



Lập trình Java

BÀI 2

Java căn bản

Nội dung



- Biến & Hằng
- Kiểu dữ liệu (kiểu cơ sở, kiểu tham chiếu)
- Toán tử, biểu thức
- Các cấu trúc điều khiển (chọn, rẽ nhánh, lặp)
- Lớp bao kiểu cơ sở
- Phương thức và cách sử dụng
- Một số ví dụ minh họa

- Biến là một vùng nhớ lưu các giá trị của chương trình
- Mỗi biến gắn với 1 kiểu dữ liệu và 1 định danh duy nhất là tên biến
- Tên biến phân biệt chữ hoa và chữ thường. Tên biến bắt đầu bằng 1 dấu _, \$, hay 1 ký tự, không được bắt đầu bằng 1 ký số.

Khai báo

<kiểu dữ liệu> <tên biến>;

<kiểu dữ liệu> <tên biến> = <giá trị>;

Gán giá trị

<tên biến> = <giá trị>;

Phân loại biến



- Biến trong Java có 2 loại:
 - local variable
 - instance variable
- Local variable
 - Được khai báo trong một phương thức, constructor
 - Không được chương trình gán giá trị khi khai báo
- instance variable
 - Được khai báo trong lớp, nhưng ngoài các phương thức
 - Instance variable được tạo khi đối tượng được tạo
 - Được tự động gán giá trị mặc định

Ví dụ Local variables



```
public class Test{
    public void pupAge()
    {
        int age = 0;
        age = age + 7;
        System.out.println("Puppy age is : " + age);
    }
    public static void main(String args[])
    {

        Test test = new Test();
        test.pupAge();
    }
}
```

Ví dụ Local variables



```
public class Test {  
    public static void pupAge()  
    {  
        int age;  
        age = age + 7;  
        System.out.println("Puppy age is : " + age);  
    }  
    public static void main(String args[])  
    {  
  
        Test test = new Test();  
        Test.pupAge();  
    }  
}
```

Lỗi

Ví dụ Instance variables



```
class Employee{
    public String name;
    private double salary.
    public Employee (String empName){
        name = empName;
    }
    public void setSalary(double empSal){
        salary = empSal;
    }
    public void printEmp(){
        System.out.println("name : " + name );
        System.out.println("salary :" + salary);  }
    public static void main(String args[]){
        Employee empOne = new Employee("Ransika");
        empOne.setSalary(1000);
        empOne.printEmp();
    }
}
```

- Là một giá trị bất biến trong chương trình
- Tên đặt theo qui ước như tên biến
- Được khai báo dùng từ khóa **final**, và thường dùng tiếp vĩ ngữ đối với các hằng số (kiểu long (l,L), kiểu double (d,D), kiểu float (f,F))

- Ví dụ:

final int x = 10; // khai báo hằng số nguyên x = 10

final long y = 20L; // khai báo hằng số long y = 20

- **Hằng ký tự**: đặt giữa cặp nháy đơn “
- **Hằng chuỗi**: là một dãy ký tự đặt giữa cặp nháy đôi “”

