

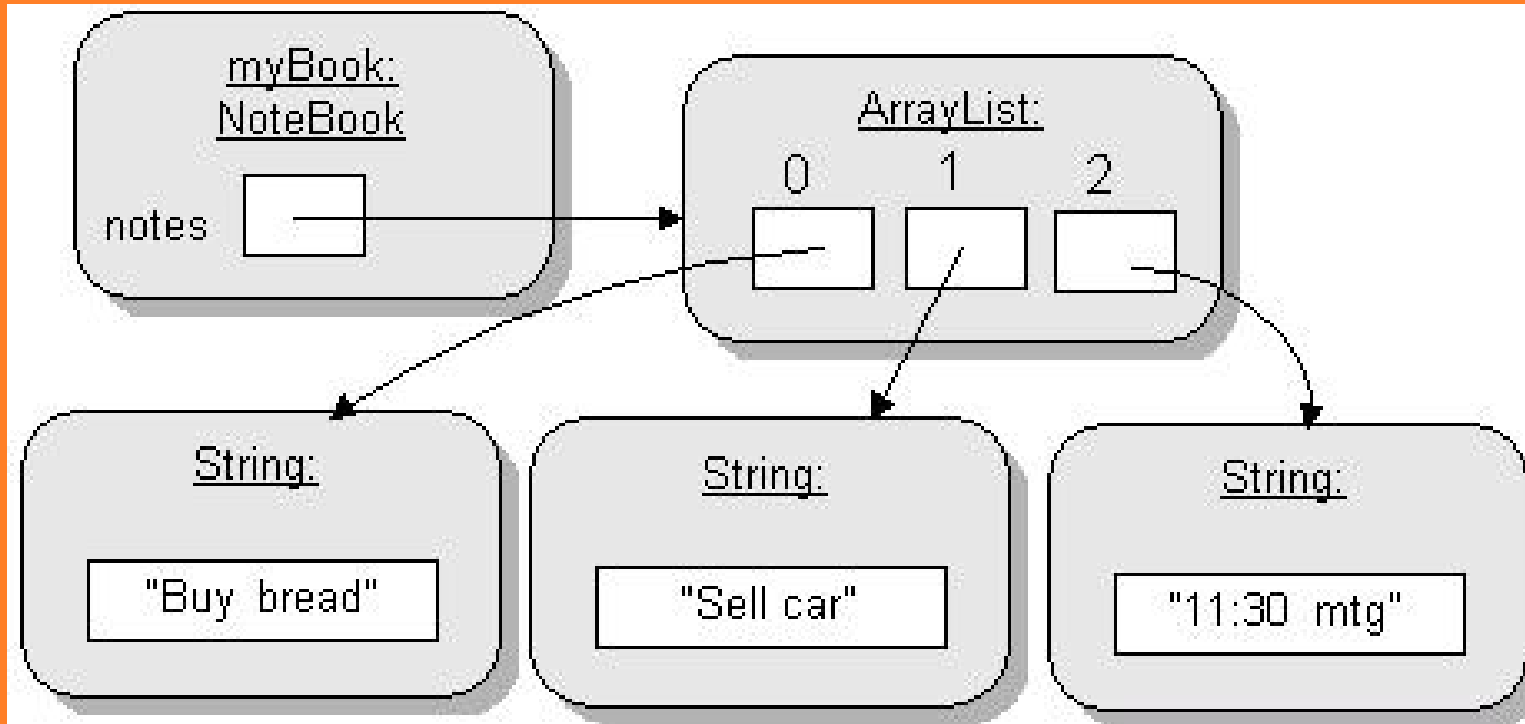
# **Bài 07:**

# **Lớp ArrayList – Mảng 2 chiều**

Giảng Viên: ThS. Giang Hào Côn

# 7.1/ Lớp ArrayList trong Java

## ArrayList example in java



## 7.1/ Lớp ArrayList trong Java

- **ArrayList** là một class 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()**.

