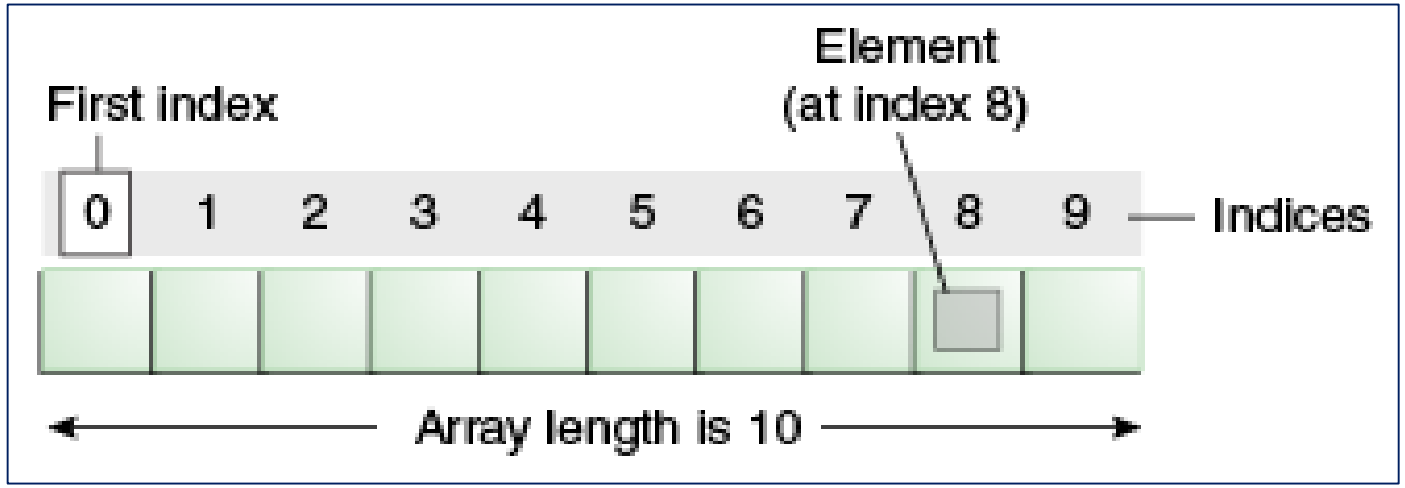
Array Element

- Phần tử của mảng là một đơn vị lưu trữ của mảng hay còn gọi là đơn vị lưu trữ của tập hợp
- Mỗi phần tử của mảng có thể có kiểu dữ liệu là kiểu nguyên thủy (Primitive type) hoặc kiểu tham chiếu (Reference type)

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array Element

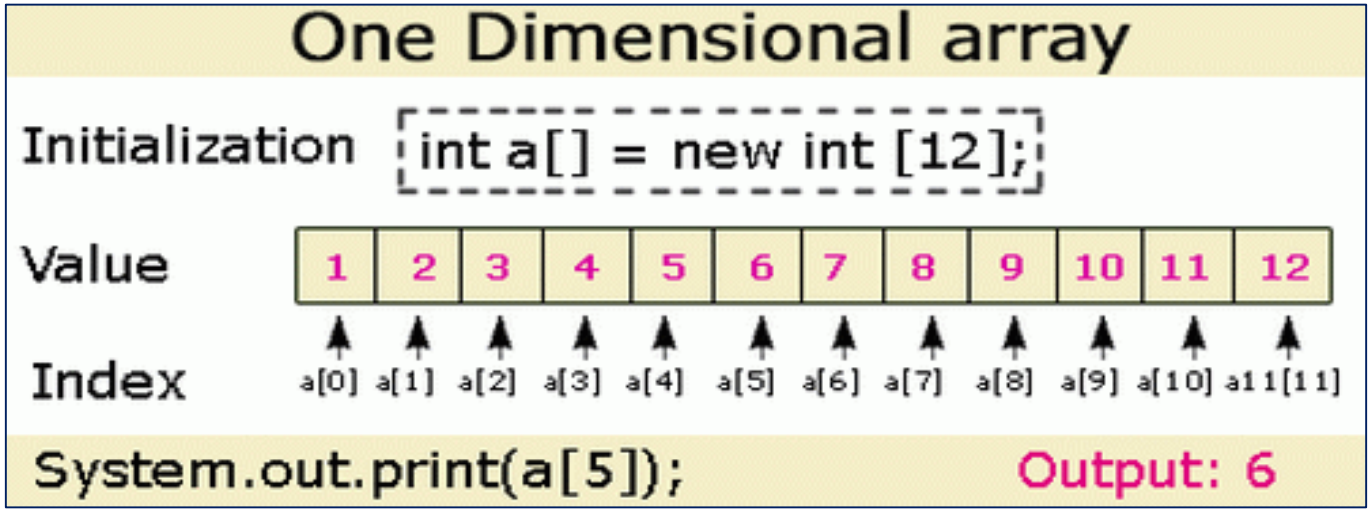
- Phần tử đầu tiên của mảng có chỉ số là 0 (*không phải 1*), các phần tử tiếp theo sẽ có chỉ số tăng dần
- Nếu mảng có *n* phần tử thì phần tử cuối cùng của mảng sẽ có chỉ số là *n* – 1
- Ví dụ: mảng có độ dài là 10 thì phần tử cuối của mảng có chỉ số là 9