Hàng ký tự đặc biệt



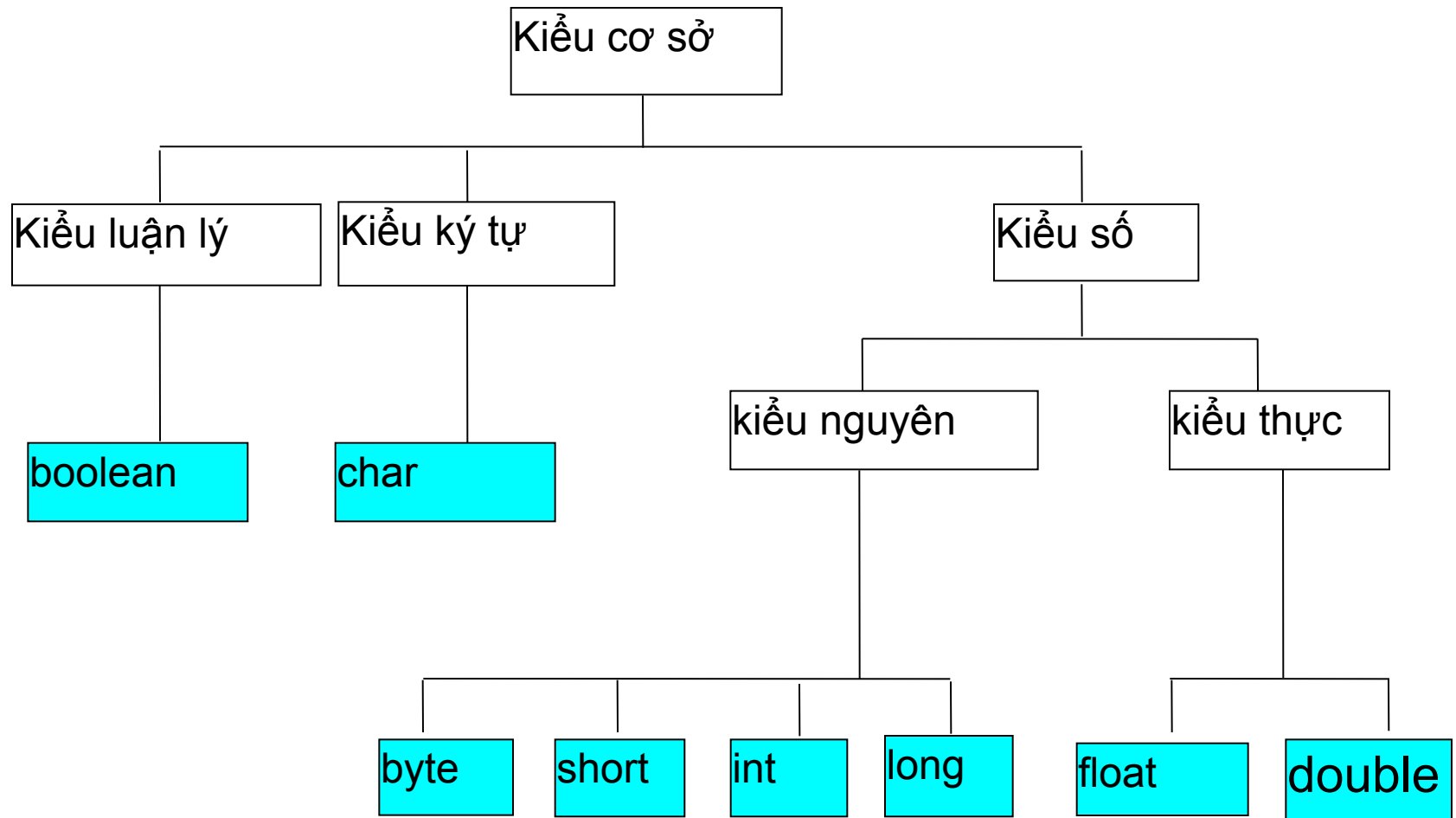
<i>Ký tự</i>	<i>Ý nghĩa</i>
\b	Xóa lùi (BackSpace)
\t	Tab
\n	Xuống hàng
\r	Dấu enter
\"	Nháy kép
'	Nháy đơn
\\	\
\f	Đẩy trang
\uxxxx	Ký tự unicode

Kiểu dữ liệu



- Kiểu dữ liệu cơ sở (primitive data type)
- Kiểu dữ liệu tham chiếu (reference data type)

Kiểu dữ liệu cơ sở



Kiểu dữ liệu cơ sở



Kiểu	Kích thước (bits)	Giá trị	Giá trị mặc định
boolean	<i>[Note: The representation of a boolean is specific to the Java Virtual Machine on each computer platform.]</i>	true và false	false
char	16	'\u0000' to '\uFFFF' (0 to 65535)	null
byte	8	-128 to +127 (-2^7 to $2^7 - 1$)	0
short	16	-32,768 to +32,767 (-2^{15} to $2^{15} - 1$)	0
int	32	-2,147,483,648 to +2,147,483,647 (-2^{31} to $2^{31} - 1$)	0
long	64	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807 (-2^{63} to $2^{63} - 1$)	0l
float	32	1.40129846432481707e-45 to 3.4028234663852886E+38	0.0f
double	64	4.94065645841246544e-324 to 1.7976931348623157E+308	0.0d