# 7.1/ Lớp ArrayList trong Java

Lớp ArrayList hỗ trợ **3 constructor**:

- Constructor đầu tiên xây dựng một danh sách mảng trống.

```
ArrayList( )
```

- Constructor thứ hai xây dựng một Array List mà được khởi tạo với các phần tử của **collection c**.

```
ArrayList(Collection c)
```

- Constructor tiếp theo xây dựng một Array List mà có dung lượng ban đầu được xác định. Dung lượng này là kích cỡ của mảng mà được sử dụng để lưu các phần tử. **Dung lượng tự động tăng khi các phần tử được thêm vào Array List này.**

```
ArrayList(int capacity)
```

# 7.1/ Lớp ArrayList trong Java

**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](#).

## 7.2/ Các phương thức của lớp ArrayList

Phương thức	Mô tả
<code>boolean add(Object o)</code>	Nó được sử dụng để nối thêm phần tử được chỉ định vào cuối một danh sách.
<code>void add(int index, Object element)</code>	Nó được sử dụng để chèn phần tử element tại vị trí index vào danh sách.
<code>boolean addAll(Collection c)</code>	Nó được sử dụng để nối tất cả các phần tử trong collection c vào cuối của danh sách, theo thứ tự chúng được trả về bởi bộ lặp iterator.
<code>boolean addAll(int index, Collection c)</code>	Nó được sử dụng để chèn tất cả các phần tử trong collection c vào danh sách, bắt đầu từ vị trí index.

## 7.2/ Các phương thức của lớp ArrayList

<code>void retainAll(Collection c)</code>	Nó được sử dụng để xóa những phần tử không thuộc collection c ra khỏi danh sách.
<code>void removeAll(Collection c)</code>	Nó được sử dụng để xóa những phần tử thuộc collection c ra khỏi danh sách.
<code>int indexOf(Object o)</code>	Nó được sử dụng để trả về chỉ mục trong danh sách với sự xuất hiện đầu tiên của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa phần tử này.
<code>int lastIndexOf(Object o)</code>	Nó được sử dụng để trả về chỉ mục trong danh sách với sự xuất hiện cuối cùng của phần tử được chỉ định, hoặc -1 nếu danh sách không chứa phần tử này.

## 7.2/ Các phương thức của lớp ArrayList

<code>Object[] toArray()</code>	Nó được sử dụng để trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự.
<code>Object[] toArray(Object[] a)</code>	Nó được sử dụng để trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự.
<code>Object clone()</code>	Nó được sử dụng để trả về một bản sao của ArrayList.
<code>void clear()</code>	Nó được sử dụng để xóa tất cả các phần tử từ danh sách này.
<code>void trimToSize()</code>	Nó được sử dụng để cắt dung lượng của thể hiện ArrayList này là kích thước danh sách hiện tại.
<code>boolean contains(element)</code>	Kết quả trả về là true nếu tìm thấy element trong danh sách, ngược lại trả về false.



## 7.3/ các ví dụ ArrayList trong Java

### Khởi tạo một ArrayList

```
// import gói thư viện java.util.ArrayList
import java.util.ArrayList;

public class KhoiTaoArrayList {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là listString
        // có kiểu là String
        ArrayList<String> listString = new ArrayList<String>();
    }
}
```

## 7.3/ các ví dụ ArrayList trong Java

Khởi tạo một ArrayList – biết trước số lượng phần tử

```
// import gói thư viện java.util.ArrayList
import java.util.ArrayList;

public class KhoiTaoArrayList {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là listString
        // có kiểu là String
        ArrayList<String> listString = new ArrayList<String>(20);
    }
}
```

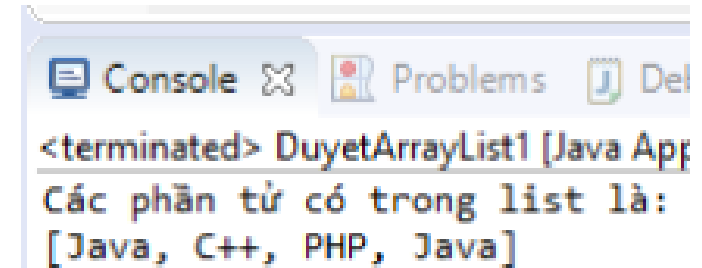
## 7.3/ các ví dụ ArrayList trong Java

### Hiển thị theo tên của ArrayList

```
import java.util.ArrayList;

public class DuyetArrayList1 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
        // hiển thị các phần tử của list
        System.out.println("Các phần tử có trong list là: ");
        System.out.println(list);
    }
}
```