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array Element

- Để truy cập đến một phần tử của mảng, ta gọi tên của biến tham chiếu đến mảng, theo sau đó là chỉ số (index hay subscript) của phần tử trong mảng được đặt trong một cặp dấu ngoặc vuông ([ ])
- Ví dụ: int[] a = new int[12];
  - Truy cập đến phần tử thứ 3 và 11 của mảng: a[2], a[10]

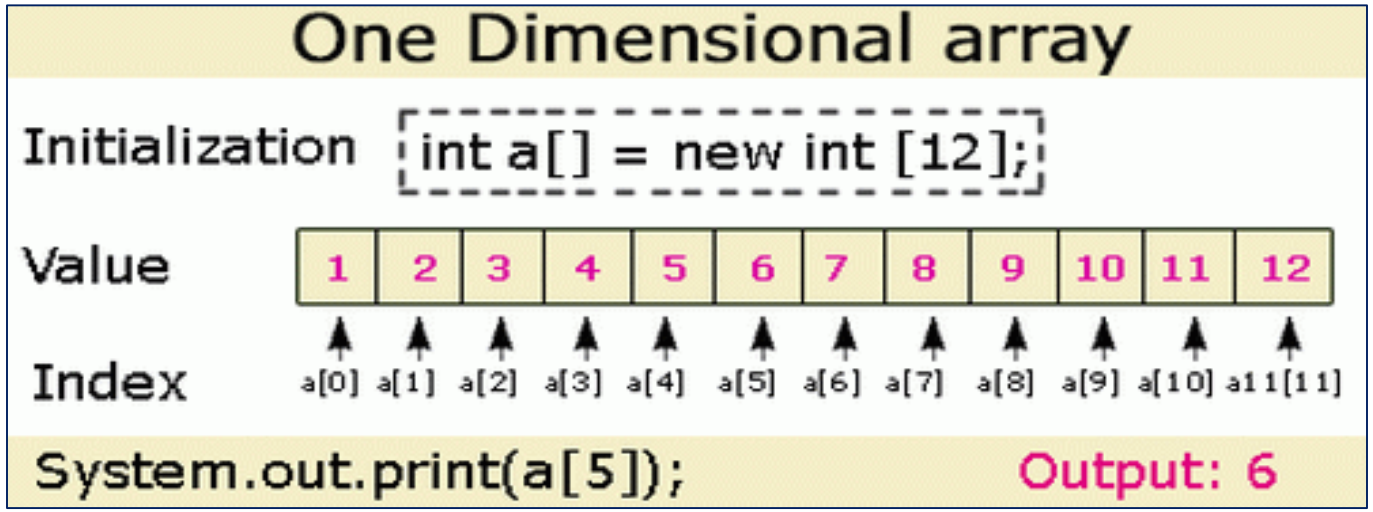


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array Element

- Ta có thể xác định chiều dài hay số phần tử của mảng thông qua thuộc tính length
- Ví dụ: **int**[] a= new **int**[12];

```
int chieuDaiMangA = a.length; // chieuDaiMangA = 12
```



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Đặc điểm

- Chiều dài hay kích thước của 1 mảng không thể bị thay đổi sau khi đã khai báo
- Ví dụ: mảng a đã khai báo có 12 phần tử, thì không thể thay đổi kích thước, cho đến khi bị loại khỏi bộ nhớ
- Kiểu dữ liệu của các phần tử trong mảng cũng không thể bị thay đổi sau khi đã khai báo
- Ví dụ: khi khai báo **int**[] a= new **int**[12]; thì mảng a và các phần tử trong mảng a sẽ luôn có kiểu dữ liệu là **int**, không thể bị thay đổi, cho đến khi bị loại khỏi bộ nhớ

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Đặc điểm

- Có thể khai báo đồng thời khởi gán giá trị cho các phần tử của mảng theo cú pháp sau

```
int[] intArray= {5, 77, 4, 9, 28, 0, -9};
// Trong trường hợp này, không cần sử dụng từ khóa new
```

