

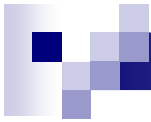


# Một số lớp và thư viện của Java



# Nội dung

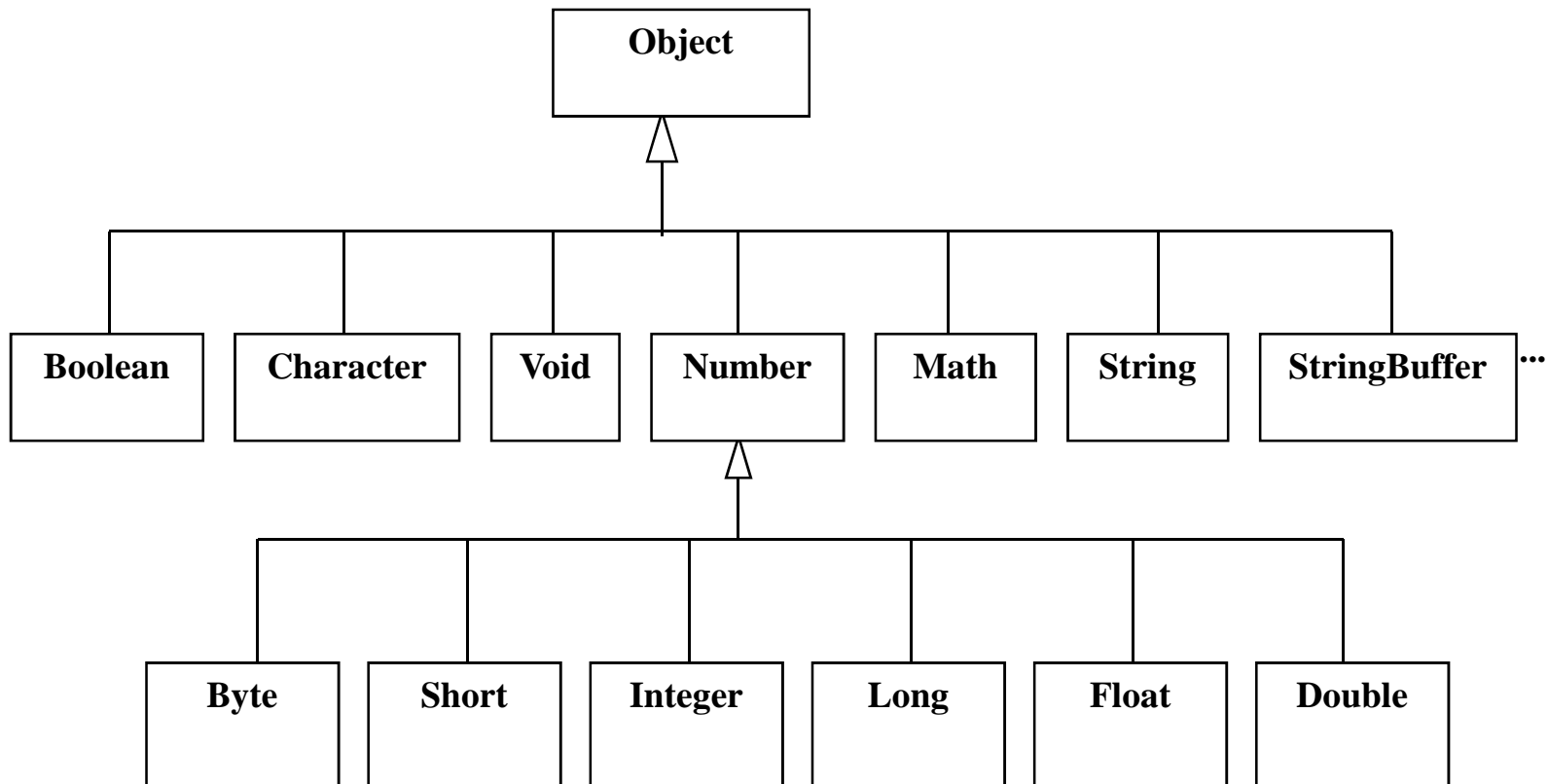
- Các lớp dữ liệu nguyên thủy
- Xâu ký tự
- Lớp Math
- Mảng
- Các lớp Container



