

Buổi 02

Học ngôn ngữ Java

(<http://docs.oracle.com/javase/tutorial/java/index.html>)

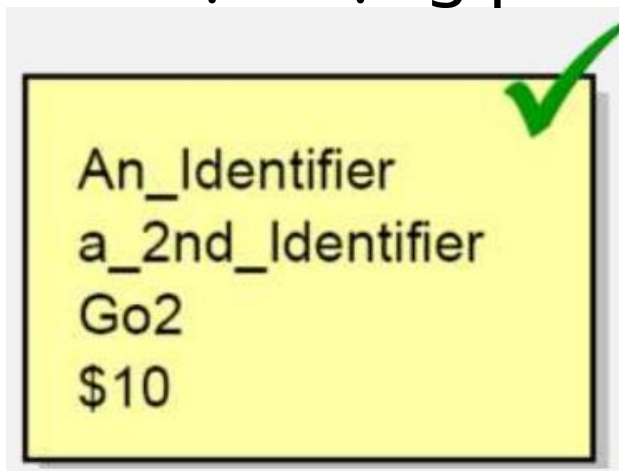
Trịnh Thị Vân Anh - PTIT

Mục tiêu

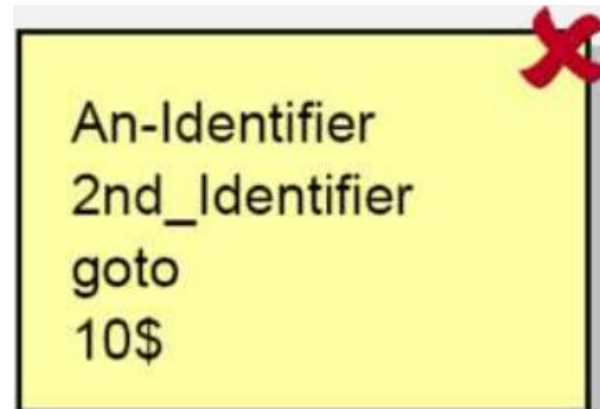
- Nghiên cứu một số nguyên tắc cơ bản của ngôn ngữ Java: Kiểu dữ liệu, biến, mảng, toán tử,...
- Các cấu trúc cơ bản: loop, if, switch ()....
- Lớp bọc
- Các biến đầu vào / đầu ra
- Lớp chuỗi
- Biểu hiện thông thường
- StringBuilder, StringBuffer, StringTokenizer

Từ khóa và số nhận dạng

- Từ khóa: Hầu hết chúng tương tự như trong ngôn ngữ C
- Các biến trong Java phân biệt chữ hoa chữ thường và có thể có dãy chữ cái và số không giới hạn. Tuy nhiên, tên biến phải bắt đầu bằng một chữ cái, ký tự gạch dưới "_" hoặc ký hiệu đô la "\$".
- Số nhận dạng phải khác với từ khóa



An_Identifier
a_2nd_Identifier
Go2
\$10



An-Identifier
2nd_Identifier
goto
10\$

Quy ước đặt tên trong java

- tên lớp: nên bắt đầu bằng chữ hoa và là một danh từ, ví dụ như Chuối, Màu, Nút, Hệ thống, Chủ đề, v.v.
- tên giao diện: nên bắt đầu bằng chữ hoa và là một tính từ, ví dụ như Runnable, Remote, ActionListener, v.v.
- tên phương thức: nên bắt đầu bằng chữ thường và là một động từ, ví dụ: actionPerformed (), main (), print (), println (), v.v.
- tên biến: nên bắt đầu bằng chữ thường, ví dụ: firstName, orderNumber, v.v.
- tên gói: phải ở dạng chữ thường, ví dụ java, lang, sql, ...
- tên hằng số: phải được viết hoa. ví dụ: RED, YELLOW, MAX_PRIORITY, v.v.

Các kiểu dữ liệu nguyên thủy - Biến

- Một *nguyên thủy* là một cái sim Ple ntrên- vật dữ liệu gõ cái đó đại diện cho một giá trị duy nhất. Của Java dữ liệu nguyên thủy loại là:

Thể loại	Byte	Tối thiểu	Tối đa
char	2	\ u0000	\ uFFFF
byte	1	- 2 ⁷	2 ⁷ - 1
ngắn	2	- 2 ¹⁵	2 ¹⁵ - 1
int	4	- 2 ³¹	2 ³¹ - 1
Dài	số 8	- 2 ⁶³	2 ⁶³ - 1
trôi nổi	4		
gấp đôi	số 8		
boolean	đúng sai		

Nhập var [=Giá trị ban đầu] ;

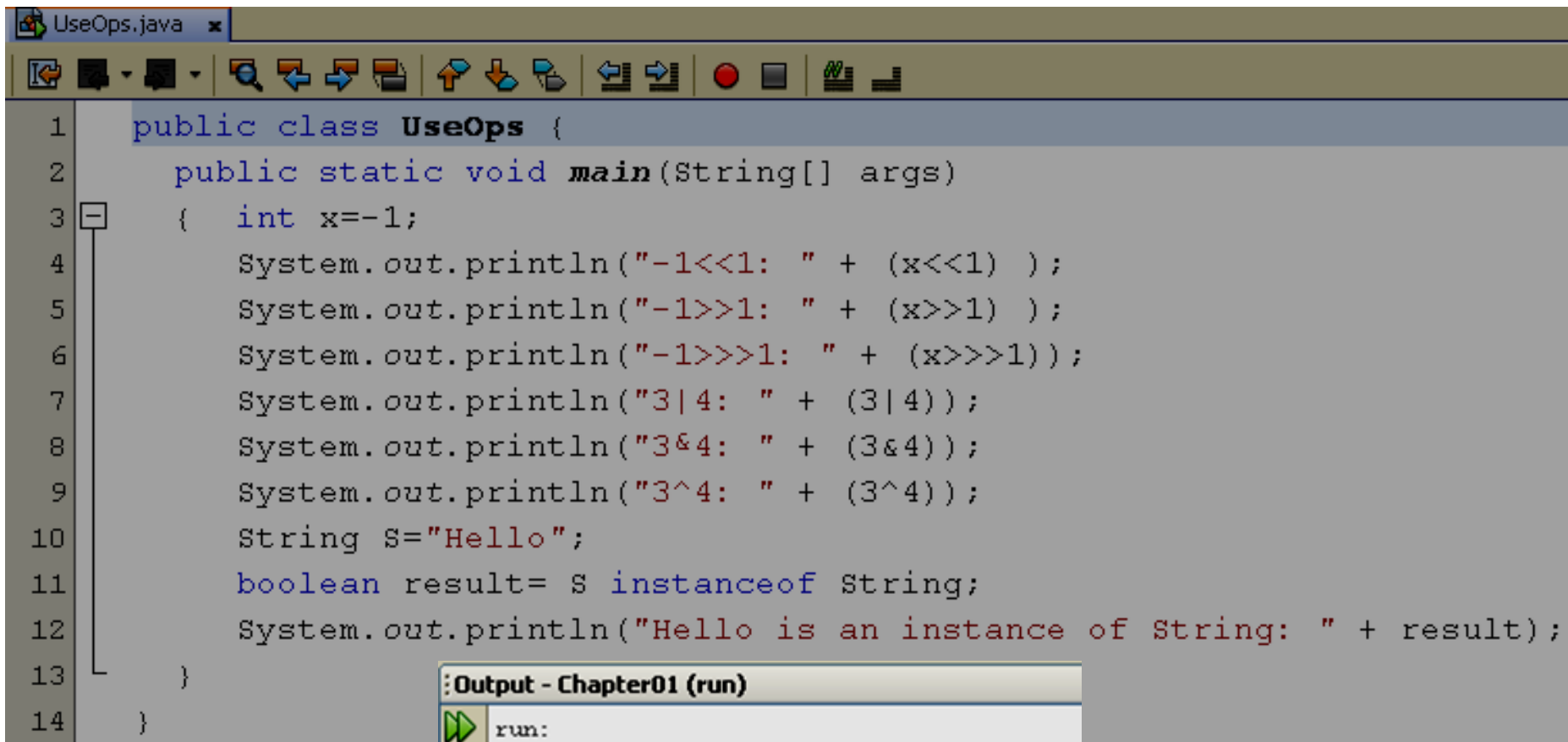
Các nhà khai thác

Danh mục (Ưu tiên giảm dần)	Các nhà khai thác
Một ngôi	+ + - - + -! ~ (loại)
Môn số học	* / % + -
Sự thay đổi	<< >> >>>
So sánh	<<=>> = instanceof ==! =
Bitwise	& ^
Ngắn mạch	&&
Có điều kiện	?:
Phân công	= op =