### Kết quả



```
<terminated> DuyetArrayList1 [Java Applet]
Các phần tử có trong list là:
[Java, C++, PHP, Java]
```

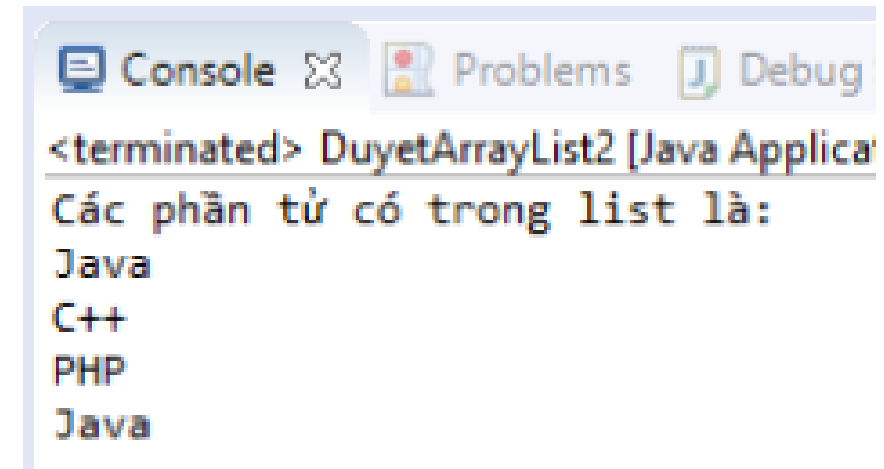
## 7.3/ các ví dụ ArrayList trong Java

### Duyệt các phần tử của ArrayList - sử dụng vòng lặp for

```
import java.util.ArrayList;

public class DuyetArrayList2 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
        // sử dụng vòng lặp for - hiển thị các phần tử của list
        System.out.println("Các phần tử có trong list là: ");
        for (int i = 0; i < list.size(); i++) {
            System.out.println(list.get(i));
        }
    }
}
```

### Kết quả



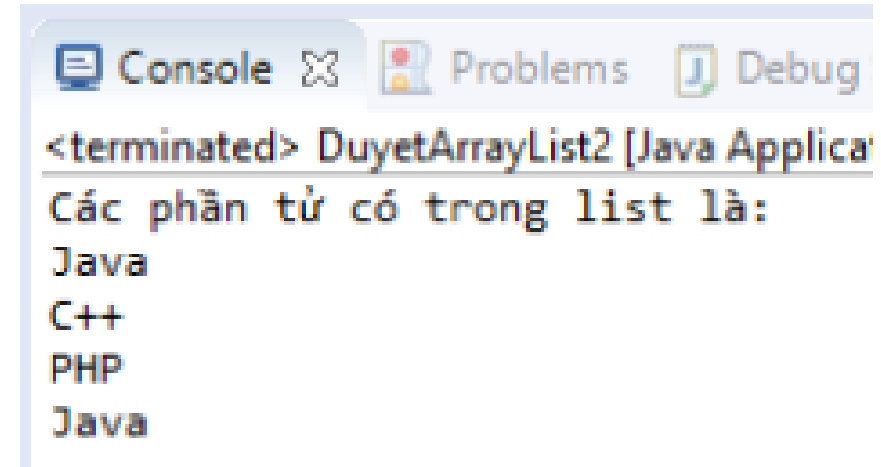
## 7.3/ các ví dụ ArrayList trong Java

### Duyệt các phần tử của ArrayList - sử dụng vòng lặp for cải tiến

```
import java.util.ArrayList;

public class DuyệtArrayList3 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
        // sử dụng vòng lặp for cải tiến - hiển thị các phần tử của list
        System.out.println("Các phần tử có trong list là: ");
        for (String str : list) {
            System.out.println(str);
        }
    }
}
```