Kiểu dữ liệu cơ sở

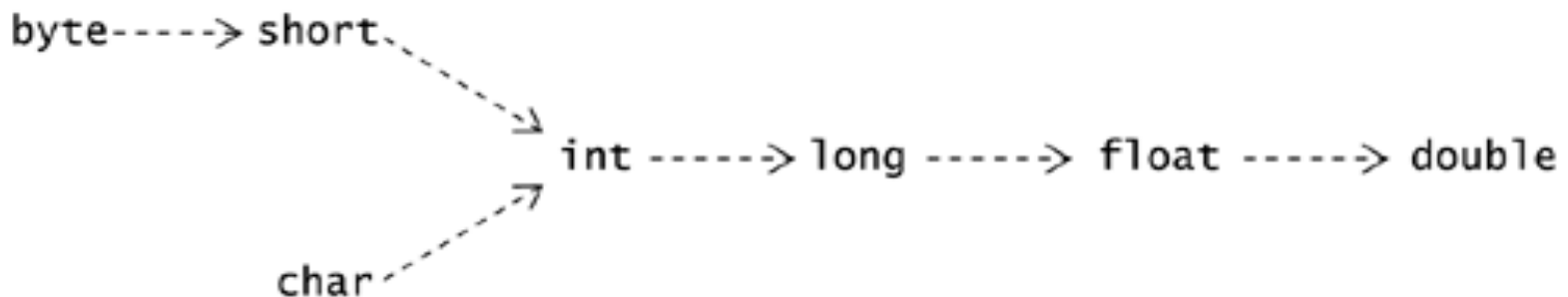


- **Chuyển đổi kiểu dữ liệu:** khi có sự không tương thích về kiểu dữ liệu (gán, tính toán biểu thức, truyền đối số gọi phương thức)

✓ Chuyển kiểu hẹp (lớn  nhỏ): **cần ép kiểu**

<tên biến 2> = (kiểu dữ liệu) <tên biến 1>;

✓ Chuyển kiểu rộng (nhỏ  lớn): **tự động chuyển**



Kiểu dữ liệu cơ sở



- **Lưu ý**

1. Không thể chuyển đổi giữa kiểu *boolean* với *int* và ngược lại.

2. Nếu 1 toán hạng kiểu ***double*** thì

“Toán hạng kia chuyển thành ***double***”

Nếu 1 toán hạng kiểu ***float*** thì

“Toán hạng kia chuyển thành ***float***”

Nếu 1 toán hạng kiểu ***long*** thì

“Toán hạng kia chuyển thành ***long***”

Ngược lại “Tất cả chuyển thành ***int*** để tính toán”

Kiểu dữ liệu cơ sở



- **Ví dụ minh họa**

1. *byte x = 5;*

2. *byte y = 10;*

3. *byte z = x + y;*

// Dòng lệnh thứ 3 báo lỗi chuyển kiểu cần sửa lại

// byte z = (byte) (x + y);

Kiểu dữ liệu cơ sở



- **Kiểu mảng**

- ✓ Mảng là tập hợp các phần tử có cùng tên và cùng kiểu dữ liệu.
- ✓ Mỗi phần tử được truy xuất thông qua chỉ số

- **Khai báo mảng**

<kiểu dữ liệu>[] <tên mảng>; // mảng 1 chiều

<kiểu dữ liệu> <tên mảng>[]; // mảng 1 chiều

<kiểu dữ liệu>[][] <tên mảng>; // mảng 2 chiều

<kiểu dữ liệu> <tên mảng>[][]; // mảng 2 chiều

Kiểu dữ liệu tham chiếu



- **Khởi tạo**

`int arrInt[] = {1, 2, 3};`

`char arrChar[] = {'a', 'b', 'c'};`

`String arrString[] = {"ABC", "EFG", "GHI"};`

- **Cấp phát & truy cập mảng**

`int [] arrInt = new int[100];`

`int arrInt[100];` // Khai báo này trong Java sẽ bị báo lỗi.