- Trong trường hợp cấp phát mảng mà không chỉ ra giá trị khởi tạo thì các phần tử của mảng mặc nhiên sẽ được gán giá trị bằng 0

- Ví dụ: `int[] a= new int[12];`

```
// Cấp phát 1 mảng a có 12 phần tử, giá trị của mỗi phần tử = 0
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

for - Duyệt qua phần tử của mảng

- Ta có thể duyệt qua các phần tử của mảng bằng cách sử dụng một vòng lặp (for, while, do…while)

- Ví dụ: sử dụng vòng lặp for

```
for (int i = 0; i < array.length; i++) {
        sum = sum + a[i];
}
```

- Ví dụ: sử dụng vòng lặp while…do

```
int i = 0;
while (i < array.length) {
        // Code xử lý
        i++;
}
```

- Ví dụ: sử dụng vòng lặp do…while

```
int i = 0;
do {
        // Code xử lý
        i++;
} while (i < array.length)
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

for - Duyệt qua phần tử của mảng

- Tuy nhiên khi duyệt qua các phần tử của mảng bằng chỉ số thì ta có thể truy cập đến một phần tử nằm ngoài mảng
- Ví dụ: **int**[] array= {5, 77, 4, 9, 28, 0, -9};

```
for (int i = 0; i < 7; i++) {
        sum = sum + a[7];
}
```

- Vì vậy ta có thể dùng cú pháp duyệt sau đây để tránh liệt kê qua chỉ số các phần tử của mảng

```
for (iterator: arrayName) {
        statement
}
```

- Ví dụ:

```
for (int number: array) {
        sum = sum + number;
}
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Truyền mảng cho phương thức

- Để truyền tham số kiểu mảng (array argument) cho một phương thức, thì ta chỉ cần truyền tên mảng mà không kèm theo cặp dấu ngoặc vuông ([ ]) cho lời gọi hàm/phương thức
- Ví dụ:

```
double[] hourlyTemperatures = new double[ 24 ];
```

Ta có thể truyền tham số hourlyTemperatures cho hàm/phương thức bằng lời gọi:

```
modifyArray(hourlyTemperatures);
```

- Cách truyền như vậy còn được gọi là truyền tham chiếu **passing-by-reference** (truyền tham chiếu của mảng cho phương thức)

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Truyền phần tử của mảng cho phương thức

- Để truyền phần tử của mảng (array element) cho một phương thức, thì ta cần chỉ đích danh ra tên mảng + chỉ số của phần tử cần truyền đặt trong cặp dấu ngoặc vuông ([ ]) cho lời gọi hàm/phương thức
- Ví dụ:

```
double[] hourlyTemperatures = new double[ 24 ];

modifyElement(hourlyTemperatures[5]);
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Truyền phần tử của mảng cho phương thức

- Với cách truyền này thì trình biên dịch sẽ tạo ra một bản copy giá trị phần tử của mảng và truyền cho phương thức. Mọi sự thay đổi đối với phiên bản copy này sẽ không làm thay đổi hay ảnh hưởng đến giá trị gốc của phần tử của mảng
- Cách truyền tham số này còn được gọi là truyền tham trị **passing-by-value**

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Truyền phần tử của mảng cho phương thức

- Với cách truyền này thì trình biên dịch sẽ tạo ra một bản copy giá trị phần tử của mảng và truyền cho phương thức. Mọi sự thay đổi đối với phiên bản copy này sẽ không làm thay đổi hay ảnh hưởng đến giá trị gốc của phần tử của mảng
- Cách truyền tham số này còn được gọi là truyền tham trị **passing-by-value**

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Truyền phần tử của mảng cho phương thức

- Ví dụ:

```
public class PassArray {
    public static void main(String[] args) {
    int[] array = { 1, 2, 3, 4, 5 };
    modifyElement(array[3]);
    System.out.println("Mang sau khi cap nhat gia tri [3]: ");
    for ( int value : array )
        System.out.printf( " %d", value );
}

public static void modifyElement(int element) {
    element = element * 2;
}
```

- Kết quả:

Mang sau khi cap nhat gia tri [3]:
1 2 3 4 5

Giá trị của array[3] vẫn được giữ nguyên

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Sao chép mảng

- Để sao chép 1 mảng, ta sử dụng phương thức ***arraycopy***() của System class, theo cú pháp như sau:

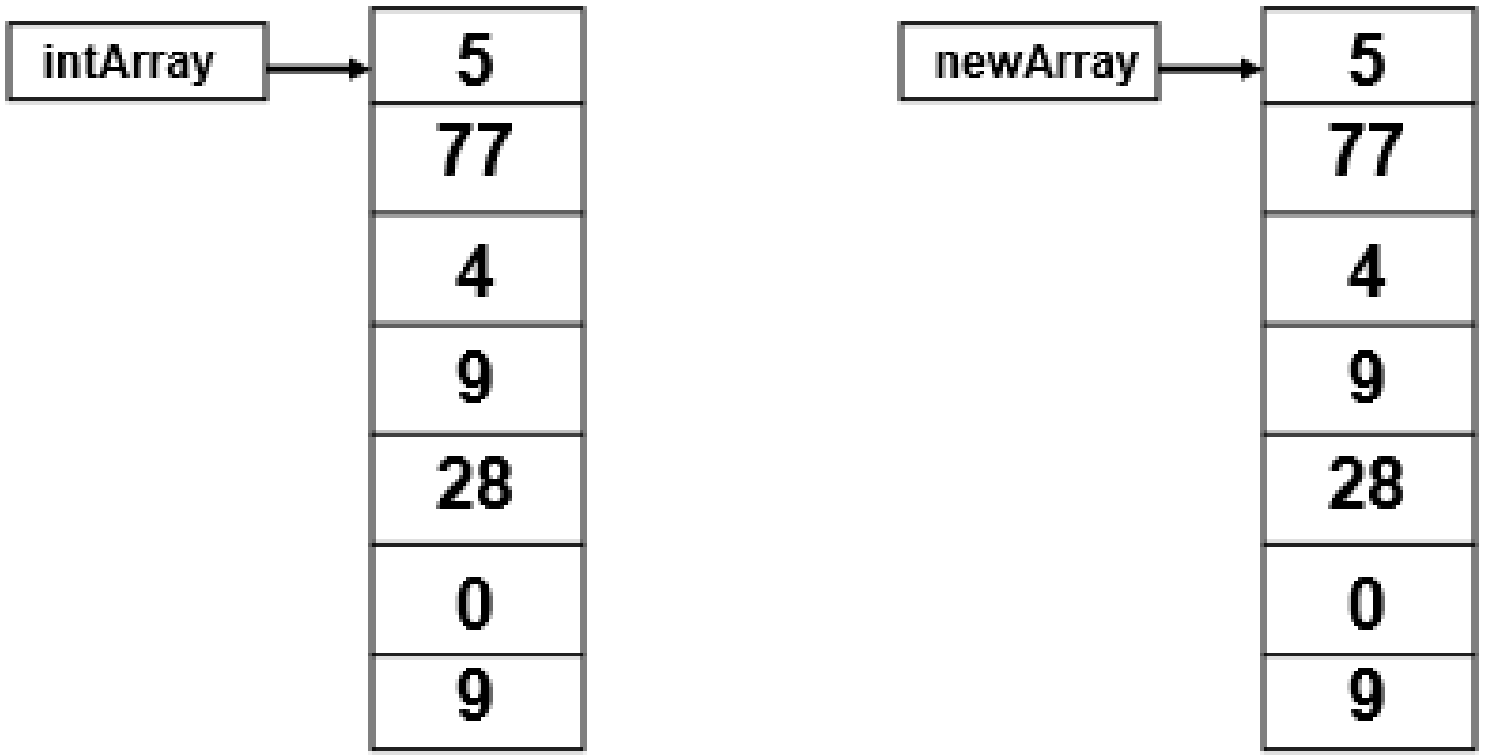
```
public static void arraycopy(Object src, int srcPos,
                             Object dest, int destPos,
                             int length)
```

  - **src** – Mảng được xem là ***nguồn*** cho quá trình sao chép
  - **srcPos** – Vị trí của phần tử bắt đầu sao chép (Index)
  - **dest** – Mảng được xem là ***đích*** của quá trình sao chép (*Mảng sẽ được tạo thành*)
  - **destPos** – Vị trí của phần tử bắt đầu nhận dữ liệu tại "*mảng đích*" (Index)
  - **length** – Số lượng phần tử sẽ sao chép