### Kết quả



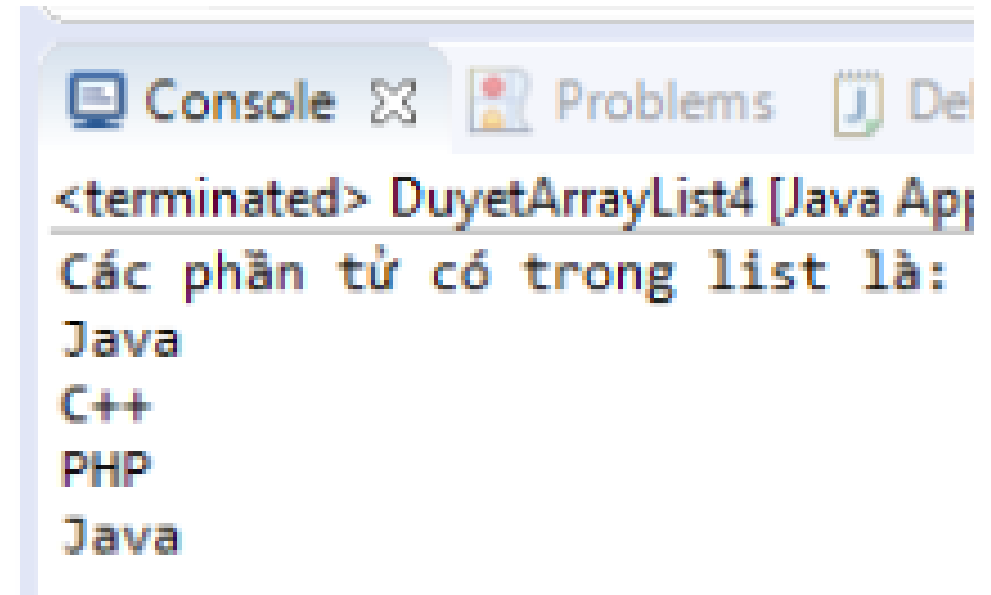
## 7.3/ các ví dụ ArrayList trong Java

### Duyệt các phần tử của ArrayList - sử dụng Iterator.

```
import java.util.ArrayList;
import java.util.Iterator;

public class DuyetArrayList4 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
        // sử dụng Iterator - hiển thị các phần tử của list
        Iterator<String> iterator = list.iterator();
        System.out.println("Các phần tử có trong list là: ");
        while (iterator.hasNext()) {
            System.out.println((String) iterator.next());
        }
    }
}
```

### Kết quả



## 7.3/ các ví dụ ArrayList trong Java

Các phương thức **addAll()**, **removeAll()**, **retainAll()** của lớp ArrayList

```
import java.util.ArrayList;

public class PhuongThucArrayList1 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // Add objects to list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
    }
}
```

## 7.3/ các ví dụ ArrayList trong Java

Các phương thức `addAll()`, `removeAll()`, `retainAll()` của lớp ArrayList

```
System.out.println("ví dụ sử dụng phương thức addAll()");  
System.out.println("-----");  
// thêm các phần tử của list vào listA  
ArrayList<String> listA = new ArrayList<String>();  
listA.addAll(list);  
System.out.print("listA:");  
showList(listA);
```



## 7.3/ các ví dụ ArrayList trong Java

Các phương thức `addAll()`, `removeAll()`, `retainAll()` của lớp ArrayList

```
System.out.println("\nví dụ sử dụng phương thức retainAll()");  
System.out.println("-----");  
// khởi tạo listB  
ArrayList<String> listB = new ArrayList<String>();  
listB.add("Java");  
// xóa những phần tử không thuộc listB khỏi listA  
listA.retainAll(listB);  
System.out.print("listA:");  
showList(listA);
```

## 7.3/ các ví dụ ArrayList trong Java

Các phương thức **addAll()**, **removeAll()**, **retainAll()** của lớp ArrayList

```
System.out.println("\nví dụ sử dụng phương thức removeAll()");  
System.out.println("-----");  
// xóa những phần tử thuộc listB khỏi list  
list.removeAll(listB);  
System.out.print("list:");  
showList(list);
```

## 7.3/ các ví dụ ArrayList trong Java

Các phương thức **addAll()**, **removeAll()**, **retainAll()** của lớp ArrayList

```
public static void showList(ArrayList<String> list) {  
    // Show list through for-each  
    for (String obj : list) {  
        System.out.print("\t" + obj + ", ");  
    }  
    System.out.println();  
}  
}
```

## 7.3/ các ví dụ ArrayList trong Java

Các phương thức **addAll()**, **removeAll()**, **retainAll()** của lớp ArrayList

Kết quả:

ví dụ sử dụng phương thức addAll()  
-----

listA: Java, C++, PHP, Java,

ví dụ sử dụng phương thức retainAll()  
-----

listA: Java, Java,

ví dụ sử dụng phương thức removeAll()  
-----

list: C++, PHP,

## 7.3/ các ví dụ ArrayList trong Java

### Truy cập phần tử của ArrayList

```
import java.util.ArrayList;

public class TruyCapArrayList1 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");

        // truy cập phần tử có chỉ số 3 của list
        System.out.println(list.get(3));
    }
}
```