- Chỉ số mảng n phần tử: từ 0 đến $n-1$

Kiểu dữ liệu tham chiếu



- Kiểu đối tượng

Khai báo đối tượng

<Kiểu đối tượng> <biến ĐT>;

Khởi tạo đối tượng

*<Kiểu đối tượng> <biến ĐT> = **new** <Kiểu đối tượng>;*

Truy xuất thành phần đối tượng

<biến ĐT>.<thuộc tính>

<biến ĐT>.<phương thức>

Toán tử, biểu thức



- Toán tử số học

<i>Toán tử</i>	<i>Ý nghĩa</i>
+	Cộng
-	Trừ
*	Nhân
/	Chia nguyên
%	Chia dư
++	Tăng 1
--	Giảm 1

Toán tử, biểu thức



- **Phép toán trên bit**

<i>Toán tử</i>	<i>Ý nghĩa</i>
&	AND
 	OR
^	XOR
<<	Dịch trái
>>	Dịch phải
~	Bù bit

Toán tử, biểu thức



- Toán tử quan hệ & logic

<i>Toán tử</i>	<i>Ý nghĩa</i>
<code>==</code>	<i>So sánh bằng</i>
<code>!=</code>	<i>So sánh khác</i>
<code>></code>	<i>So sánh lớn hơn</i>
<code><</code>	<i>So sánh nhỏ hơn</i>
<code>>=</code>	<i>So sánh lớn hơn hay bằng</i>
<code><=</code>	<i>So sánh nhỏ hơn hay bằng</i>
<code> </code>	<i>OR (biểu thức logic)</i>
<code>&&</code>	<i>AND (biểu thức logic)</i>
<code>!</code>	<i>NOT (biểu thức logic)</i>

Toán tử, biểu thức



- Toán tử gán

<i>Toán tử</i>	<i>Ví dụ</i>	<i>Ý nghĩa</i>
=	$a = b$	$gán\ a = b$
+=	$a += 5$	$a = a + 5$
-=	$b -= 10$	$b = b - 10$
*=	$c *= 3$	$c = c * 3$
/=	$d /= 2$	$d = d/2$
%=	$e \% = 4$	$e = e \% 4$

Toán tử, biểu thức



- Toán tử điều kiện

Cú pháp: *<điều kiện> ? <biểu thức 1> : <biểu thức 2>*

Ví dụ:

```
int x = 10;
```

```
int y = 20;
```

```
int Z = (x < y) ? 30 : 40;
```

```
// Kết quả z = 30 do biểu thức (x < y) là đúng.
```

Nhập xuất trong java



- Xuất

- `System.out.println();`

- Nhập

- `BufferedReader br= new BufferedReader(new
InputStreamReader(System.in));`

- `String number=br.readLine();`

Kiểu dữ liệu chuỗi

Ví dụ nhập xuất



```
package nhapxuat; import java.io.BufferedReader; import java.io.IOException;
import java.io.InputStreamReader;
public class NhapXuat {
public static int a, b,tong,tich,hieu,thuong;
public static void main(String[] args) throws IOException{
    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
    String str;
    System.out.println("Nhap a:");
    str=br.readLine();
    a=Integer.parseInt(str);
    System.out.println("Nhap b:");
    str=br.readLine();
    b=Integer.parseInt(str);
    tong=a+b;
    System.out.println("tong "+tong);
}
}
```

Nhập xuất từ console



Lớp java.util.Scanner

<i>public boolean</i>	<i>nextBoolean()</i>
<i>public byte</i>	<i>nextByte()</i>
<i>public byte</i>	<i>nextByte(int radix)</i>
<i>public double</i>	<i>nextDouble()</i>
<i>public float</i>	<i>nextFloat()</i>
<i>public int</i>	<i>nextInt()</i>
<i>public int</i>	<i>nextInt(int radix)</i>
<i>public <u>String</u></i>	<i>nextLine()</i>
<i>public long</i>	<i>nextLong()</i>
<i>public long</i>	<i>nextLong(int radix)</i>
<i>public short</i>	<i>nextShort()</i>
<i>public short</i>	<i>nextShort(int radix)</i>

Ví dụ nhập sử dụng lớp Scanner



```
Scanner in = new Scanner(System.in);
```

```
int number = in.nextInt();
```

```
String string = in.nextLine();
```

```
float real = in.nextFloat();
```

```
String string2 = in.nextLine();
```