Chúng giống nhau với

những người trong ngôn ngữ C

Sử dụng Trình diễn Người vận hành



The image shows a screenshot of an IDE window titled 'UseOps.java'. The code is as follows:

```
1 public class UseOps {
2     public static void main(String[] args)
3     {
4         int x=-1;
5         System.out.println("-1<<1: " + (x<<1) );
6         System.out.println("-1>>1: " + (x>>1) );
7         System.out.println("-1>>>1: " + (x>>>1));
8         System.out.println("3|4: " + (3|4));
9         System.out.println("3&4: " + (3&4));
10        System.out.println("3^4: " + (3^4));
11        String S="Hello";
12        boolean result= S instanceof String;
13        System.out.println("Hello is an instance of String: " + result);
14    }
15 }
```

Below the code editor, there is an 'Output - Chapter01 (run)' window showing the following output:

```
run:
-1<<1: -2
-1>>1: -1
-1>>>1: 2147483647
3|4: 7
3&4: 0
3^4: 7
Hello is an instance of String: true
BUILD SUCCESSFUL (total time: 0 seconds)
```

Chữ viết và các biến giá trị

- Ký tự: 'a'
- Chuỗi: String S = "Xin chào";
- Chữ tích phân:
28, 0x1c, 0X1A (mặc định: int).
123l, 123L (dài)
- Dấu chấm động:
1.234 (mặc định: gấp đôi)
1,3f 1,3F
1,3E + 21
1,3 ngày 1.3D

Biến giá trị

Cây rơm

n

10

int n = 10;

Trình tự thoát

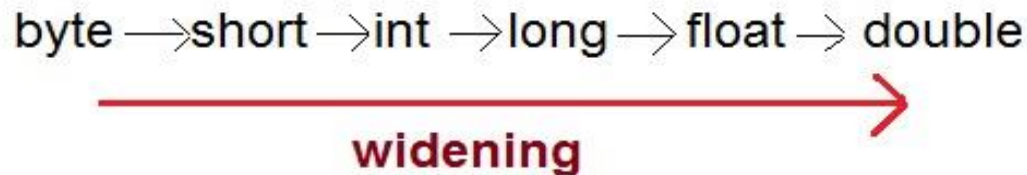
- `\t` Chèn một tab trong văn bản tại điểm này. Chèn một
- `\b` backspace trong văn bản tại điểm này.
- `\n` Chèn một dòng mới trong văn bản tại điểm này. Chèn một
- `\r` dấu xuống dòng trong văn bản tại điểm này.
- `\f` Chèn một nguồn cấp dữ liệu trong văn bản tại điểm này.
- `\'` Chèn một ký tự trích dẫn duy nhất trong văn bản tại thời điểm này.
- `\"` Chèn một ký tự dấu ngoặc kép trong văn bản tại thời điểm này.
- `\\` Chèn một ký tự gạch chéo ngược trong văn bản tại điểm này.

Biểu thức Java

- Java là một ngôn ngữ hướng biểu thức. Một biểu thức đơn giản trong Java là:
 - Một hằng số: 7, sai
 - Một ký tự - chữ được đặt trong dấu ngoặc kép: 'A', '3'
 - Một chuỗi - nghĩa đen được đặt trong dấu ngoặc kép: "foo"
 - Tên của bất kỳ biến nào được khai báo đúng: x
 - Bất kỳ hai | một trong các kiểu biểu thức đứng trước được kết hợp với một trong các toán tử nhị phân Java: `i ++`, `x + 2`, `(x + 2)`

Loại Đúc

- Việc gán giá trị của một kiểu cho một biến của kiểu khác được gọi là **Loại Đúc**. Trong Java, kiểu ép kiểu được phân loại thành hai kiểu,
- Mở rộng truyền (Ngụ ý)



- Thu hẹp truyền (Thực hiện rõ ràng)



Truyền ví dụ

- gấp đôi d = 100,04;
- dài l = (dài) d; // yêu cầu ép kiểu rõ ràng
int i = (int) l; // yêu cầu truyền kiểu rõ ràng

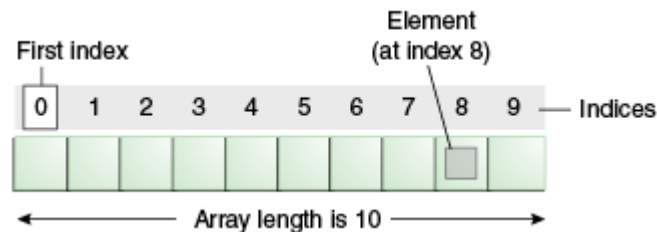
```
int a, b;  
ngắn c;  
a = b + (int) c;
```

```
int d;  
e ngắn;  
e = (ngắn) d;
```

```
gấp đôi f;  
dài g;  
f = g;  
g = f; //lỗi
```

Mảng một chiều (1)

- Một *mảng* là một đối tượng vùng chứa chứa một số lượng giá trị cố định của một kiểu duy nhất.
- Độ dài của mảng được thiết lập khi mảng được tạo.
- Mỗi mục trong một mảng được gọi là một *yếu tố* và mỗi phần tử được truy cập bằng số của nó *mục lục*.



Mảng một chiều (2)

- Khai báo một biến để tham chiếu đến một mảng `int [] anArray;`
`hoặc float anArrayOfFloats [];`
- Tạo, khởi tạo và truy cập một mảng

`anArray = new int [10];`
- Sao chép mảng
 - Sử dụng phương thức `arraycopy` từ lớp `System`.

Mảng một chiều (3)

```
int [] ar;
```

```
ar = new int [3];
```

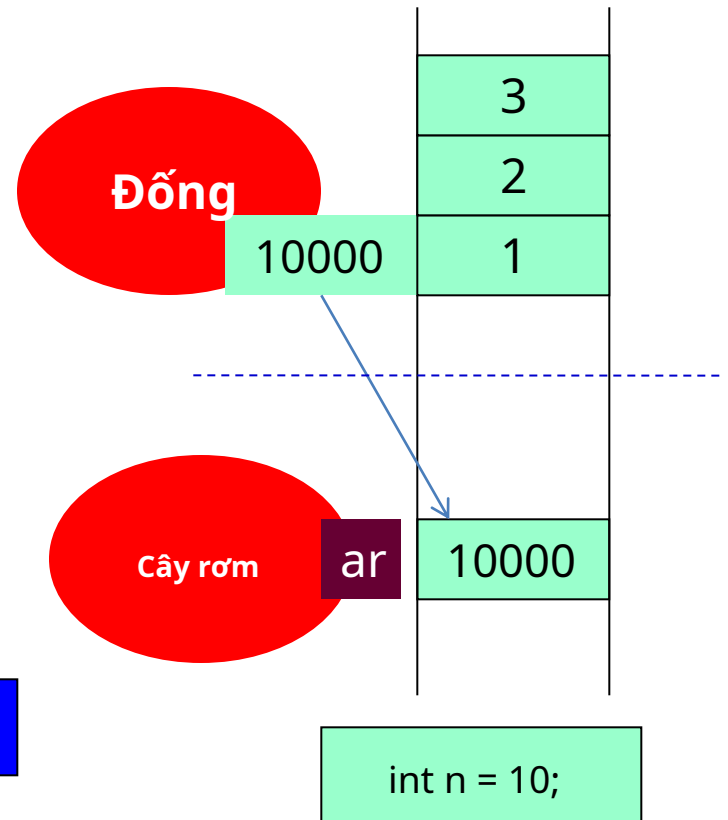
```
ar [0] = 1; ar [1] = 2; ar [2] = 3;
```

```
int a2 [];
```

```
int [] a3 = {1,2,3,4,5};
```

```
int a4 [] = {1,2,3,4,5};
```

Mảng là một biến tham chiếu



ví dụ mảng một chiều

```
giai cấp công cộng DaySo{  
    int calSum(int ...a) { // int [] a  
        int t = 0;  
        for (int x: a)  
            t += x;  
        trả lại t;  
    }  
    int calMin(int ... a) {  
        int t = a [0];  
        for (int x: a)  
            nếu (t> x)  
                t = x;  
        trả lại t;  
    }
```



```

int calMax(int ... a) {
    int t = a [0];
    for (int x: a)
        nếu (t < x)
            t = x;
    trả lại t;
}

int []loại(int ... a) {
    int t;
    for (int i = 0; i < a.length-1; i ++)
        for (int j = i + 1; j < a.length; j ++)
            if (a [i] > a [j]) {
                t = a [i];
                a [i] = a [j];
                a [j] = t;
            }
    trả lại a;
}