Kết quả:

Java

## 7.3/ các ví dụ ArrayList trong Java

### Cập nhật giá trị của phần tử Arraylist

```
import java.util.ArrayList;

public class CapNhatArrayList1 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");

        System.out.println("list trước khi cập nhật: ");
        System.out.println(list);
        // cập nhật giá trị cho phần tử có chỉ số là 3 (Java)
        list.set(3, "Python");
        System.out.println("list trước khi cập nhật: ");
        System.out.println(list);
    }
}
```

Kết quả:

```
list trước khi cập nhật:
[Java, C++, PHP, Java]
list trước khi cập nhật:
[Java, C++, PHP, Python]
```

## 7.3/ các ví dụ ArrayList trong Java

### Xóa phần tử ArrayList - Phương thức clear()

```
import java.util.ArrayList;
```

```
public class XoaArrayList1 {  
    public static void main(String[] args) {  
        // khai báo 1 ArrayList có tên là list  
        // có kiểu là String  
        ArrayList<String> list = new ArrayList<String>();  
        // thêm các phần tử vào list  
        list.add("Java");  
        list.add("C++");  
        list.add("PHP");  
        list.add("Python");  
  
        System.out.println("Số phần tử của list ban đầu : " + list);  
        System.out.println("Các phần tử của list ban đầu: " + list.size());  
        // clear list  
        list.clear();  
        System.out.println("\nSố phần tử của list sau khi clear: " + list);  
        System.out.println("Các phần tử của list sau khi clear: " + list.size());  
    }  
}
```

Kết quả:

```
Số phần tử của list ban đầu : [Java, C++, PHP, Python]  
Các phần tử của list ban đầu: 4
```

```
Số phần tử của list sau khi clear: []  
Các phần tử của list sau khi clear: 0
```

# 7.3/ các ví dụ ArrayList trong Java

## Xóa phần tử ArrayList - Phương thức remove()

```
import java.util.ArrayList;
```

Kết quả:

Số phần tử của list ban đầu : [Java, C++, PHP, Python]  
Các phần tử của list ban đầu: 4

Số phần tử của list sau khi remove phần tử có index = 1: [Java, PHP, Python]  
Các phần tử của list sau khi remove phần tử có index = 1: 3

Số phần tử của list sau khi remove phần tử "PHP": [Java, Python]  
Các phần tử của list sau khi remove phần tử "PHP": 2

```
public class XoaArrayList1 {  
    public static void main(String[] args) {  
        // khai báo 1 ArrayList có tên là list  
        // có kiểu là String  
        ArrayList<String> list = new ArrayList<String>();  
        // thêm các phần tử vào list  
        list.add("Java");  
        list.add("C++");  
        list.add("PHP");  
        list.add("Python");  
  
        System.out.println("Số phần tử của list ban đầu : " + list);  
        System.out.println("Các phần tử của list ban đầu: " + list.size());  
        // remove phần tử có chỉ số index = 1 khỏi list  
        list.remove(1);  
        System.out.println("\nSố phần tử của list sau khi remove phần tử có index = 1: " + list);  
        System.out.println("Các phần tử của list sau khi remove phần tử có index = 1: " + list.size());  
        // remove phần tử có chỉ số index = 1 khỏi list  
        list.remove("PHP");  
        System.out.println("\nSố phần tử của list sau khi remove phần tử \"PHP\": " + list);  
        System.out.println("Các phần tử của list sau khi remove phần tử \"PHP\": " + list.size());  
    }  
}
```



## 7.3/ các ví dụ ArrayList trong Java

### Tìm kiếm một phần tử ArrayList - Tìm kiếm trực tiếp phần tử.

```
import java.util.ArrayList;

public class TimKiemArrayList1 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Python");

        // kiểm tra xem PHP có tồn tại trong list hay không?
        System.out.println(list.contains("PHP"));
        // kiểm tra xem ANDROID có tồn tại trong list hay không?
        System.out.println(list.contains("ANDROID"));
    }
}
```