Ví dụ lớp bao kiểu dữ liệu



```
System.out.println();*/  
Float a=new Float(5.5);  
System.out.println(a.floatValue());
```

```
Float fObj1 = new Float("5.35");  
Float fObj2 = new Float("5.34");  
int i2 = fObj1.compareTo(fObj2);
```

```
if(i2 > 0){  
    System.out.println("First is grater");  
}else if(i2 < 0){  
    System.out.println("Second is grater");  
}else{  
    System.out.println("Both are equal");  
}  
}
```

Lớp bao kiểu dữ liệu



Data type	Wrapper Class (java.lang.*)	Ghi chú
boolean	Boolean	
byte	Byte	<ul style="list-style-type: none">- Gói (package): chứa nhóm nhiều class.- Ngoài các Wrapper Class, gói java.lang còn cung cấp các lớp nền tảng cho việc thiết kế ngôn ngữ java như: String, Math, ...
short	Short	
char	Character	
int	Integer	
long	Long	
float	Float	
double	Double	

Cấu trúc điều khiển



- **Cấu trúc *if ... else***

Dạng 1: *if* (<điều_kiện>) {
 <khởi_lệnh>;

}

Dạng 2: *if* (<điều_kiện>) {
 <khởi_lệnh1>;

}

else {
 <khởi_lệnh2>;

}

Cấu trúc điều khiển



- **Cấu trúc *switch ... case***

```
switch (<biến>) {  
    case <giá trị_1>:  
        <khởi_lệnh_1>;  
        break;  
  
    ....  
    case <giá trị_n>:  
        <khởi_lệnh_n>;  
        break;  
    default:  
        <khởi_lệnh default>;  
}
```

Cấu trúc điều khiển



- **Cấu trúc lặp**

- **Dạng 1:** `while (<điều_kiện_lặp>) {`
`<khởi_lệnh>;`
`}`

- **Dạng 2:** `do {`
`<khởi_lệnh>;`
`} while (điều_kiện);`

- **Dạng 3:** `for (khởi_tạo_biến_đếm;đk_lặp;tăng_biến) {`
`<khởi_lệnh>;`
`}`

Cấu trúc điều khiển



- **Cấu trúc lệnh nhảy jump:** dùng kết hợp nhãn (label) với từ khóa ***break*** và ***continue*** để thay thế cho lệnh ***goto*** (trong C).

Ví dụ:

```
label:
for (...) {
    for (...) {
        if (<biểu thức điều kiện>)
            break label;
        else
            continue label;
    }
}
```

static trong java



- Phần mở rộng **static** được khai báo trước kiểu dữ liệu của thuộc tính, trước kiểu dữ liệu trả về của phương thức
- Phương thức **static** chỉ truy xuất đến các thành phần có phần mở rộng **static**
- Thuộc tính có phần mở rộng **static** được tạo ra một thể hiện cho tất cả các lớp

Ví dụ static



```
Class PhanSo
{
    public static int Tu;
    void f(){ Tu=10;}
    public static void ln(){System.out.println(Tu);}
}
public class LopInner {
    public static void main(String[] args)
    {
        PhanSo p1=new PhanSo();
        p1.f();
        p1.ln();
    }
}
```

Output
10

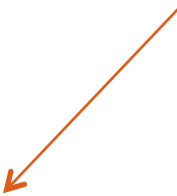
Ví dụ static



```
class PhanSo
{
    public int Tu;
    void f(){ Tu=10;}
    public static void ln(){System.out.println(Tu);}
}

public class LopInner {
    public static void main(String[] args)
    {
        PhanSo p1=new PhanSo();
        p1.f();
        p1.ln();
    }
}
```

Chương trình báo lỗi

A red arrow originates from the text box and points to the `p1.ln();` line in the code, indicating that this is the source of the error because it is calling a static method on a non-static object.