# Tài liệu tham khảo

- Bruce Eckel, *Thinking in Java*, chapter 11
- Deitel, *Java – How to program*, chapter 7, 11, 20
- *Giáo trình Lập trình HĐT*, chương 6

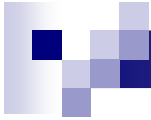
# Một số lớp cơ bản





# Lớp Object

- `Class getClass()`: trả lại tên lớp của đối tượng hiện thời.
- `boolean equals(Object)`: so sánh đối tượng, thường được định nghĩa lại.
- `String toString()`: trả lại biểu diễn văn bản của đối tượng, thường được định nghĩa lại.



```
Person p = new Person("John");  
Class c = p.getClass();  
System.out.println(c);  
----  
class Person
```

# Các lớp dữ liệu nguyên thủy

## ■ Các phương thức tiện ích

- `valueOf(String s)`: trả đối tượng thuộc kiểu tương ứng

```
Integer k = Integer.valueOf( "12"); // k = 12
```

- `intValue()`: trả giá trị nguyên thủy tương ứng

```
int i = k.intValue(); // i = 12
```

- `parseInt(String s)`: trả giá trị nguyên thủy tương ứng

```
int i = Integer.parseInt("12"); // i = 12
```

## ■ Hằng số

- `Integer.MAX_VALUE`, `Integer.MIN_VALUE`



# Lớp Character

## ■ Các phương thức

- ☐ `static boolean isUppercase(char ch)`
- ☐ `static boolean isLowercase(char ch)`
- ☐ `static boolean isDigit(char ch)`
- ☐ `static boolean isLetter(char ch)`
- ☐ `static boolean isLetterOrDigit(char ch)`
- ☐ `static char toUpperCase(char ch)`
- ☐ `static char toLowerCase(char ch)`





# Lớp String

- Xâu ký tự không thay đổi được nội dung
- Khởi tạo
  - `String(String),`  
`String(StringBuffer)`
  - `String(byte[]), String(char[])`
- Phương thức
  - `int length()`: kích thước của xâu
  - `char charAt(int index)`: ký tự ở vị trí `index`



# Lớp String

## ■ So sánh

- ☐ `boolean equals(String)`
- ☐ `boolean equalsIgnoreCase(String)`
- ☐ `boolean startsWith(String)`
- ☐ `boolean endsWith(String)`
- ☐ `int compareTo(String)`



# Lớp String

## ■ Chuyển đổi

- `String toUpperCase()`

- `String toLowerCase()`

## ■ Ghép xâu

- `String concat(String)`

- toán tử “+”



# Lớp String

## ■ Tìm kiếm

- ☐ `int indexOf(char), int  
indexOf(char ch, int from)`
- ☐ `int indexOf(String), int  
indexOf(String s, int from)`
- ☐ `int lastIndexOf(char),  
lastIndexOf(char, int)`
- ☐ `lastIndexOf(String),  
lastIndexOf(String, int)`



# Lớp String

## ■ Thay thế

- `String replace(char ch, char new_ch)`

## ■ Trích xuất

- `String trim()`: loại bỏ ký tự trắng

- `String substring(int startIndex)`

- `String substring(int startIdx, int endIdx)`



# Lớp StringBuffer

- Xâu ký tự thay đổi được nội dung
- Khởi tạo
  - `StringBuffer(String)`
  - `StringBuffer(int length)`
  - `StringBuffer()`: đặt kích thước mặc định 16
- Các phương thức
  - `int length(), void setLength()`
  - `char charAt(int index)`
  - `void setCharAt(int index, char ch)`
  - `String toString()`



# Lớp StringBuffer

## ■ Thêm, xóa

- `append(String), append(type)`
- `insert(int offset, String s),`  
`insert(int offset, char[] chs),`  
`insert(int offset, type t)`
- `delete(int start, int end):` xóa xâu con
- `delete(int index):` xóa một ký tự
- `reverse():` đảo ngược