```

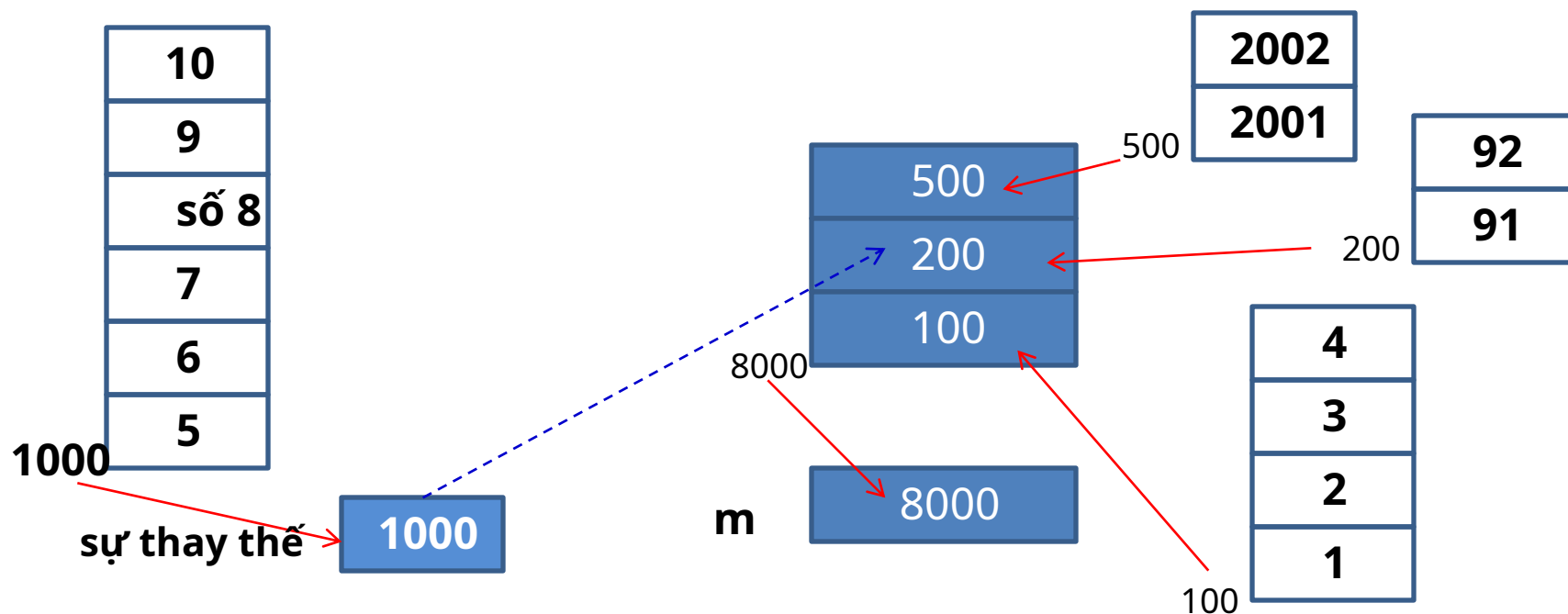
```

int []đầu vào(int n) {
    Scanner in = new Scanner (System.in); int [] b =
    new int [n];
    for (int i = 0; i <n; i ++) {
        System.out.print ("\ n thu" + i + ":");

        b [i] = in.nextInt ();
    }
    trả lại b;      }
vô hiệu(int ... a) {
    System.out.print ("\ n Day so:" +Mảng
    .toString (a));
}

```

Mảng nhiều chiều



```
int m [] [] = {{1,2,3,4}, {91,92}, {2001,2002}};
```

```
int [] thay thế = {5,6,7,8,9,10};
```

```
m [1] = thay thế;
```

m [i] [j]

```
int [] [] m; // khai báo một ma trận
int r = 10, c = 5; // số hàng, số cột m =
new int [r] [c]; // cấp phát bộ nhớ
```

- `int b [3] [4];`
- `int row = b.length;`
- `int col = b [0] .length;`

đến *cộng* các yếu tố cho một *mảng*

cộng đôi [] [] **cộng**(gấp đôi,
gấp đôi [] [] b) {

```
    int m = a.length; int n = a [0] .length;  
    double [] [] c = new double [m] [n]; for (int i  
    = 0; i <m; i ++)
```

```
        for (int j = 0; j <n; j ++)  
            c [i] [j] = a [i] [j] +  
            b [i] [j]; trả lại c;
```

```
}
```

đến *Trừ đi*

```
công đôi [] [] trừ đi(gấp đôi,  
    gấp đôi [] [] b) {  
    int m = a.length; int n = a [0] .length;  
    double [] [] c = new double [m] [n]; for (int i  
        = 0; i <m; i ++)  
  
        for (int j = 0; j <n; j ++)  
            c [i] [j] = a [i] [j] - b [i] [j]; trả lại c;  
  
}
```

Nhân lên

```
công đôi [] [] nhân(gấp đôi,  
    gấp đôi [] [] b) {  
    int m1 = a.length; int n1 = a  
    [0].length; int m2 =  
    b.length; int n2 = b  
    [0].length;  
    if (n1 != m2) ném mới RuntimeException ("Kích thước ma trận  
    không hợp lệ.");  
    double [] [] c = new double [m1] [n2]; for (int i  
    = 0; i < m1; i ++)  
        for (int j = 0; j < n2; j ++)  
            for (int k = 0; k < n1; k ++)  
                c [i] [j] + = a [i] [k] * b [k] [j]; trả lại c;  
}
```

đến *đổi chỗ* ma trận

công đôi [] []

đổi chỗ(gấp đôi) {

int m = a.length; int n = a
[0].length;

double [] [] b = mới **gấp đôi [n] [m]**; for
(int i = 0; i < m; i ++)

for (int j = 0; j < n; j ++)

b [j] [i] = a [i] [j];

trả lại b;

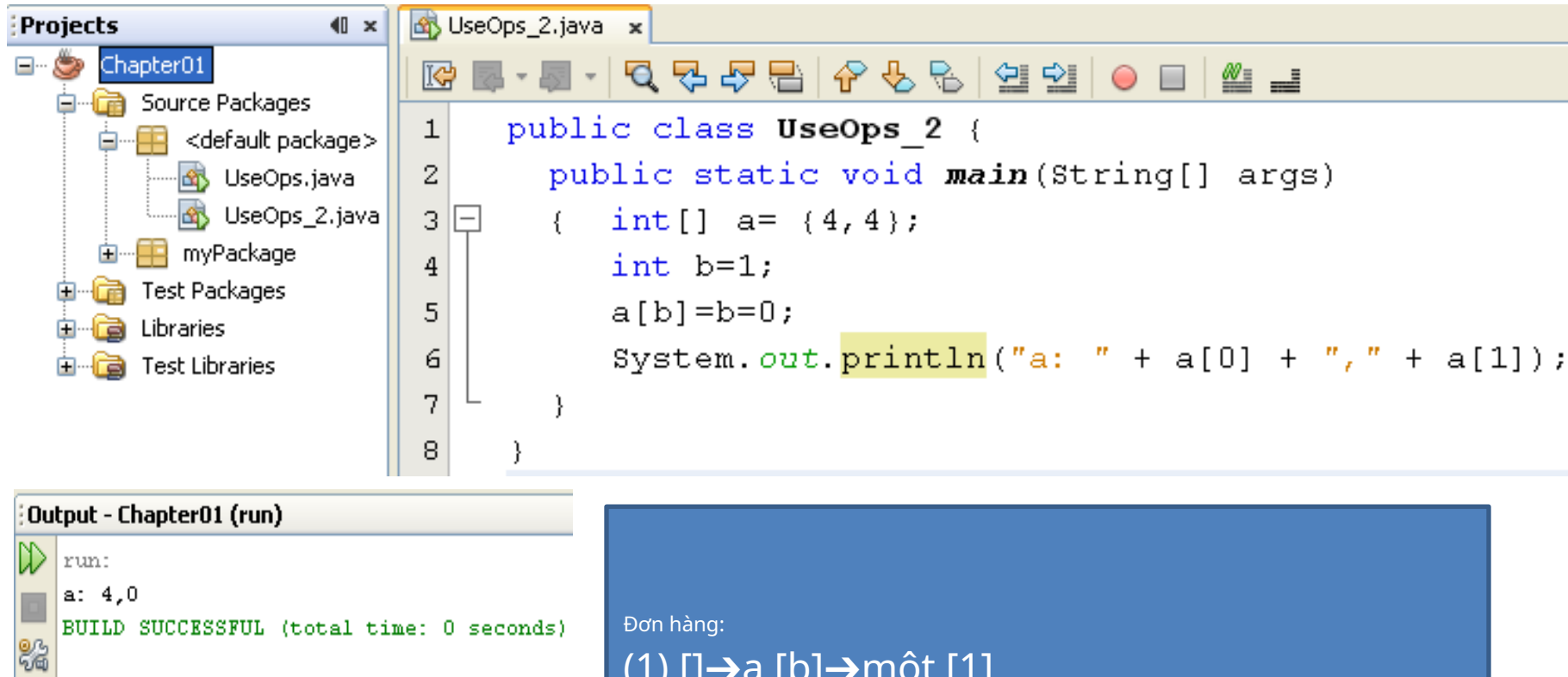
}

Đánh giá Biểu thức và ưu tiên điều hành

- Trình biên dịch thường đánh giá các biểu thức như vậy từ dấu ngoặc trong cùng đến ngoài cùng, từ trái sang phải.

```
int x = 1; int y = 2; int z = 3; int  
answer = ((8 * (y + z)) + y) * x;  
sẽ được đánh giá từng phần như sau: ((8 * (y  
+ z)) + y) * x  
((8 * 5) + y) * x  
(40 + y) * x  
42 * x  
42
```