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

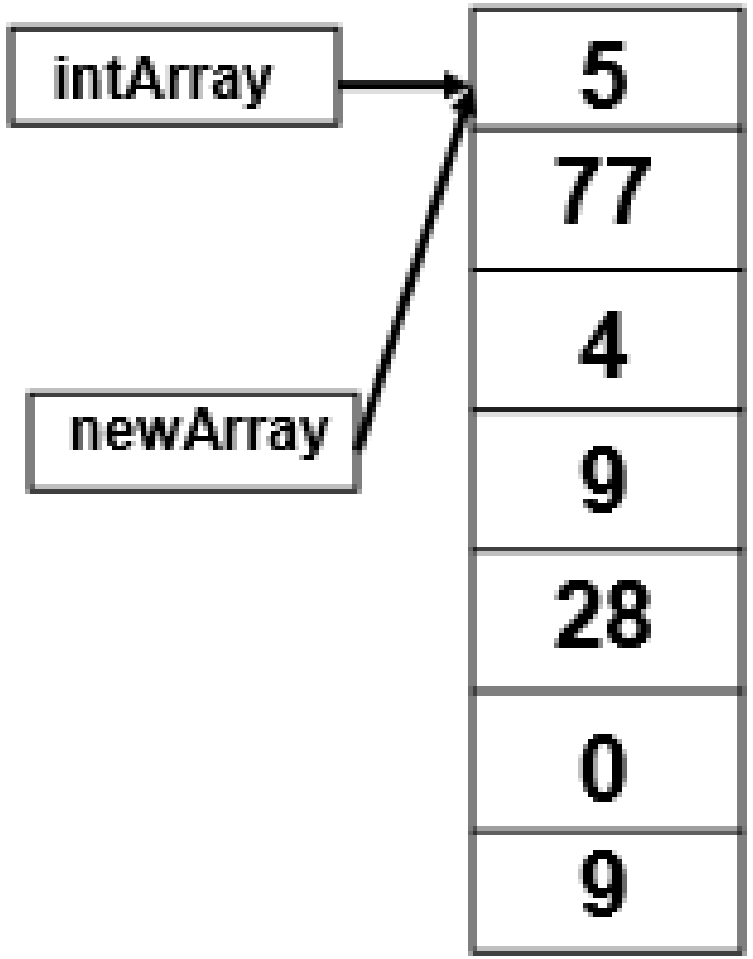
Sao chép nội dung == arraycopy

```
int[ ] intArray= {5, 77, 4, 9, 28, 0, 9};
int[ ] newArray = new int[intArray.length];
System.arraycopy(intArray,0,newArray,0,intArray.length)
```



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

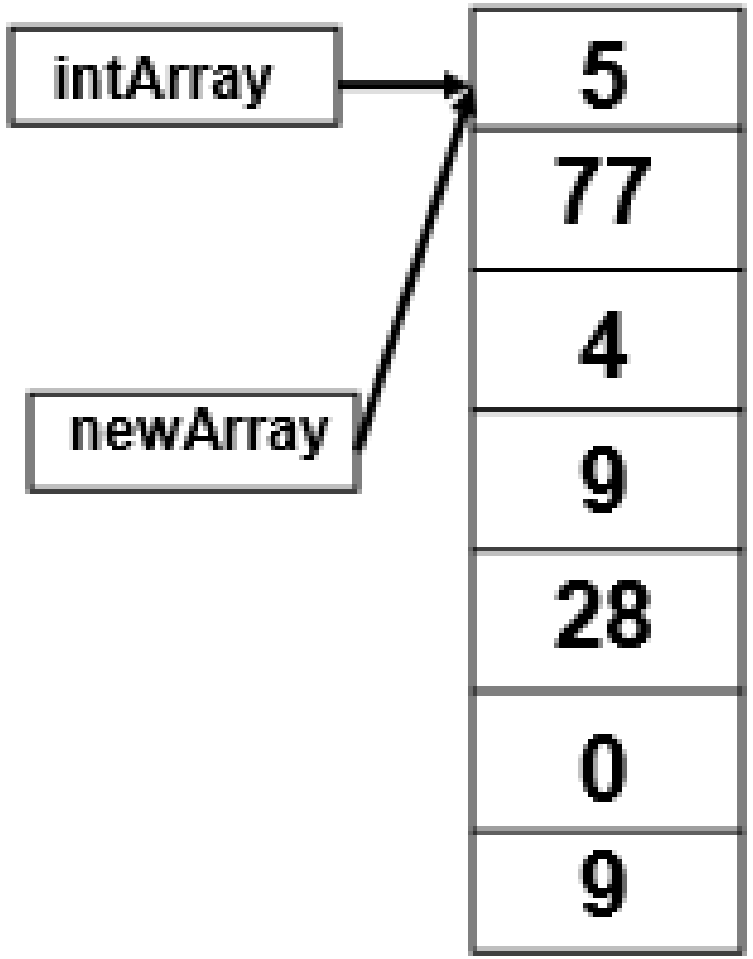
Sao chép tham chiếu




```
int[ ] intArray= {5, 77, 4, 9, 28, 0, 9};
int[ ] newArray = intArray;
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Sao chép tham chiếu