# Lớp Math

- Hằng số

- ☐ `Math.E`
- ☐ `Math.PI`

- Các phương thức static

- ☐ `type abs(type)`
- ☐ `double ceil(double), double floor(double)`
- ☐ `int round(float), long round(double)`
- ☐ `type max(type, type), type min(type, type)`
- ☐ `double random()`: sinh số ngẫu nhiên trong đoạn `[0.0,1.0]`





# Lớp Math

## ■ Lũy thừa

- ☐ `double pow(double, double)`
- ☐ `double exp(double)`
- ☐ `double log(double)`
- ☐ `double sqrt(double)`

## ■ Lượng giác

- ☐ `double sin(double)`
- ☐ `double cos(double)`
- ☐ `double tan(double)`

# Mảng

- Mảng là đối tượng
  - chứa một tập các đối tượng khác
  - cần tạo ra trước khi sử dụng (new)

Ví dụ:

```
int a[];  
a = new int[10];  
for (int i=0; i<a.length; i++) a[i] = i*i;  
  
int b[] = {2, 3, 5, 7};  
a = b;  
int m, n[];  
double[] arr1, arr2;
```



# Truyền tham số và nhận giá trị trả lại

```
int[] myCopy(int[] a) {  
    int b[] = new int[a.length];  
    for (i=0; i<a.length; i++)  
        b[i] = a[i];  
    return b;  
}  
...  
int a[] = {0, 1, 1, 2, 3, 5, 8};  
int b[] = myCopy(a);
```



# Mảng nhiều chiều

```
int a[][];  
a = new int[10][20];  
a[2][3] = 10;  
for (int i=0; i<a[0].length; i++)  
    a[0][i] = i;
```

```
int b[][] = { {1 , 2}, {3, 4} };  
int c[][] = new int[2][];  
c[0] = new int[5];  
c[1] = new int[10];
```



# Copy mảng

- `System.arraycopy(src, s_off, des, d_off, len)`
  - `src`: mảng nguồn, `s_off`: offset của mảng nguồn
  - `des`: mảng đích, `d_off`: offset của mảng đích
  - `len`: số phần tử cần copy
- Copy nội dung của dữ liệu nguyên thủy, copy tham chiếu đối với đối tượng



# Lớp Arrays

- Nằm trong gói `java.util`
- Cung cấp các phương thức static để làm việc với mảng
  - `fill()`: khởi tạo các phần tử của mảng với một giá trị như nhau
  - `sort()`: sắp xếp mảng
  - `equals()`: so sánh hai mảng
  - `binarySearch()`: tìm kiếm nhị phân trên mảng đã sắp xếp



# So sánh mảng equals()

- So sánh mảng dữ liệu nguyên thủy
- Gọi phương thức `equals()` để so sánh mảng đối tượng

----

```
int a[] = { 1, 3 , 2 , 4 };  
int b[] = new int[a.length];  
System.arraycopy(a,0,b,0,a.length);  
System.out.println(Arrays.equals(a,b));
```