Ưu tiên người vận hành- Thứ tự đánh giá



The screenshot displays an IDE interface. On the left, the 'Projects' panel shows a project named 'Chapter01' with a package structure including 'Source Packages', '<default package>', 'myPackage', 'Test Packages', 'Libraries', and 'Test Libraries'. The main editor window shows the file 'UseOps_2.java' with the following code:

```
1 public class UseOps_2 {  
2     public static void main(String[] args)  
3     {  
4         int[] a= {4,4};  
5         int b=1;  
6         a[b]=b=0;  
7         System.out.println("a: " + a[0] + "," + a[1]);  
8     }  
}
```

Below the editor, the 'Output - Chapter01 (run)' panel shows the execution results:

```
run:  
a: 4,0  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Đơn hàng:

(1) []→a [b]→một [1]

(2) = (từ bên phải)→b = 0→trả về 0

→ a [1] = 0

Các cấu trúc cơ bản

- Java cung cấp ba cấu trúc vòng lặp. Lấy từ C và C ++, đây là:
 - trong khi(),
 - làm ,
 - vì().

Cái *while* (Vòng

trong khi (*boolean_condition*) {

các câu lệnh);

}

Ví dụ

```
số int = 1;  
    trong khi (số <= 200) {  
        System.out.print (số + " ");  
  
        số *= 2;  
    }
```

Đầu ra :

1 2 4 8 16 32 64 128

Cái doVòng

làm {

 làm việc gì đó

 làm nhiều hơn

} trong khi (*boolean_condition*);

Ví dụ

```
// cuộn cho đến khi chúng ta nhận được một số khác 3
Random rand = new Random (); int
chết;
làm {
    die = rand.nextInt ();
} while (die == 3);
```

Các *foreach*(Vòng

```
for (start_expr;  
    test_expr; increment_expr) {  
    // mã để thực thi lặp lại}
```


Ví dụ

- `for (int index = 0; index < 10; index++) {

 System.out.println (chỉ mục); }`
- `for (int i = -3; i <= 2; i++)
 {System.out.println (i);
 }`
- `for (int i = 3; i >= 1; i--) {
 System.out.println (i);
}`

Nâng cao Vòng lặp

- Các vòng lặp for của Java đã được cải tiến trong bản phát hành 1.5 để hoạt động dễ dàng hơn với các mảng và bộ sưu tập.
- Cú pháp:
for (kiểu tên_biến: mảng)

```
int sumOfLengths (Chuỗi [] chuỗi) {  
    int totalLength = 0;  
    cho (Chuỗi s: chuỗi)  
        totalLength += s.length ();  
    trả về totalLength;  
}
```

Ví dụ

Ví dụ về lớp công khai {

```
public static void main (String args []) {  
    int []con số= {10, 20, 30, 40, 50}; for (int x: number)  
    {  
        System.out.print (x); //số [i] System.out.print  
        (",");  
    }  
    System.out.print ("\n");  
    Sợi dây []những cái tên= {"James "," Larry "," Tom ","  
    Lacy "};  
    for (Tên chuỗi: tên) {  
        System.out.print (tên          );  
        System.out.print (",");  
    }  
}
```

- Đầu ra:
10,20,30,40,50,
James, Larry, Tom, Lacy,

Tuyên bố lựa chọn

- **Các *nếu khác* Xây dựng**
- **Các *công tắc()* Xây dựng**

chuyển đổi (x) {

trường hợp 1:

```
System.out.println ("Có 1");  
nghỉ;
```

trường hợp 2:

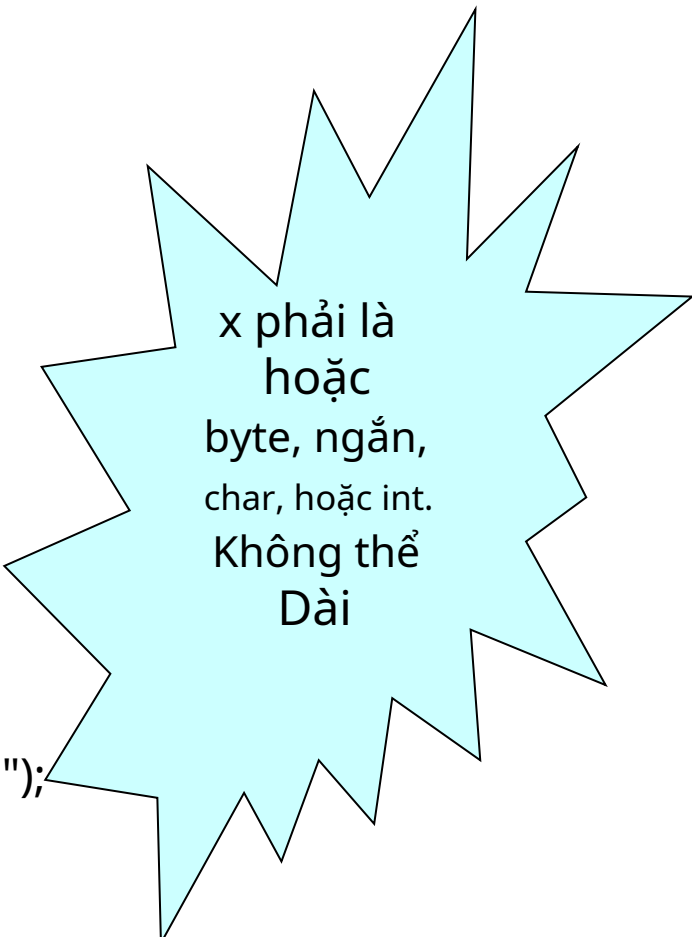
trường hợp 3:

```
System.out.println ("Có 2 hoặc 3");  
nghỉ;
```

mặc định:

```
System.out.println ("Không phải là 1, 2 hoặc 3");  
nghỉ;
```

}



x phải là
hoặc
byte, ngắn,
char, hoặc int.
Không thể
Dài

Nếu ví dụ

```
lớp CheckNumber {  
    public static void main (String args []) {  
  
        int num = 10;  
        nếu (num% 2 == 0)  
            System.out.println (num + "là chẵn"); khác  
  
            System.out.println (num + "là số lẻ");  
        }  
    }  
}
```

```
lớp công khai SwitchDemo {  
    public static void main (String [] args) {  
        int m = 8;  
        Chuỗi tháng;  
        chuyển đổi (m) {  
            trường hợp 1: month = "January"; break;  
            trường hợp 2: month = "tháng 2"; nghỉ; trường  
            hợp 3: month = "March"; nghỉ;  
            trường hợp 4: month = "April"; nghỉ; trường hợp  
            5: month = "May"; nghỉ; trường hợp 6: tháng =  
            "tháng 6"; nghỉ; trường hợp 7: month = "July";  
            nghỉ; trường hợp 8: month = "August"; nghỉ;  
            trường hợp 9: tháng = "tháng 9"; nghỉ;
```

```
trường hợp 10: tháng = "tháng 10"; trường hợp  
phá vỡ 11: tháng = "tháng 11"; nghỉ; trường hợp  
12: tháng = "tháng 12"; nghỉ; mặc định: tháng =  
"không phải một tháng";
```

```
    nghỉ;
```

```
}
```

```
System.out.println (tháng); }
```

```
}
```


Các tiếp tục

Các câu lệnh trong vòng lặp (cho, trong khi, làm)

vì(.;.;.)

{

// xử lý phần 1

nếu (điều kiện)

tiếp tục;

// xử lý phần 2

}

vòng lặp chính: vì(.;.;.) {

vì(.;.;.)

{

// xử lý phần 1 if

(boolean_exp)

tiếp tục *vòng lặp chính;*

// xử lý phần 2

}

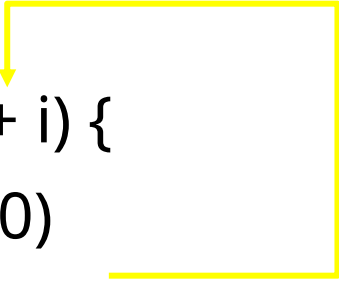
}

JAD Mè sion 3 - Kiểm soát dòng chảy,

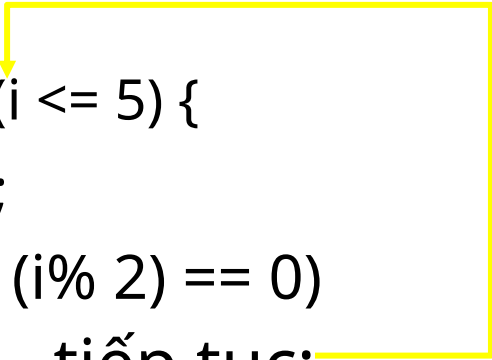
Khẳng định và
Xử lý ngoại lệ

tiếp tục

```
for (i = 0; i <= 5; ++ i) {  
    nếu (i% 2 == 0)  
        tiếp tục;  
    System.out.println ("Đây là lần lặp" + i + "");  
}
```