Ví dụ static



```
public class LopInner {  
    public static void main(String[] args)  
    {  
        PhanSo p1=new PhanSo();  
        PhanSo p2= new PhanSo();  
        p1.f();  
        p1.ln();  
        p2.ln();  
    }  
}
```

Output

10

10

Không có static

```
class PhanSo{
    public int Tu;
    void f()
    { Tu=10;
    }
    public void ln()
    {
        System.out.println(Tu);
    }
}
```

```
public class LopInnner {
    public static void main
    (String[] args)
    {
        PhanSo
        p1=new PhanSo();
        PhanSo p2=
        new PhanSo();
        p1.f();
        p1.ln();
        p2.ln();
    }
}
```

Output

10

0

ArrayList



- ArrayList sử dụng một mảng động để lưu trữ các phần tử.
- Khai báo một ArrayList
 - `ArrayList al=new ArrayList();`
 - `ArrayList<String> al =new ArrayList<String>();`

Các hàm trong ArrayList



- Thêm 1 phần tử:
 - add(Object);
 - add(index, Object)

```
class TestCollection2{
    public static void main(String args[]){
        ArrayList<String> al=new ArrayList<String>();
        al.add("HTTT");
        al.add("CNPM");
        al.add("MMT");
        al.add("KHMT");
        for(String obj:al)
            System.out.println(obj);
    }
}
```


Xóa một phần tử



- remove(Object);
- remove(index)

```
class TestCollection2{
    public static void main(String args[]){
        ArrayList<String> al=new ArrayList<String>();
        al.add("HTTT");
        al.add("CNPM");
        al.add("MMT");
        al.add("KHMT");
        al.remove(1);
        for(String obj:al)
            System.out.println(obj);
    }
}
```

Sửa một phần tử



- `set(int index, Object);`

```
class TestCollection2{
    public static void main(String args[]){
        ArrayList<String> al=new ArrayList<String>();
        al.add("HTTT");
        al.add("CNPM");
        al.add("MMT");
        al.add("KHMT");
        al.set(1,KTMT);
        for(String obj:al)
            System.out.println(obj);
    }
}
```

Lấy vị trí một phần tử



- `int indexOf(Object);`

```
class TestCollection2{
    public static void main(String args[]){
        ArrayList<String> al=new ArrayList<String>();
        al.add("HTTT");
        al.add("CNPM");
        al.add("MMT");
        al.add("KHMT");
        int i=indexOf ("HTTT");
        System.out.print("Vị trí lấy được"+i);
    }
```

Lấy phần tử tại vị trí cho trước



- Object get(int index);

```
class TestCollection2{  
    public static void main(String args[]){  
        ArrayList<String> al=new ArrayList<String>();  
        al.add("HTTT");  
        al.add("CNPM");  
        al.add("MMT");  
        al.add("KHMT");  
        String str=get (0);  
        System.out.print("Phần tử lấy được"+str);  
    }  
}
```

Lấy kích thước của ArrayList



- `int obj.size();`

```
class TestCollection2{
    public static void main(String args[]){
        ArrayList<String> al=new ArrayList<String>();
        al.add("HTTT");
        al.add("CNPM");
        al.add("MMT");
        al.add("KHMT");
        int num=al.size();
        System.out.print("Số phần tử"+num);
    }
```

Xóa tất cả các phần tử



- `obj.clear();`

```
class TestCollection2{  
    public static void main(String args[]){  
        ArrayList<String> al=new ArrayList<String>();  
        al.add("HTTT");  
        al.add("CNPM");  
        al.add("MMT");  
        al.add("KHMT");  
        al.clear();  
    }  
}
```

Nối 2 ArrayList



- `ArrayList.addAll(ArrayList);`

```
ArrayList<String> al=new ArrayList<String>();  
al.add("Ravi");  
al.add("Vijay");  
al.add("Ajay");
```

```
ArrayList<String> al2=new ArrayList<String>();  
al2.add("Sonoo");  
al2.add("Hanumat");
```

```
al.addAll(al2);
```

Xóa các phần tử nằm trong ArrayList khác



- `ArrayList.removeAll(ArrayList);`

```
ArrayList<String> al=new ArrayList<String>();  
al.add("Ravi");  
al.add("Vijay");  
al.add("Ajay");
```

```
ArrayList<String> al2=new ArrayList<String>();  
al2.add("Ravi");  
al2.add("Hanumat");
```

```
al.removeAll(al2);
```