# Sắp xếp mảng `sort ( )`

- Làm việc với các mảng dữ liệu nguyên thủy

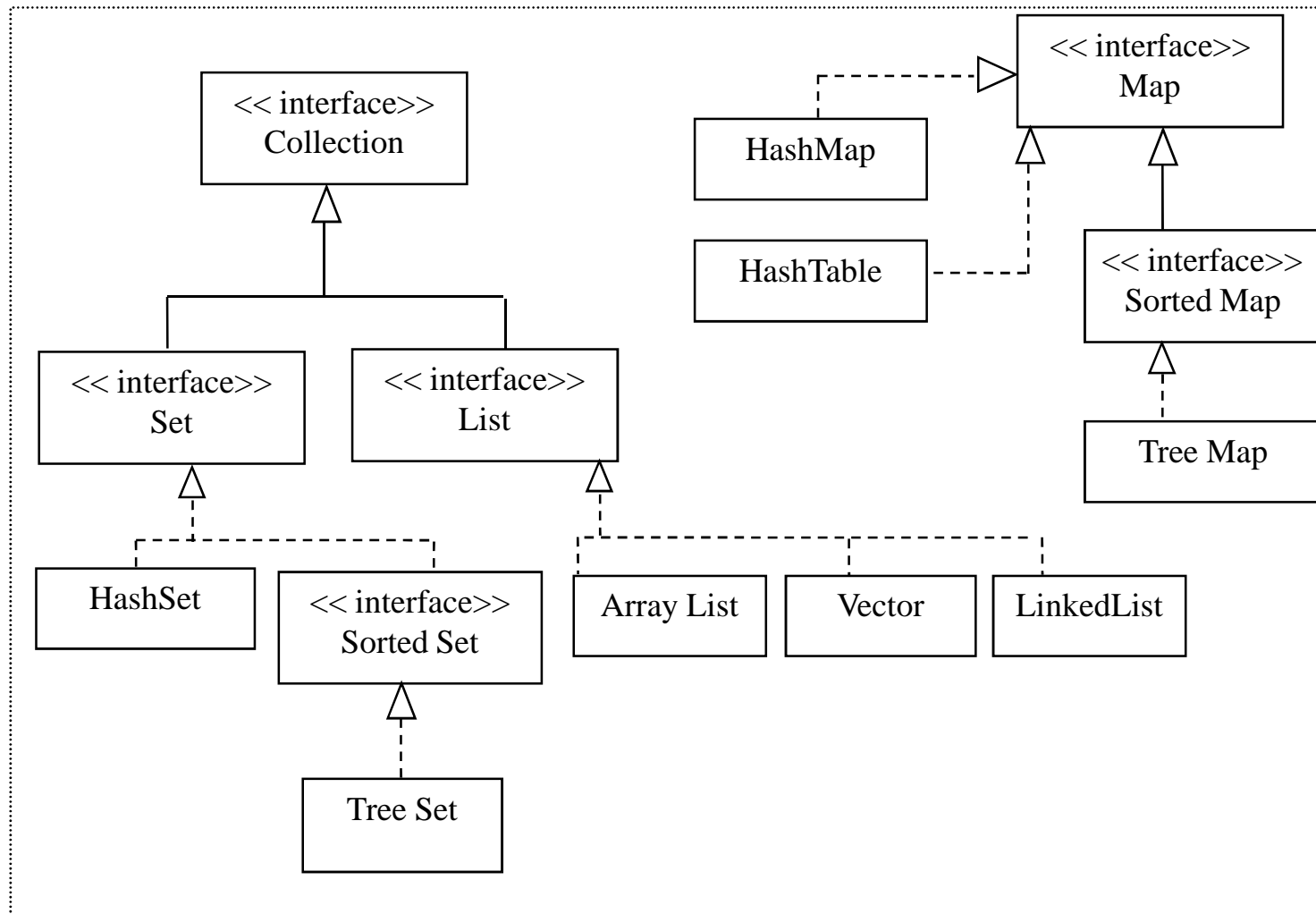
`Arrays.sort(a);`

- Làm việc với các lớp đối tượng có cài đặt giao diện `Comparable`

- phương thức `int compareTo(Object)` trả lại giá trị âm, 0, hoặc dương



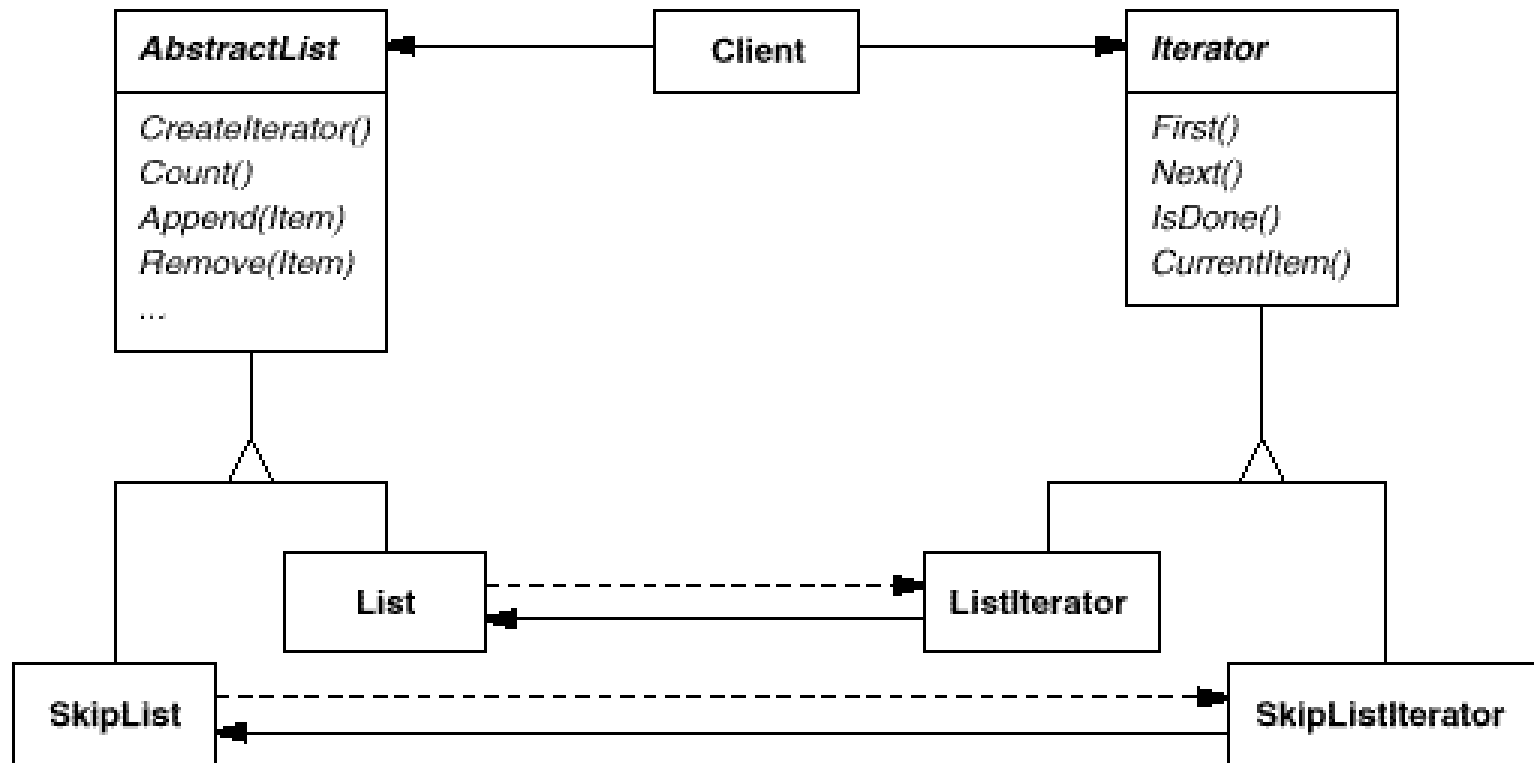
# Các lớp tuyển tập (Container)






# Iterator

- Mẫu dùng để duyệt các phần tử của một tập hợp
- Là một interface trong Java:
  - ☐ hasNext()
  - ☐ next()
  - ☐ remove()
- Các lớp Collection cài đặt Iterator






```
import java.util.*;

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

        Collection list = new LinkedList();

        list.add(3);
        list.add(2);
        list.add(1);
        list.add(0);
        list.add("happy new year!");

        Iterator i = list.iterator();
        while (i.hasNext()) {
            System.out.println(i.next());
        }
    }
}
```



```
import java.util.*;

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

        List list = new LinkedList();

        list.add(3);
        list.add(2);
        list.add(1);
        list.add(0);
        list.add("go!");

        for (int i=0; i<list.size(); i++) {
            System.out.println(list.get(i));
        }
    }
}
```