Kết quả:

```
true
false
```

## 7.3/ các ví dụ ArrayList trong Java

Tìm kiếm vị trí xuất hiện đầu tiên của 1 phần tử trong ArrayList.

```
import java.util.ArrayList;

public class TimKiemArrayList2 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Python");

        // kiểm tra xem Java có tồn tại trong list hay không?
        System.out.println(list.indexOf("Java"));
        // kiểm tra xem ANDROID có tồn tại trong list hay không?
        System.out.println(list.indexOf("ANDROID"));
    }
}
```

Kết quả:

0  
-1

## 7.3/ các ví dụ ArrayList trong Java

Tìm kiếm vị trí xuất hiện cuối cùng của 1 phần tử trong List.

```
import java.util.ArrayList;

public class TimKiemArrayList3 {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");

        // kiểm tra xem Java có tồn tại trong list hay không?
        System.out.println(list.lastIndexOf("Java"));
        // kiểm tra xem ANDROID có tồn tại trong list hay không?
        System.out.println(list.lastIndexOf("ANDROID"));
    }
}
```

Kết quả:

3  
-1

## 7.3/ các ví dụ ArrayList trong Java

### Chuyển ArrayList sang mảng (Array) trong Java

```
import java.util.ArrayList;

public class ConvertToArray {
    public static void main(String[] args) {
        // khai báo 1 ArrayList có tên là list
        // có kiểu là String
        ArrayList<String> list = new ArrayList<String>();
        // thêm các phần tử vào list
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");

        // sử dụng phương thức toArray() chuyển list thành mảng
        // kết quả của phương thức này sẽ trả về mảng arr
        Object[] arr = list.toArray();

        // hiển thị các phần tử có trong mảng arr
        for (int i = 0; i < arr.length; i++) {
            System.out.println("Phần tử tại vị trí " + i + " "
                               + "trong arr là " + arr[i]);
        }
    }
}
```

Kết quả:

```
Phần tử tại vị trí 0 trong arr là Java
Phần tử tại vị trí 1 trong arr là C++
Phần tử tại vị trí 2 trong arr là PHP
Phần tử tại vị trí 3 trong arr là Java
```

## 7.3/ các ví dụ ArrayList trong Java

### Tạo ArrayList có kiểu generic là String

```
import java.util.ArrayList;
import java.util.Iterator;

public class ArrayListExample1 {
    public static void main(String args[]) {
        // Creating arraylist
        ArrayList<String> list = new ArrayList<String>();
        // Add objects to arraylist
        list.add("Java");
        list.add("C++");
        list.add("PHP");
        list.add("Java");
        // Show list through Iterator
        Iterator<String> itr = list.iterator();
        while (itr.hasNext()) {
            System.out.print(itr.next() + ", ");
        }
        // Show list through for-each
        System.out.println();
        for (String obj : list) {
            System.out.print(obj + ", ");
        }
        // Show list through index
        System.out.println();
        int size = list.size();
        for (int i = 0; i < size; i++) {
            System.out.print(list.get(i) + ", ");
        }
    }
}
```

## 7.3/ các ví dụ ArrayList trong Java

Tạo ArrayList có kiểu generic là đối tượng do người dùng định nghĩa

```
import java.util.ArrayList;

class Student {
    private String name;
    private int age;
    public Student(String name, int age) {
        super();
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public String toString() {
        return "Student@[name=" + name + ", age=" + age + "]";
    }
}
```

```
public class ArrayListExample2 {
    public static void main(String[] args) {
        // Create listStudent
        ArrayList<Student> listStudent = new ArrayList<Student>();
        // Create students
        Student student1 = new Student("Bac", 17);
        Student student2 = new Student("Nam", 20);
        Student student3 = new Student("Trung", 19);
        // Add objects to listStudent
        listStudent.add(student1);
        listStudent.add(student2);
        listStudent.add(student3);
        // Show listStudent
        for (Student student : listStudent) {
            System.out.println(student.toString());
        }
    }
}
```

Output:

```
Student@[name=Bac, age=17]
Student@[name=Nam, age=20]
Student@[name=Trung, age=19]
```

## 7.4/ Sự khác nhau giữa Array và ArrayList

Array	ArrayList
1) Kích thước <b>cố định</b> .	Kích thước có thể <b>thay đổi được</b> .
2) Có thể lưu trữ dữ liệu kiểu <b>nguyên thủy</b> và <b>đối tượng</b> .	Chỉ có thể lưu trữ dữ liệu kiểu <b>đối tượng</b> . Kể từ Java 5, kiểu nguyên thủy được tự động chuyển đổi trong các đối tượng được gọi là <b>auto-boxing</b> .
3) Tốc độ lưu trữ và thao tác <b>nhanh hơn</b> .	Tốc độ lưu trữ vào thao tác <b>chậm hơn</b> .
4) Chỉ có thuộc tính <b>length</b> .	Có nhiều phương thức để thao tác với dữ liệu.