Giữ lại phần tử có chứa trong ArrayList khác



- `ArrayList.retainAll (ArrayList);`

```
ArrayList<String> al=new ArrayList<String>();  
al.add("Ravi");  
al.add("Vijay");  
al.add("Ajay");
```

```
ArrayList<String> al2=new ArrayList<String>();  
al2.add("Ravi");  
al2.add("Hanumat");
```

```
al.retainAll(al2);
```

Mỗi phần tử là 1 Object



```
class TestCollection2{  
    public static void main(String args[]){  
        ArrayList<SinhVien> al=new ArrayList<SinhVien>();  
    }  
}
```

Lớp LinkedList



- ❖ Dùng danh sách liên kết đơn để lưu các đối tượng
- ❖ Khai báo:
 - ✧ `LinkedList<String> l=new LinkedList<String>();`

Hàm thêm vào cuối



❖ void add(Object)

```
public static void main(String args[]){  
    LinkedList<String> al=new LinkedList<String>();  
    al.add("Mai");  
    al.add("Nhưng");  
    al.add("Nhi");  
    al.add("Lan");  
    Iterator<String> itr=al.iterator();  
    while(itr.hasNext()){  
        System.out.println(itr.next());  
    }  
}
```

Hàm thêm vào vị trí index



❖ void add(index, Object)

```
public static void main(String args[]){  
    LinkedList<String> al=new LinkedList<String>();  
    al.add("Mai");  
    al.add("Nhưng");  
    al.add("Nhi");  
    al.add(2,"Lan");  
    Iterator<String> itr=al.iterator();  
    while(itr.hasNext()){  
        System.out.println(itr.next());  
    }  
}
```

Nối ArrayList vào LinkedList



❖ `boolean list1.addAll(ArrayList);`

```
LinkedList<String> llistobj = new LinkedList<String>();  
ArrayList<String> arraylist= new ArrayList<String>();  
arraylist.add("String1");  
arraylist.add("String2");  
llistobj.addAll(arraylist);
```

Nối ArrayList vào LinkedList ở vị trí



❖ `boolean list1.addAll(index, ArrayList);`

```
LinkedList<String> llistobj = new LinkedList<String>();  
ArrayList<String> arraylist= new ArrayList<String>();  
arraylist.add("String1");  
arraylist.add("String2");  
llistobj.addAll(1,arraylist);
```

Thêm đầu List



❖ void addFirst(Object)

```
public static void main(String args[]){  
    LinkedList<String> al=new LinkedList<String>();  
    al.add("Mai");  
    al.add("Nhưng");  
    al.add("Nhi");  
    al.addFirst("Lan");  
    Iterator<String> itr=al.iterator();  
    while(itr.hasNext()){  
        System.out.println(itr.next());  
    }  
}
```


Thêm cuối List



❖ void addLast(Object)

```
public static void main(String args[]){  
    LinkedList<String> al=new LinkedList<String>();  
    al.add("Mai");  
    al.add("Nhưng");  
    al.add("Nhi");  
    al.addLast("Lan");  
    Iterator<String> itr=al.iterator();  
    while(itr.hasNext()){  
        System.out.println(itr.next());  
    }  
}
```

Tạo một bản sao của List



❖ Object clone()

```
Object str= llistobj.clone();  
System.out.println(str);
```

Lấy phần tử



- ❖ Lấy tại vị trí: `Object get(int index);`
- ❖ Lấy đầu: `Object getFirst();`
- ❖ Lấy cuối: `Object getLast();`

```
public static void main(String args[]){  
    LinkedList<String> al=new LinkedList<String>();  
    al.add("Mai");  
    al.add("Nhưng");  
    al.add("Nhi");  
    String str=al.get(1);  
    System.out.println(str);  
}
```

Lấy phần tử



- ❖ Lấy tại vị trí: `Object get(int index);`
- ❖ Lấy đầu: `Object getFirst();`
- ❖ Lấy cuối: `Object getLast();`

```
public static void main(String args[]){  
    LinkedList<String> al=new LinkedList<String>();  
    al.add("Mai");  
    al.add("Nhưng");  
    al.add("Nhi");  
    String str=al.get(1);  
    System.out.println(str);  
}
```