```
int[ ] intArray= {5, 77, 4, 9, 28, 0, 9};
int[ ] newArray = intArray;
```

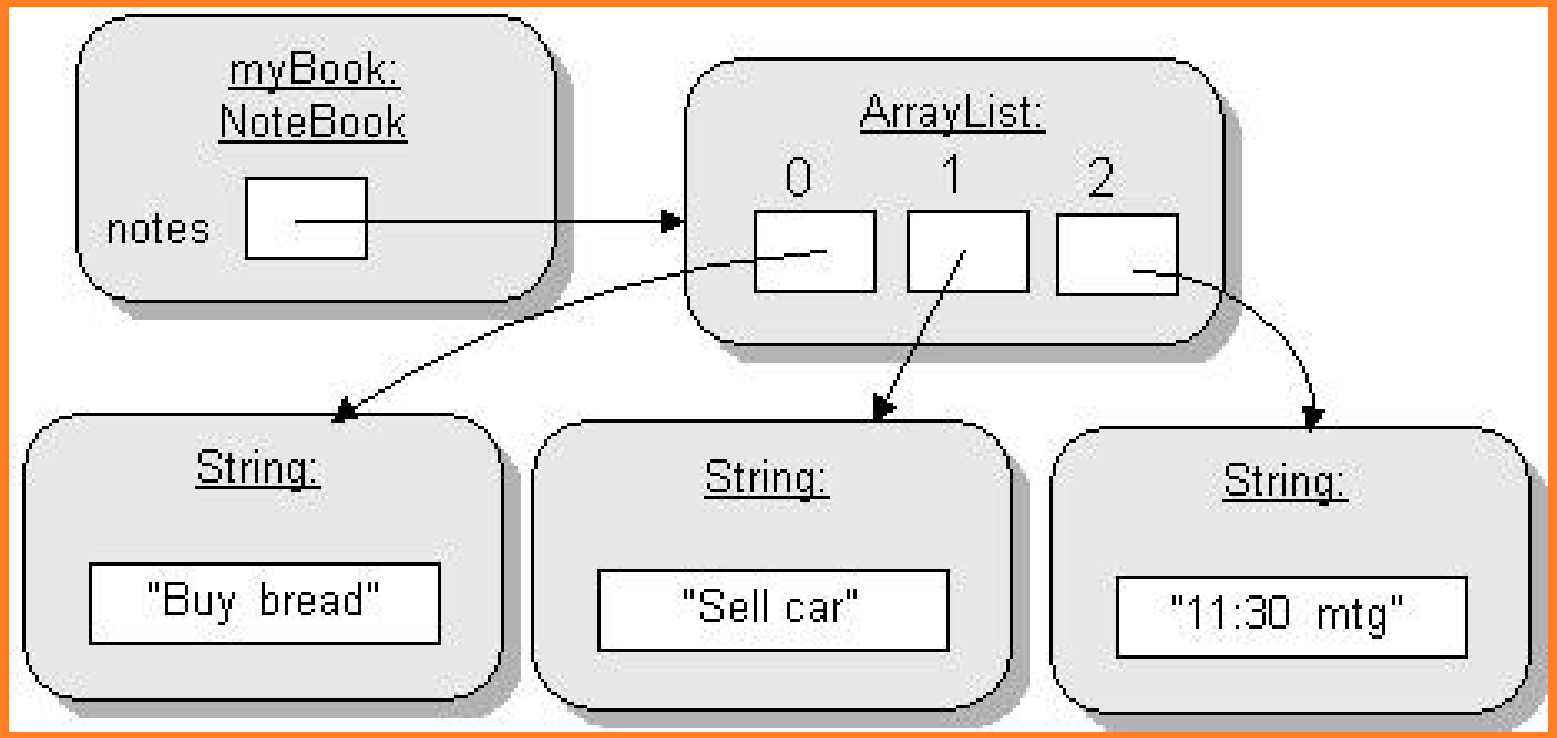

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

ArrayList

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

ArrayList

ArrayList example in java



ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

ArrayList

- **ArrayList** là một lớp được định nghĩa ở dạng collection, thuộc package java.util
- **ArrayList** mang nhiều đặc điểm gần giống như Array, nhưng có thêm các phương thức hỗ trợ tốt hơn so với Array trước đây
- Một điểm mới của **ArrayList** so với Array là **ArrayList** có thể thay đổi kích thước tùy thuộc vào "*nhu cầu sử dụng*" khác hoàn toàn với "*kích thước cố định*" đối với Array
  - Kích thước có thể tăng khi có nhu cầu
  - Để xác định kích thước của **ArrayList**, sử dụng phương thức size()

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

ArrayList

- **ArrayList** cho phép lưu trữ các phần tử khác loại
  - Mỗi phần tử có trong **ArrayList** được xem như là 1 Object mà nó tham chiếu đến
  - **ArrayList** không cho phép chứa dữ liệu "*nguyên thủy – primitive data*" như: int, double, ...
  - Nếu cần phải chứa các giá trị nguyên, số thực, … trong **ArrayList**, hãy sử dụng wrapper classes (Integer, Double, Boolean, …)

- Ví dụ: int ⇨ Integer

  double ⇨ Double

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Kiểu generic (Generic type)

- Khi khai báo một **ArrayList**, ta có thể không cần khai báo kiểu dữ liệu của các phần tử thuộc **ArrayList** đó, nhưng khi lấy ra giá trị của phần tử thì ta cần phải ép về kiểu dữ liệu phù hợp mới có thể sử dụng được
- Một cách khai báo khác là ta chỉ định rõ kiểu dữ liệu của **ArrayList** thông qua việc khai báo kiểu dữ liệu đặt trong cặp dấu < >. Cách khai báo như vậy gọi là khai báo kiểu Generic (Generic type)
- Ví dụ:

  Khai báo kiểu Object: **ArrayList** list = new **ArrayList**();

  Khai báo kiểu Generic: **ArrayList<Integer>** list = new **ArrayList<Integer>**();

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Kiểu generic (Generic type)

- Chỉ ra loại đối tượng cụ thể cần lưu trữ trong collection. Điều này mang lại 3 lợi ích rất lớn trong lập trình
  - **Type-safety:** Chỉ cho phép lưu giữ 1 loại đối tượng trong collection tại một thời điểm. Điều này khác hẳn với việc lưu trữ "tùm lum" loại dữ liệu trong cùng một nơi chứa
  - **Type casting is not required:** Không cần phải ép kiểu do mỗi collection khi sử dụng chỉ lưu trữ 1 loại đối tượng
  - **Compile-Time Checking:** Cho phép kiểm tra tại thời điểm biên dịch nhằm giảm lỗi phát sinh trong quá trình chạy chương trình

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Kiểu generic (Generic type)

Before Generics, we need to type cast.