```
i = 0;  
trong khi (i <= 5) {  
    + + i;  
    nếu (i% 2) == 0)  
        tiếp tục;  
    System.out.println ("Đây là lần lặp lẻ -" + i);  
}
```



Các *ngữ* Các câu lệnh trong vòng lặp (for, while, làm)

```
vì( .;.;.)
```

```
{
```

```
    // xử lý phần 1
```

```
    nếu (điều kiện)
```

```
        ngữ;
```

```
    // xử lý phần 2
```

```
}
```

Nghỉ

```
int i = 1;  
trong khi (đúng) {  
    nếu (i == 3)  
        nghỉ;  
    System.out.println ("Đây là lần lặp" + i + "");  
    ++ i;  
}
```

Thực đơn

trong khi (đúng) {

System.out.print ("\ n 1.Chọn 1 ");

System.out.print (" \ n 2.Chọn 2 ");

System.out.print ("\ n 3. Chọn 3 ");

System.out.print (" \ n 0. Thoát ") ;

System.out.print ("\ n Chọn 1,2,3 hoặc 0:");

```
Máy quétin = new  
Scanner (System.in);  
int c = in.nextInt ();  
System.out.print ("\ n");
```

```
chuyển đổi (c) {  
    case 1: // cần làm để chọn 1  
        nghỉ;  
    case 2: // việc cần làm để chọn 2  
        nghỉ;  
    case 3: // việc cần làm để chọn 3  
        nghỉ;  
    case 0: System.out.print ("\ n Tạm biệt !!!");  
        System.exit (0);  
        nghỉ;  
    mặc định:  
    System.out.print ("\ n Chọn 1,2,3 hoặc 0");  
}  
}
```

Ví dụ về cấu trúc logic cơ bản

- Chúng giống với những gì trong câu lệnh C

```
2 package com;
3 import java.lang.*;
4 public class Chao {
5     public static void main(String args[]) {
6         System.out.println("Hello");
7         int a[] = { 1,2,3,4,5};
8         for (int i=0;i<a.length;i++) System.out.print(a[i] + ",");
9         System.out.println();
10        for (int x : a) System.out.print(x + ",");
11        System.out.println();
12        for (int x : a) x+=10;
13        for (int i=0;i<a.length;i++) System.out.print(a[i] + ",");
14        System.out.println();
15    }
16 }
```

Vòng lặp for nâng cao

Read only

Một 1 2 3 4 5

x

1

Output - P1 (run)

```
run:
Hello
1,2,3,4,5,
1,2,3,4,5,
1,2,3,4,5,
```

Loại chuỗi

- Chuỗi đại diện cho một chuỗi không hoặc nhiều ký tự Unicode.
 - Tên chuỗi = "Steve";
 - Chuỗi s = "";
 - Chuỗi s = null;
- Nối chuỗi.
 - Chuỗi x = "foo" + "bar" + "!";
- Java là một ngôn ngữ phân biệt chữ hoa chữ thường.

Nhập chuyển đổi và truyền rõ ràng

```
Casting_Convert_1.java * x
public class Casting_Convert_1 {
    public static void main (String[] args)
    {
        short x, y = 256;
        byte m, n = 6;
        x = n ; // Systematic Conversion
        n = y; // narrow conversion
        n = (byte) y; // narrow casting, possible loss of precision
        System.out.println(n);
    }
}
```

```
Casting_Convert_1.java * x
public class Casting_Convert_1 {
    public static void main (String[] args)
    {
        short x, y = 256;
        byte m, n = 6;
        x = n ; // Systematic Conversion
        n = (byte) y; // narrow casting, possible loss of p
        System.out.println(n);
    }
}
```

Output - Chapter04 (run)

```
run:
0
BUILD SUCCESSFUL (total time: 0 seconds)
```

* Mở rộng chuyển đổi: OK

- Chuyển đổi thu hẹp: Không được phép. Chúng ta phải sử dụng đúc rõ ràng.
- Một boolean không thể được chuyển đổi thành bất kỳ loại nào khác.
- Một kiểu không boolean có thể được chuyển đổi thành một kiểu nonboolean khác.

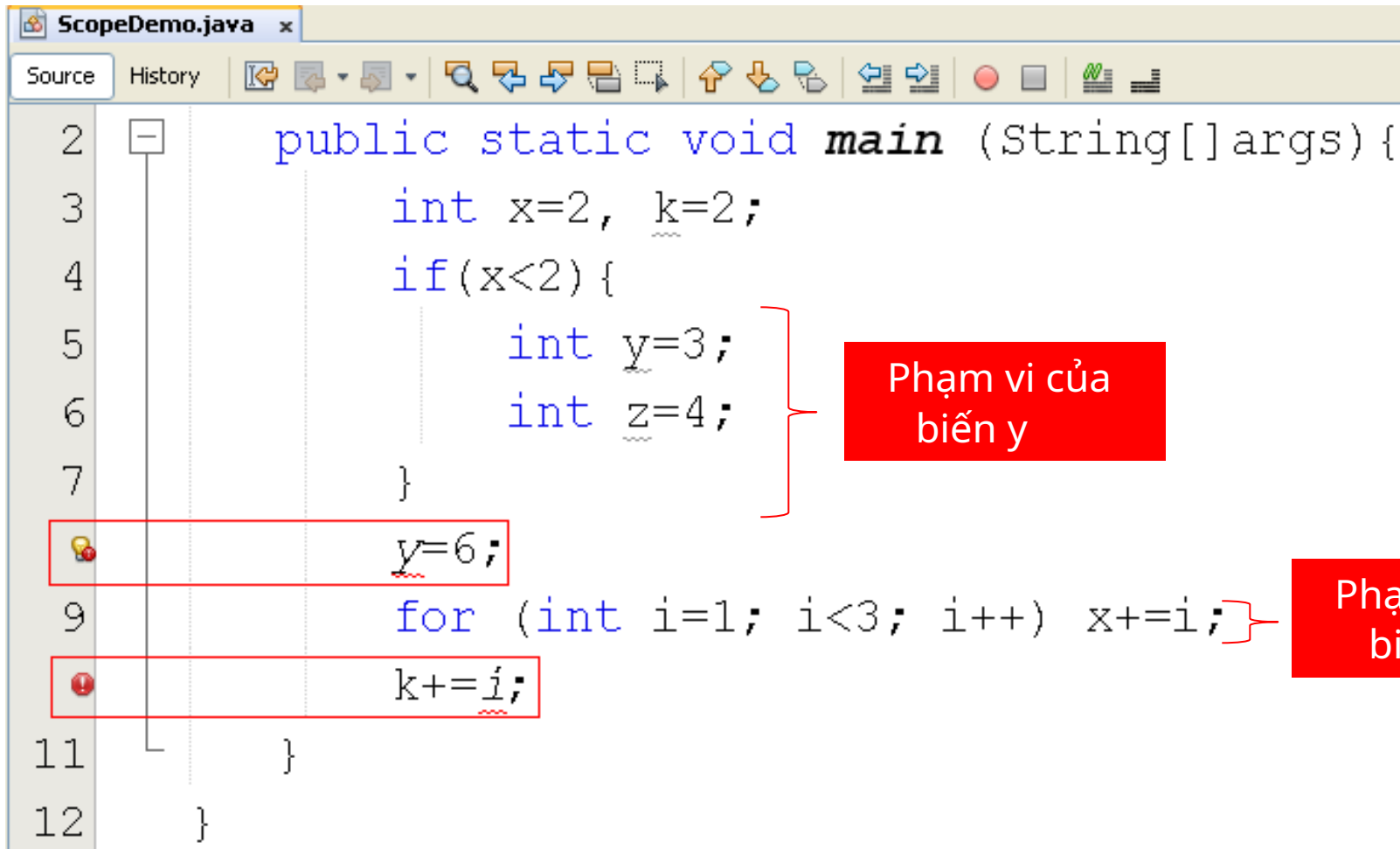
0000 0001

0000 0000

y

n

Phạm vi của một biến



The screenshot shows a Java IDE window titled "ScopeDemo.java". The code is as follows:

```
2 public static void main (String[] args) {  
3     int x=2, k=2;  
4     if(x<2) {  
5         int y=3;  
6         int z=4;  
7     }  
8     y=6;  
9     for (int i=1; i<3; i++) x+=i;  
10    k+=i;  
11 }  
12 }
```

Annotations and scope analysis:

- A red bracket on the right side of lines 5 and 6 points to a red box containing the text "Phạm vi của biến y" (Scope of variable y).
- A red bracket on the right side of lines 8 and 9 points to a red box containing the text "Phạm vi của biến i" (Scope of variable i).
- A red box highlights line 8, containing the code `y=6;`. A yellow lightbulb icon is visible in the left margin next to this line.
- A red box highlights line 10, containing the code `k+=i;`. A red error icon is visible in the left margin next to this line.

Lớp ngẫu nhiên

- nhập khẩu `java.util.Random;`
- Ngẫu nhiên `rd = new Random ();`
- `int a = rd.nextInt (n); // 0- (n-1)`
- `int t = rd.nextInt (b-a + 1)`
`+ a; // (từ a đến b)`
- `Float = rd.nextFloat (); // 0-1`

Lop toan

- java.lang
- Math.PI
- Math.abs (-20);
- double c = Math.ceil (7,342); // 8.0
- double f = Math.floor (7.343); // 7.0
- double p = Math.pow (2, 3); // 8.0
- double s = Math.sin (Math.PI / 2); // 1
- double a = Math.sqrt (9); // 3
-

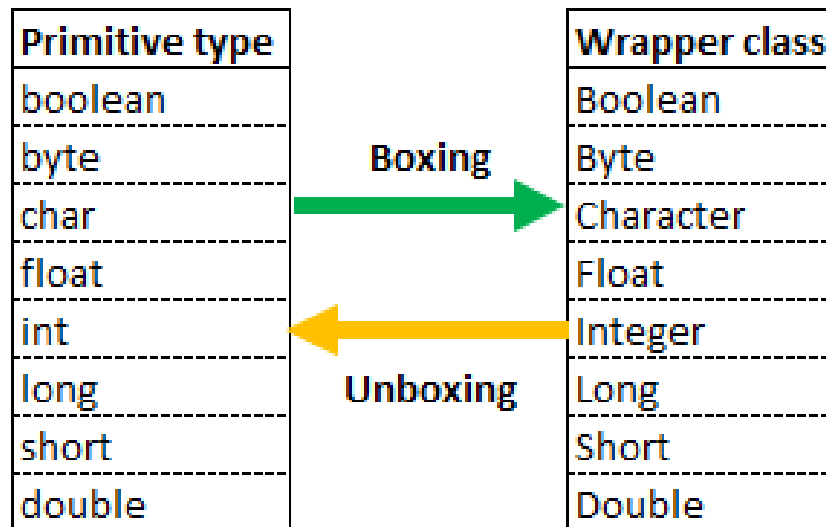
Lớp bọc

● java.lang

Primitive Type	Size	Minimum Value	Maximum Value	Wrapper Type
char	16-bit	Unicode 0	Unicode $2^{16}-1$	Character
byte	8-bit	-128	+127	Byte
short	16-bit	-2^{15} (-32,768)	$+2^{15}-1$ (32,767)	Short
int	32-bit	-2^{31} (-2,147,483,648)	$+2^{31}-1$ (2,147,483,647)	Integer
long	64-bit	-2^{63} (-9,223,372,036,854,775,808)	$+2^{63}-1$ (9,223,372,036,854,775,807)	Long
float	32-bit	32-bit IEEE 754 floating-point numbers		Float
double	64-bit	64-bit IEEE 754 floating-point numbers		Double
boolean	1-bit	true or false		Boolean
void	-----	-----	-----	Void

Tự động đóng hộp và mở hộp (1)

- java 5.0 giới thiệu hai chức năng rất đơn giản nhưng thuận tiện mở bao bọc đồ vật và gói (lại) nguyên thủy.
- Việc chuyển đổi một giá trị nguyên thủy thành một đối tượng của lớp trình bao bọc tương ứng được gọi là autoboxing.
- Chuyển đổi một đối tượng của một loại wrapper thành giá trị nguyên thủy tương ứng của nó được gọi là unboxing.



Đầu ra ???

- Số nguyên `y = 567; //` tạo một trình bao bọc
- `Integer y = new Integer (567);`
- Số nguyên `x = y; //` chỉ định một tham chiếu thứ hai
 `//` var tới trình bao bọc
`System.out.println (y == x);`
`y ++; //` mở ra, sử dụng, "rewrap"
`System.out.println (x + "" + y);`
`System.out.println (y == x);`

Autoboxing và Unboxing... (2)

● Mẫu autoboxing và unboxing

Số nguyên được bọc `Int = 25; // quyền anh hoặc quyền anh tự động`

Diện tích gấp đôi (bán kính gấp đôi) {
 trả về `Math.PI * radius * radius; // quyền anh`
}

Số nguyên `wi = 234;`
`int times9 = wi * 9; // mở hộp`

Trả về các kiểu nguyên thủy

- Để trả về các kiểu nguyên thủy: using method **typeValue ()**

// tạo một đối tượng wrapper mới

Integer i2 = new Integer (42); // byte

byte b = i2.byteValue (); // ngắn

short s = i2.shortValue (); // gấp đôi

double d = i2.doubleValue ();

để chuyển đổi từ Chuỗi sang kiểu nguyên thủy

- Sử dụng các phương thức (tĩnh) của các lớp trình bao bọc
static <type>**phân tích cú pháp**Loại (Chuỗi s)

- Ví dụ

Chuỗi s = "123"; // int

int i = Integer.parse**Int**(S); // ngắn

short j = Short.parse**Ngắn**(S); Sợi dây
txt = "13,5";

Gấp đôi x = Double.parse**Gấp đôi**(txt);

Lớp Java.lang.Integer

1. static int MAX_VALUE: $2^{31}-1$.
2. static int MIN_VALUE: -2^{31} .
3. Số nguyên (giá trị int), Số nguyên (Chuỗi s)
4. java.lang.Integer.compare ()
5. Integer obj1 = new Integer ("25"); Integer obj2 = new Integer ("10");
6. int **sự kiểm tra lại** = obj1.compareTo (obj2);
7. int **sự kiểm tra lại** = Integer.compare (obj1, obj2);
- số 8. byte byteValue (), doubleValue (), float floatValue ()...
9. static int parseInt (String s), String toString (), static String toString (int i)
10. boolean bằng (Đối tượng đối tượng)

Lớp `Java.lang.Double`

- Gấp đôi giá trị),
`Double (Chuỗi S)`
- `byte byteValue ()`, `doubleValue ()`,
`float`
`floatValue ()`, `int intValue ()`, ..
- `static int so sánh (double d1, double d2)`,
`int so sánhTo (Double anotherDouble)`
- `boolean isNaN()`, phân tích cú pháp kép
`staticDouble (String s)`, `String toString ()`,
`static` Sợi dây
`toString (đôi d)`

java.lang.Character Class

- static char toUpperCase (char ch)
- static char toLowerCase (char ch)
- static String toString (char c)
- Chuỗi toString ()
- static boolean isWhitespace (char ch)
- tĩnh boolean isUpperCase (char ch)
- tĩnh boolean isLowerCase (char ch)
- tĩnh boolean isLetter (char ch)

Dây

- Java sử dụng **Sợi dây**, **StringBuffer**, **StringBuilder** và **StringTokenizer** các lớp để đóng gói các chuỗi ký tự (16-bit Unicode).

java.lang.Vật

java.lang.Sợi dây (thực hiện java.lang.Trình tự Char ,
java.lang.Có thể so sánh được <T>, java.io.Serializable)

java.lang.StringBuffer (thực hiện java.lang.Trình tự Char ,
java.io.Serializable)

java.lang.StringBuilder (dụng cụ
java.lang.Trình tự Char , java.io.Serializable)

Lớp java.lang.String

- Lớp String đại diện cho các chuỗi ký tự. Tất cả các ký tự chuỗi trong chương trình Java, chẳng hạn như "abc", được triển khai dưới dạng các thể hiện của lớp này.

```
String a = "Chuỗi A"; Chuỗi b  
= "";
```

- Xây dựng một chuỗi

```
Chuỗi c = MớiSợi dây();
```

```
Chuỗi d = MớiString ("Khác  
Sợi dây");
```

```
Chuỗi e = String.valueOf (1.23); Chuỗi f =  
null;
```

Toán tử chuỗi

- Nhà điều hành +

Chuỗi a = "This" + "là một" +
"Sợi dây";

// a = "Đây là một chuỗi"

- Chuỗi với print ()