Lấy vị trí của mục



❖ `int indexOf(Object);`

```
public static void main(String args[]){  
    LinkedList<String> al=new LinkedList<String>();  
    al.add("Mai");  
    al.add("Nhưng");  
    al.add("Nhi");  
    int index= al.indexOf("Nhi");  
    System.out.print("Vị trí lấy được"+index);  
}
```

Xóa một phần tử



- ❖ Xóa đầu: `Object remove();`
- ❖ Xóa tại vị trí: `Object remove(int index);`
- ❖ Xóa phần tử có nội dung: `Object remove(Object);`
- ❖ Xóa đầu: `removeFirst();`
- ❖ Xóa cuối: `removeLast();`

```
public static void main(String args[]){  
    LinkedList<String> al=new LinkedList<String>();  
    al.add("Mai");  
    al.add("Nhưng");  
    al.add("Nhi");  
    al.remove("Nhưng");  
}
```

Cập nhật phần tử



❖ Object set(index, Object);

```
public static void main(String args[]){  
    LinkedList<String> al=new LinkedList<String>();  
    al.add("Mai");  
    al.add("Nhưng");  
    al.add("Nhi");  
    al.set(2,"Khanh");  
}
```

Đếm số phần tử



❖ `int size();`

```
public static void main(String args[]){  
    LinkedList<String> al=new LinkedList<String>();  
    al.add("Mai");  
    al.add("Nhưng");  
    al.add("Nhi");  
    int size=al.size();  
    System.out.print("Số phần tử trong list:"+size);  
}
```


Xóa tất cả các phần tử



❖ void clear();

```
public static void main(String args[]){  
    LinkedList<String> al=new LinkedList<String>();  
    al.add("Mai");  
    al.add("Nhưng");  
    al.add("Nhi");  
    al.clear();  
}
```

Lớp Date trong Java



Để lấy thời gian trong Java ta dùng lớp Date:

```
Date d=new Date();  
System.out.print(d.toString());  
//Mon Sep 21 16:32:15 GMT+07:00 2015
```

Định dạng thời gian



- ❖ Ta dùng lớp: `SimpleDateFormat` để định dạng thời gian

```
Date d=new Date();  
SimpleDateFormat df= new SimpleDateFormat("M-d-yyyy");  
Object oj=df.format(d);  
System.out.print("Ngày tháng năm:"+oj);
```

✧ Các định dạng được tra cứu trong bảng 1

Dùng ký tự 't' để định dạng thời gian



```
Date d=new Date();  
    String strDate= String.format("Ngày tháng năm:%td-  
%<tm-%<tY",d);  
    System.out.println(strDate);  
    //Ngày tháng năm:21-09-2015
```

✧ Các định dạng được tra cứu trong bảng 2

Lấy thời gian từ lớp Date



```
Date d = new Date();  
int ngay=d.getDate();  
int thang=d.getMonth();  
int nam=d.getYear();  
nam+=1900;  
System.out.println("Ngày lay duoc:"+ngay);  
System.out.println("Thang lay duoc:"+thang);  
System.out.println("Nam lay duoc:"+nam);
```

Đổi chuỗi sang Date



```
SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");  
String dateInString = "07/06/2013";  
Date date = formatter.parse(dateInString);  
System.out.println(date);  
System.out.println(formatter.format(date));  
ngay=date.getDate();  
System.out.println("Ngày doi tu chuoi:"+ngay);
```

Bài tập 1



- Cài đặt lớp mảng 1 chiều
 - Nhập mảng 1 chiều
 - Xuất mảng một chiều
 - Tìm một phần tử trong mảng
 - Đếm theo một tiêu chí nào đó
 - Xóa 1 phần tử
 - Tìm max
 - Tìm min
 - Sắp xếp tăng

Bài tập 2



- Cài lớp Quản lý sinh viên biết rằng thông tin của một sinh viên như sau: tên, mã số sinh viên, điểm trung bình
- Lớp ngày, tháng năm: nhập vào một ngày: cho biết ngày thứ mấy trong tuần
- Cài lớp mảng 2 chiều:
 - Tổng, tích hai mảng
- Lớp số phức:
- Lớp giờ phút giây:
 - Khoảng cách giữa 2 thời gian