## 7.4/ Sự khác nhau giữa Array và ArrayList

Làm sao để chuyển đổi Array thành ArrayList và ngược lại?

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Example1 {
    public static void main(String[] args) {
        // create arrayList
        List<String> arrayList = new ArrayList<>();
        // adding String object to arrayList
        arrayList.add("Java");
        arrayList.add("C");
        arrayList.add("C++");
        arrayList.add("PHP");
        arrayList.add("Python");
    }
}
```



## 7.4/ Sự khác nhau giữa Array và ArrayList

Làm sao để chuyển đổi Array thành ArrayList và ngược lại?

```
// convert ArrayList to Array
System.out.println("Convert ArrayList to Array:");
String[] item = arrayList.toArray(new String[arrayList.size()]);
// show item
for (String s : item) {
    System.out.println(s);
}
```

## 7.4/ Sự khác nhau giữa Array và ArrayList

Làm sao để chuyển đổi Array thành ArrayList và ngược lại?

```
// convert Array to ArrayList
System.out.println("Convert Array to ArrayList:");
List<String> list2 = new ArrayList<>();
list2 = Arrays.asList(item);
// show list2
System.out.println(list2);
```

## 7.4/ Sự khác nhau giữa Array và ArrayList

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

```
// 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));
    }
}
```

## 7.4/ Sự khác nhau giữa Array và ArrayList

- 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));
    }
}
```

## 7.4/ Sự khác nhau giữa Array và ArrayList

- Khác biệt về đối tượng lưu trữ
- Mảng có thể chứa dữ liệu nguyên thủy như int, double, float, ... hoặc cũng có thể chứa object
- ArrayList chỉ có thể chứa object, không chứa primitive data

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

        // allowed, however, need to be initialized
        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<>();
    }
}
```

## 7.5/ Generic trong Java



## 7.5/ Generic trong Java

### 1. Các phương thức generic trong Java

Ta có thể viết một khai báo phương thức generic đơn mà có thể được gọi với các tham số của các kiểu khác nhau. Dựa trên các kiểu tham số được truyền tới phương thức generic này, bộ biên dịch xử lý mỗi lần gọi phương thức một cách thích hợp. Dưới đây là các quy tắc để định nghĩa các phương thức Generic:

- a) Tất cả khai báo phương thức generic có một khu vực tham số kiểu được giới hạn bởi các dấu ngoặc nhọn (< và >) mà đứng trước kiểu trả về của phương thức ( < E > trong ví dụ sau đây).

## 7.5/ Generic trong Java

### 1. Các phương thức generic trong Java

- b) Mỗi khu vực tham số kiểu chứa một hoặc nhiều tham số kiểu phân biệt nhau bởi dấu phẩy. Một tham số kiểu, cũng được biết như là biến kiểu, là một định danh mà xác định một tên kiểu generic.
- c) Các tham số kiểu có thể được sử dụng để khai báo kiểu trả về và hoạt động như là nơi giữ (placeholder) cho các kiểu của các tham số được truyền tới phương thức generic, mà được biết như là các tham số kiểu thực sự.
- d) Phần thân phương thức generic được khai báo giống như bất kỳ phương thức nào khác. Ghi chú rằng các tham số kiểu chỉ có thể biểu diễn các kiểu tham chiếu, không phải là các kiểu gốc (như int, double, và char).



## 7.5/ Generic trong Java

Ví dụ này minh họa cách chúng ta có thể in mảng các kiểu khác nhau bởi sử dụng một phương thức generic đơn.

```
public class GenericMethodTest {  
  
    // phương thức generic có tên là printArray  
    public static < E > void printArray(E[] inputArray) {  
        // Hiển thị các phần tử mảng  
        for (E element: inputArray) {  
            System.out.printf("%s ", element);  
        }  
        System.out.println();  
    }  
}
```

## 7.5/ Generic trong Java

Ví dụ này minh họa cách chúng ta có thể in mảng các kiểu khác nhau bởi sử dụng một phương thức generic đơn.