```
System.out.println ("answer =" + 1 + 2 + 3);
```

```
System.out.println ("answer =" + (1 + 2 + 3));
```

```
System.out.println ("Số:" +1.45);
```


Phương thức chuỗi

- so sánh (), concat (), bằng (), split (), length (), Replace (), so sánhTo (), chuỗi con (),...
- Ví dụ
 - String name = "Ly Lao Lo";
 - name.toLowerCase (); // "ly lao lo"
 - name.toUpperCase (); // "LY LAO LO"
 - " Ly Lao Lo ".trim (); //" Ly Lao Lo "
 - "Lý Lão Lộ ".indexOf ('L'); // 1
 - "Lý Lão Lộ ".indexOf (" La ");
 - "Lý Lão Lộ ".length (); // 9
 - "Lý Lão Lộ ".charAt (5); // 'o'
 - "Ly Lao Lo ".substring (5); //" o Lo "
 - "Lý Lão Lộ ".substring (2,5); //" La "

- **int so với**(Chuỗi khác Chuỗi)
- **int so sánh ToIgnoreCase**(Chuỗi str)
- Sợi dây **kết hợp**(Chuỗi str)
- **boolean kết thúc**(Hậu tố chuỗi) String str =
"www.tutorialspoint.com"; Chuỗi endstr1 = ".com";

Chuỗi endstr2 = ".org";

boolean retval1 = str.endsWith (endstr1); **boolean**
retval2 = str.endsWith (endstr2);

- **boolean bằng**(Đối tượng anObject)
- **boolean equalsIgnoreCase**(Sợi dây chuỗi khác)
- **int Chỉ số**(Chuỗi str)
- **int lastIndexof**(Chuỗi str)

- chuỗi công khai []**tách ra**(Chuỗi regex, giới hạn int)
- chuỗi công khai []**tách ra**(Chuỗi regex)
- **1.** Chuỗi s1 = "*Ví dụ phương pháp Tách ra một dòng*
vào trong *Từ ngữ*";
Sợi dây[] st1 = s1.split ("\\ s +");
cho (Chuỗi w1: st1) {
System.out.println (w1); }
- **2.**String s2 = "Bạn là ai? Bạn có đẹp không? một số đã gọi bạn
là dễ thương trong khi những người khác chỉ gọi *bạn xinh*
đẹp. ";
String [] st2 = s2.split ("[\\ . \\ ? \\ !]"); for (Chuỗi w2:
st2) {
System.out.println (w2); }

- công cộng Sợi dây **thay thế tất cả**(Sợi dây
regex, Sợi dây thay thế)
- công cộng Sợi dây **thay thế**(Sợi dây
regex, Sợi dây thay thế)

1. "một số đã gọi bạn là dễ thương trong khi những người khác chỉ gọi *bạn xinh đẹp*".

`ReplaceAll ("\\ s +", "");`

2. Chuỗi s3 = "Hà Nội nổi tiếng với nhiều sông, hồ, núi bên cạnh và xung quanh. ";

`System.out.println (s3.`

`thay thế tất cả("\\,\\S*", "", "));`

3. "tên tôi là khanh tên tôi là *java*".

`ReplaceAll ("là", "là");`

Dữ liệu đầu vào / đầu ra

- **nhập `java.util.Scanner`;**
- **Đầu vào máy quét = Máy quét mới (`System.in`);**
- công cộng Sợi dây `tiếp theo()`
- công cộng Sợi dây `hàng tiếp theo()`
- công cộng `byte` `nextByte ()`
- công cộng ngắn `nextShort ()`
- công cộng `int` `nextInt ()`
- công cộng Dài `nextLong ()`
- công cộng trôi nổi `nextFloat ()`
- công cộng gấp đôi `nextDouble ()`

Sự cố với `.nextLine ()` trong Java

- `System.out.println ("Nhập giá trị số");`
- `tùy chọn int;`
- `option = input.nextInt (); System.out.println ("Nhập chuỗi đầu tiên"); Chuỗi string1 = input.nextLine ();`
`System.out.println ("Nhập chuỗi thứ 2"); Chuỗi string2 = input.nextLine ();`
- Giải pháp:
 - `int option = input.nextInt ();`
 - `input.nextLine ();` // Tiêu thụ dòng mới còn thừa
 - `Chuỗi str1 = input.nextLine ();`
 - `int option =`
`Số nguyên.phân tích cú phápInt (input.nextLine ());`

Ví dụ

```
InputOutputDemo.java x
1  /* Write a program that will accept an array of integers then
2     print out entered value and the sum of values
3  */
4  import java.util.Scanner;
5  public class InputOutputDemo {
6      public static void main (String args[])
7      {
8          int a[]; // array of integers
9          int n; // number of elements of the array
10         int i; // variable for traversing the array
11         Scanner sc= new Scanner(System.in); // object for the keyboard
12         System.out.print("Enter number of elements: ");
13         n = Integer.parseInt(sc.nextLine());
14         a = new int[n]; // mem. allocating for elements of the array
15         for (i=0;i<n;i++)
16         {
17             System.out.print("Enter the " + (i+1) + "/" + n + " element: ");
18             a[i]=Integer.parseInt(sc.nextLine());
19         }
20         System.out.print("Entered values: ");
21         for (i=0;i<n;i++) System.out.format("%5d", a[i]);
22         int S=0;
23         for (int x: a) S+=x;
24         System.out.println("\nSum of values: " + S);
25     }
26 }
```

`n = sc.nextInt ();`

Tham khảo tài liệu Java:
java.lang.String lớp,

- các định dạng phương pháp,
- chuỗi định dạng

để biết thêm chi tiết

```
Output - Chapter01 (run) #2
run:
Enter number of elements: 5
Enter the 1/5 element: 1
Enter the 2/5 element: 4
Enter the 3/5 element: 2
Enter the 4/5 element: 0
Enter the 5/5 element: 7
Entered values:      1      4      2      0      7
Sum of values: 14
BUILD SUCCESSFUL (total time: 11 seconds)
```

Biểu thức chính quy trong Java

- Các `java.util.regex` gói chủ yếu bao gồm ba lớp: [Mẫu](#) , [Matcher](#) , và [PatternSyntaxException](#) .
 - MỘT **Mẫu** đối tượng là một đại diện đã biên dịch của một biểu thức chính quy. Lớp `Pattern` không cung cấp hàm tạo công khai. Để tạo một mẫu, trước tiên bạn phải gọi một trong các phương thức biên dịch tĩnh công khai của nó, sau đó sẽ trả về một đối tượng `Mẫu`.
 - MỘT **Matcher** đối tượng là công cụ diễn giải mẫu và thực hiện các hoạt động so khớp với một chuỗi đầu vào.
 - MỘT **PatternSyntaxException** đối tượng là một ngoại lệ không được kiểm tra chỉ ra lỗi cú pháp trong mẫu biểu thức chính quy.


```

lớp công khai MatchPhoneNumber {
    boolean tĩnh công cộng isPhoneValid(Điện thoại chuỗi)
    {
        boolean retval = false; Chuỗi
        regex =
        "\\^\\ (? \\d {3} \\)? -? \\s * \\d {3} \\s * -? \\d {4} $"; sự
        kiểm tra lại=phone.matches (regex); nếu (kiểm tra lại)

        System.out.println ("TRẬN ĐẤU" + điện thoại + "\\ n"); hỏi
        đáp lại; }
    khoảng trống tĩnh công cộng chủ yếu(Chuỗi args []) {
        isPhoneValid ("(234) - 765 -8765"); isPhoneValid
        ("999-585-4009"); isPhoneValid ("1-585-4009"); }

}

```

```
lớp công khai MatchRollNumber {  
    boolean tĩnh công cộng isRollNumber(Id chuỗi) {  
  
        boolean retval = false; Chuỗi  
        regex =  
            "^[Bb]{1} \\ d {2} [A-Za-z]{4} \\ d {3} $"; //  
            "^[NXnx]{1} \\ d {3} [ A-Za-z]{4} $ " sự kiểm tra  
            lại=id.matches (regex); nếu (kiểm tra lại)
```

```
        System.out.println ("MATCH" + id + "\\ n"); hồi đáp  
        lại; }  
}
```