```
ArrayList list = new ArrayList();
list.add("hello");
String s = (String) list.get(0);//typecasting
```

After Generics, we don't need to typecast the object.

```
ArrayList<String> list = new ArrayList<String>();
list.add("hello");
String s = list.get(0);
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

ArrayList methods

.add(Object x)
Thêm mới 1 object x vào ArrayList

.remove(Object x)
Loại bỏ 1 object x khỏi ArrayList

.remove(int index)
Loại bỏ object x tại vị trí Index

.get(int index)
Lấy phần tử x tại vị trí index

.set(int index, Object x)
Cập nhật giá trị cho phần tử tại vị trí index bởi object x

.contains(Object o)
Trả về kết quả True, nếu x có trong ArrayList

.isEmpty()
Trả về True nếu ArrayList hiện đang trống rỗng

.toArray()
Chuyển ArrayList thành 1 mảng Objets

.clone()
Sao chép toàn bộ ArrayList hiện tại vào trong 1 ArrayList mới

.clear()
Xóa toàn bộ nội dung của 1 ArrayList

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

for - Duyệt qua phần tử của ArrayList

- Các phần tử của ArrayList không thể truy cập trực tiếp bằng chỉ số vì vậy để duyệt qua các phần tử của ArrayList, ta sử dụng vòng lặp for với cú pháp như sau (còn được gọi là vòng lặp foreach):

```
for(data_type variable : array | collection)
{
    //-- do some things
    .....
}
```

- **Data_type**: kiểu dữ liệu của variable
- **Array | collection**: Tên của mảng, hay tên của Collection chứa dữ liệu cần duyệt

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Foreach trên Array

```
class ForEachExample1{
    public static void main(String args[]){
        int arr[]={12,13,14,44};
        for(int i:arr){
            System.out.println(i);
        }
    }
}
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Foreach trên ArrayList

```
import java.util.*;
class ForEachExample2{
    public static void main(String args[]){
        ArrayList<String> list=new ArrayList<String>();
        list.add("vimal");
        list.add("sonoo");
        list.add("ratan");
        for(String s:list){
            System.out.println(s);
        }
    }
}
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Ví dụ

- Class SinhVien :: Thêm, xóa, truy xuất phần tử trên ArrayList
- Class SanPham :: Lặp trên danh sách sản phẩm
- Class PhuongTienGiaoThong :: Sinh viên tự làm
- Class NhanVien :: Sinh viên tự làm

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array vs ArrayList

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array vs ArrayList

	Array	ArrayList
Length:	Fixed Length	Variable Length
Data Types:	Primitives, Objects	Objects, Generics
Performance:	Get and Insert in O(1)	Get and Insert in O(1)