```
public static void main(String args[]) {  
    // Tao cac mang Integer, Double va Character  
    Integer[] intArray = { 1, 2, 3, 4, 5 };  
    Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };  
    Character[] charArray = { 'H', 'E', 'L', 'L', 'O' };  
  
    System.out.println("Mang intArray bao gom:");  
    printArray(intArray); // truyen mot mang Integer  
  
    System.out.println("\nMang doubleArray bao gom:");  
    printArray(doubleArray); // truyen mot mang Double  
  
    System.out.println("\nMang charArray bao gom:");  
    printArray(charArray); // truyen mot mang Character  
}
```

Nó sẽ cho kết quả:

Mang intArray bao gom: 1 2 3 4 5 6

Mang doubleArray bao gom: 1.1 2.2 3.3 4.4

Mang charArray bao gom: H E L L O

## 7.5/ Generic trong Java

Ví dụ sau minh họa cách extends được sử dụng: hoặc "extends" khi trong các lớp hoặc "implements" khi trong các interface. Ví dụ về phương thức generic trả về giá trị lớn nhất trong ba đối tượng Comparable.

```
public class MaximumTest {  
    // xác định giới hạn max của các đối tượng Comparable  
    public static < T extends Comparable < T >> T maximum(T x, T y, T z) {  
        T max = x; // giả sử ban đầu x là lớn nhất  
        if (y.compareTo(max) > 0) {  
            max = y; // y là lớn nhất  
        }  
        if (z.compareTo(max) > 0) {  
            max = z; // bây giờ z là lớn nhất  
        }  
        return max; // trả về đối tượng lớn nhất  
    }  
}
```

## 7.5/ Generic trong Java

Ví dụ sau minh họa cách extends được sử dụng: hoặc "extends" khi trong các lớp hoặc "implements" khi trong các interface. Ví dụ về phương thức generic trả về giá trị lớn nhất trong ba đối tượng Comparable.

```
1 public static void main(String args[]) {  
    System.out.printf("Max của %d, %d và %d là %d\n\n", 3, 4, 5, maximum(3, 4, 5));  
  
    System.out.printf("Max của %.1f, %.1f và %.1f là %.1f\n\n", 6.6, 8.8, 7.7, maximum(6.6, 8.8, 7.7));  
  
    System.out.printf("Max của %s, %s và %s là %s\n", "pear", "apple", "orange", maximum("pear", "apple", "orange"));  
2 }  
}
```

# 7.5/ Generic trong Java

## 2. Các lớp Generic trong Java.

- Một khai báo lớp Generic trông giống như một khai báo lớp không phải Generic, ngoại trừ là tên lớp được theo sau bởi một khu vực tham số kiểu.
- Như với phương thức Generic, khu vực tham số kiểu của một lớp Generic có thể có một hoặc nhiều tham số kiểu phân biệt nhau bởi dấu phẩy. Những lớp này còn được biết như là các lớp tham số hóa hoặc các kiểu tham số hóa bởi vì chúng chấp nhận một hoặc nhiều tham số.

# 7.5/ Generic trong Java

## 2. Các lớp Generic trong Java.

Ví dụ này minh họa cách chúng ta định nghĩa một lớp Generic trong Java:

```
public class Box < T > {  
    private T t;  
  
    public void add(T t) {  
        this.t = t;  
    }  
  
    public T get() {  
        return t;  
    }  
}
```

```
public static void main(String[] args) {  
    Box < Integer > integerBox = new Box < Integer > ();  
    Box < String > stringBox = new Box < String > ();  
  
    integerBox.add(new Integer(10));  
    stringBox.add(new String("Hello World"));  
  
    System.out.printf("Gia tri integer la :%d\n\n", integerBox.get());  
    System.out.printf("Gia tri string la :%s\n", stringBox.get());  
}
```

# Câu hỏi thảo luận

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

- Jose M. Garrido, “**Object-Oriented Programming: From Problem Solving to Java**”
- Paul Deitel, Harvey Deitel, “**Java : How to program**”, 9<sup>th</sup> edition, 2012
- Oracle, “**The Java™ Tutorials**”,  
<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html>, 6:20PM, 18/01/2018
- Java tutorial, <https://www.javatpoint.com/java-tutorial> , 6:20PM, 18/01/2018