```
khoảng trống tĩnh công cộng chủ yếu(Chuỗi args []) {  
    isRollNumber ("B13DCCN765");  
    isRollNumber ("B13DCCN8584");  
    isRollNumber ("b12dcat321"); }}
```

StringBuffer, StringBuilder Các lớp học

- Của Java **StringBuffer** và **StringBuilder** các lớp đại diện cho các chuỗi có thể được sửa đổi động.
 - StringBuffer là **threadsafe**.
 - StringBuilder (được giới thiệu trong 5.0) không phải là threadsafe.
- Hầu như các phương thức của chúng giống với các phương thức trong lớp String.
- Các lớp này không sử dụng nhóm chuỗi, do đó chúng ta không thể viết StringBuffer t = "ABC";
- Chúng ta không thể sử dụng toán tử + cho các đối tượng của chúng.

Chủ đề: Đơn vị mã (phương thức) đang chạy

Chương trình đa luồng: Một chương trình có một số luồng chạy đồng thời. Nếu 2 luồng truy cập dữ liệu chung, giá trị của chúng không phải là không dự đoán được. Vì vậy, trong lập trình đa luồng, JVM hỗ trợ một cơ chế trong đó việc truy cập vào các tài nguyên chung phải được thực hiện theo trình tự dựa trên các phương thức được đồng bộ hóa.

Lớp ThreadSafe: Một lớp học với **ThreadSafe** interface. Một ngôn ngữ

StringBuilder

lớp cuối cùng công khai **StringBuilder** kéo dài Vật dụng cụ Serializable , Trình tự Char

- Lớp StringBuilder được giới thiệu trong phiên bản 5.0. Nó gần giống với StringBuffer.
- Sự khác biệt chính: trình tạo chuỗi là **không phải threadsafe**.
- Nếu bạn muốn có nhiều chủ đề **truy cập đồng thời** thành một thể biến đổi chuỗi, sử dụng một bộ đệm chuỗi.
- Nếu chuỗi có thể thay đổi của bạn chỉ được truy cập bởi một chuỗi duy nhất, ở đó là một lợi thế khi sử dụng trình tạo chuỗi, thường sẽ thực thi nhanh hơn bộ đệm chuỗi.

StringBuilder -Các hàm tạo lớp

Constructor & Description

StringBuilder()

This constructs a string builder with no characters in it and an initial capacity of 16 characters.

StringBuilder(int capacity)

This constructs a string builder with no characters in it and an initial capacity specified by the capacity argument.

StringBuilder(String str)

This constructs a string builder initialized to the contents of the specified string.

StringBuilder -Phương thức lớp

StringBuilder append (String str) Phương thức này nối chuỗi được chỉ định vào chuỗi ký tự này.

[StringBuilder append \(StringBuffer sb\)](#) Phương thức này nối StringBuffer được chỉ định vào chuỗi này.

char charAt (int index) Phương thức này trả về giá trị char trong chuỗi này tại chỉ mục được chỉ định.

[Xóa StringBuilder \(int start, int end\)](#) Phương thức này loại bỏ các ký tự trong một chuỗi con của chuỗi này.

[StringBuilder deleteCharAt \(int index\)](#) Phương thức này loại bỏ ký tự tại vị trí được chỉ định trong chuỗi này.

[int indexOf \(Chuỗi str\)](#) Phương thức này trả về chỉ mục trong chuỗi này của lần xuất hiện đầu tiên của chuỗi con được chỉ định.

[int indexOf \(Chuỗi str, int fromIndex\)](#) Phương thức này trả về chỉ mục trong chuỗi này của lần xuất hiện đầu tiên của chuỗi con được chỉ định, bắt đầu từ chỉ mục được chỉ định.

int length () Phương thức này trả về độ dài (số ký tự).

Thay thế StringBuilder (int start, int end, String str) Phương thức này thay thế các ký tự trong một chuỗi con của chuỗi này bằng các ký tự trong Chuỗi được chỉ định.

[Đảo ngược StringBuilder \(\)](#) Phương pháp này làm cho chuỗi ký tự này được thay thế bằng sự đảo ngược của chuỗi ký tự.

[Chuỗi con \(int start\)](#) Phương thức này trả về một Chuỗi mới chứa một dãy con của các ký tự hiện có trong chuỗi ký tự này.

[Chuỗi con \(int start, int end\)](#) Phương thức này trả về một Chuỗi mới chứa một dãy con của các ký tự hiện có trong chuỗi này.

[Chuỗi toString \(\)](#) Phương thức này trả về một chuỗi đại diện cho dữ liệu trong chuỗi này.

[StringBuilder insert \(int offset, String str\)](#) Phương thức này chèn chuỗi vào chuỗi ký tự này.

Ví dụ

1. `StringBuilder s = mới
StringBuilder ("D14-"); s.append
("CN"); // D14- CN`
2. `s.insert (4, "CQ"); // D14-CQ CN`
3. `s.delete (3,5); // D14 CN`
4. `s.reverse ();`

Văn bản bình thường

```
lớp công khai WrapperDemo {  
    public static String normalText (Dòng chuỗi)  
    {  
        Chuỗi ra = "";  
        line = line.toLowerCase ();  
        line = line.replaceAll ("\\ s +", ""); line =  
        line.replaceAll ("\\. ", "\\ "); line = line.replaceAll ("\\  
\\ ", "\\ "); line = line.replaceAll ("\\,", "\\,"); line =  
        line.replaceAll ("\\,", "\\,"); line = line.replaceAll ("\\ s  
+", ""); line = line.trim ();
```

```
ra = dòng;  
boolean isCap = true; ký tự  
c;  
StringBuilder strb = new StringBuilder (""); for (int i = 0; i  
<out.length () - 1; i ++) {  
    c = out.charAt (i);  
    if (c == '.') {  
        isCap = true;  
    }  
    if (isCap && Character.isAlphabetic (c)) {  
        c = Character.toUpperCase (c); isCap =  
        false;  
    }  
    strb.append (c);  
}
```

```

out = strb.toString ();
if (out.charAt (out.length () - 1) != '.') {
    out = out + ".";
}
trở ra;
}
public static void main (String [] args) {
    String line = "Chúng tôi      là      cả hai      trẻ tuổi ,
        khi nào      Lần đầu tiên tôi nhìn thấy bạn.
        tôi      nhắm mắt lại và hồi tưởng
bắt đầu ";
System.out.println (normalText (dòng));
}

```

Có thể đảo ngược

```
đảo ngược boolean công khai (int n) {  
    StringBuilder sn = new  
    StringBuilder (Integer.toString (n));  
    trở về  
    sn.toString (). equals (sn.reverse (). to String ());  
  
}
```

Các *StringBuffer* - *threadsafe*

lớp cuối cùng công khai **StringBuffer** kéo dài Vật
dụng cụ Serializable , Trình tự Char

```
public class StringBufferDemo {  
    public static void main(String args[]){  
        StringBuffer sBuf= new StringBuffer ("01234567");  
        System.out.println(sBuf);  
        sBuf.append("ABC");  
        System.out.println(sBuf);  
        sBuf.insert(2, "FAT PERSON");  
        System.out.println(sBuf);  
        sBuf.reverse();  
        System.out.println(sBuf);  
    }  
}
```

run:

01234567

01234567ABC

01FAT PERSON234567ABC

CEA765432NOSREP TAF10

Người xây dựng

Constructor & Description

StringBuffer()

This constructs a string buffer with no characters in it and an initial capacity of 16 characters.

StringBuffer(int capacity)

This constructs a string buffer with no characters in it and the specified initial capacity.

StringBuffer(String str)

This constructs a string buffer initialized to the contents of the specified string.

- công cộng **đồng bộ** StringBuffer append (String S)
- công cộng **đồng bộ** Chèn StringBuffer (int offset, String s)
- công cộng **đồng bộ** Thay thế StringBuffer (int startIndex, int endIndex, String str)
-

Ví dụ: StringBuffer vs StringBuilder

```
lớp công khai ConcatTest {  
    public static void main (String [] args) {  
        long startTime = System.currentTimeMillis (); StringBuffer  
        sb1 = MớiStringBuffer ("Java"); for (int i = 0; i <10000; i ++) {  
  
            sb1.nối thêm("D14CN-PTIT");} System.out.println ("Tổng số  
            thời gian(StringBuffer): "  
            + (System.currentTimeMillis () - startTime) + "ms");  
  
            startTime = System.currentTimeMillis (); StringBuildersb2 =  
            MớiStringBuilder ("Java"); for (int i = 0; i <10000; i ++) {  
  
                sb2.nối thêm("D14CN-PTIT");} System.out.println ("Tổng số  
                thời gian (StringBuilder): "  
                + (System.currentTimeMillis () - startTime) + "ms");  
  
            }  
        }  
    }  
}
```

Tổng số*thời gian*(StringBuffer):15ms
Tổng số*thời gian*(StringBuilder): 0ms

Lớp StringTokenizer

- Các **java.util.StringTokenizer** lớp cho phép bạn chia một chuỗi thành các mã thông báo. Đó là cách đơn giản để phá vỡ chuỗi.
- Người xây dựng:
 - StringTokenizer (String str): tạo StringTokenizer với chuỗi và métích được chỉ định.
 - StringTokenizer (String str, String delim): tạo StringTokenizer với chuỗi và métích được chỉ định.

Các phương thức của lớp StringTokenizer

Public method	Description
<code>boolean hasMoreTokens()</code>	checks if there is more tokens available.
<code>String nextToken()</code>	returns the next token from the StringTokenizer object.
<code>String nextToken(String delim)</code>	returns the next token based on the delimiter.
<code>boolean hasMoreElements()</code>	same as <code>hasMoreTokens()</code> method.
<code>Object nextElement()</code>	same as <code>nextToken()</code> but its return type is Object.
<code>int countTokens()</code>	returns the total number of tokens.

Ví dụ

```
nhập java.util.StringTokenizer; lớp công  
khai Đơn giản {  
public static void main (Chuỗi  
    args []) {  
    StringTokenizer st = new StringTokenizer  
    ("Tôi làm việc tại HN, tôi là một giảng viên.  
    Tôi yêu HN.", "\\.");  
    while (st.hasMoreTokens ())  
    {System.out.println (st.nextToken ()); }}
```

- Theo mặc định, StringTokenizer ngắt chuỗi

String str = "Tôi là chuỗi mẫu và sẽ là
được mã hóa trên không gian";

```
StringTokenizer dt = new StringTokenizer (str); trong khi  
(dt.hasMoreTokens ()) {  
    System.out.println (dt.nextToken ());}
```

- Nhiều dấu phân cách

```
String s = "Tôi là ai? Lan là bạn của tôi. Tôi yêu Lan! Lan  
yêu Anh ấy như thế nào? tất nhiên là tôi biết!";  
StringTokenizer mt = MớiStringTokenizer (s, ".?!");
```

```
while (mt.hasMoreTokens ()) {  
    System.out.println (mt.nextToken ());  
}
```

Tóm lược

- Các tính năng truyền thống của ngôn ngữ, bao gồm: biến, mảng, kiểu dữ liệu, toán tử và luồng điều khiển.
- Lớp Wrapper, 4 lớp Chuỗi
- Biểu thức chính quy để xác thực dữ liệu