Array vs ArrayList:: Điểm giống

- **Cách thức thêm và lấy phần tử:** Tác vụ thêm phần tử và truy xuất phần tử của Array và ArrayList đều có cùng thời gian xử lý O(1)
- **Phần tử trùng lặp:** Array và ArrayList cho phép các phần tử có cùng giá trị với nhau
- **Giá trị null :** Cho phép chứa giá trị null và sử dụng chỉ số (index) để tham chiếu và truy xuất đến phần tử
- **Không có thứ tự:** Array và ArrayList là kiểu lưu trữ không có thứ tự

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array vs ArrayList:: Điểm khác

❑**Cách thức truy xuất phần tử**:

Để truy xuất đến các phần tử trên mảng, ta sử dụng cặp dấu []. Trong ArrayList, ta dùng phương thức .get()

```
// A Java program to demonstrate differences between array
// and ArrayList
import java.util.ArrayList;
import java.util.Arrays;

class Test
{
    public static void main(String args[])
    {
        /* ........... Normal Array.............. */
        int[] arr = new int[3];
        arr[0] = 1;
        arr[1] = 2;
        System.out.println(arr[0]);

        /*............ArrayList..............*/
        // Create an arrayList with initial capacity 2
        ArrayList<Integer> arrL = new ArrayList<Integer>(2);

        // Add elements to ArrayList
        arrL.add(1);
        arrL.add(2);

        // Access elements of ArrayList
        System.out.println(arrL.get(0));
    }
}
```

Array vs ArrayList:: Điểm khác

- Khác biệt về tính linh hoạt đối với kích thước
- Kích thước của mảng là cố định, không thể thay đổi kích thước sau khi khai báo
- Kích thước của ArrayList "co giãn" tùy theo nhu cầu sử dụng. Tự động tăng lên khi thêm mới, tự động giảm bớt khi bị loại bỏ

```
// A Java program to demonstrate differences between array
// and ArrayList
import java.util.ArrayList;
import java.util.Arrays;
class Test
{
    public static void main(String args[])
    {
        /* ........... Normal Array.............. */
        // Need to specify the size for array
        int[] arr = new int[3];
        arr[0] = 1;
        arr[1] = 2;
        arr[2] = 3;
        // We cannot add more elements to array arr[]

        /*............ArrayList.............*/
        // Need not to specify size
        ArrayList<Integer> arrL = new ArrayList<Integer>();
        arrL.add(1);
        arrL.add(2);
        arrL.add(3);
        arrL.add(4);
        // We can add more elements to arrL

        System.out.println(arrL);
        System.out.println(Arrays.toString(arr));
    }
}
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Array vs ArrayList:: Điểm khác

- Khác biệt về đối tượng lưu trữ
- Array có thể chứa dữ liệu nguyên thủy như int, double, float, … hoặc cũng có thể chứa object
- ArrayList chỉ có thể chứa dữ liệu kiểu Wrapper Class và kiểu tham chiếu, không chứa kiểu nguyên thủy

```
import java.util.ArrayList;
class Test
{
    public static void main(String args[])
    {
        // allowed
        int[] array = new int[3];

        // allowed, however, need to be intialized
        Test[] array1 = new Test[3];

        // not allowed (Uncommenting below line causes
        // compiler error)
        // ArrayList<char> arrL = new ArrayList<char>();

        // Allowed
        ArrayList<Integer> arrL1 = new ArrayList<>();
        ArrayList<String> arrL2 = new ArrayList<>();
        ArrayList<object> arrL3 = new ArrayList<>();
    }
}
```


ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Tóm tắt

- ✓**Array** & **ArrayList** dùng để làm gì ? Truy xuất phần tử ra sao ?
- ✓**Array** & **ArrayList** giống và khác nhau thế nào ?
- ✓Các phương thức tiêu biểu của **ArrayList**
- ✓Cú pháp và cách sử dụng foreach trong duyệt dữ liệu **Array**, hay **ArrayList**

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn

Tài liệu tham khảo

- Y. Daniel Lang, "**Introduction to Java Programming Comprehension Version**" 10th Edition.
- Jose M. Garrido, "**Object-Oriented Programming: From Problem Solving to Java**"
- Paul Deitel, Harvey Deitel, "**Java : How to program**", 9th edition, 2012
- Oracle, "**The Java™ Tutorials**", https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html, 6:20PM, 18/01/2018
- Java tutorial, https://howtodoinjava.com/java/basics/, 15:00PM, 20/02/2020

ThS. Trần Đức Hiếu – Khoa Công Nghệ Thông Tin - Trường ĐH Nguyễn Tất Thành – 0769.829.899 – tranduchieu@nttu.edu.vn