



Lập Trình Java (Cơ bản) IT20050

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Chương 1. Cơ bản về ngôn ngữ lập trình Java

- 1.1. Giới thiệu về ngôn ngữ lập trình Java
- 1.2. Các cú pháp cơ bản của Java
- 1.3. Mảng trong Java
- 1.4. Tham số
- 1.5. Chuỗi trong Java
- 1.6. Điều khiển luồng

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



1.1. Giới thiệu về ngôn ngữ lập trình Java



Giới thiệu

- ❑ Java được phát triển đầu tiên bởi James Gosling tại Sun Microsystems (giờ là công ty con của Oracle Corporation) và được công bố năm 1995.
- ❑ Java là ngôn ngữ lập trình máy tính có tính hướng đối tượng, dựa trên các lớp, thường được sử dụng cho các hệ thống có tính độc lập cao. Nó được sử dụng để hướng tới các lập trình viên viết ứng dụng "write one, run everywhere" (viết một lần, chạy mọi nơi) nghĩa là đoạn code Java sau khi được biên dịch có thể chạy được trên tất cả các nền tảng hỗ trợ Java mà không cần phải được biên dịch lại. Các ứng dụng Java sau khi đã được biên dịch thành bytecode có thể chạy trên bất kỳ máy ảo Java nào (Java virtual machine)



Đặc điểm và tính năng của Java

- Đơn giản
- Hướng đối tượng
- Độc lập nền tảng
- Portable (có thể mang Java Bytecode tới bất cứ nền tảng nào)
- Hiệu suất cao
- Đa luồng (Multi-thread)
- Distributed
- ...



Đặc điểm và tính năng của Java

Đơn giản

- ❑ Cú pháp của nó dựa trên C++.
- ❑ Gỡ bỏ nhiều đặc điểm gây bối rối và hiếm khi được sử dụng chẳng hạn như các con trỏ tường minh, nạp chồng toán tử, ...
- ❑ Không cần xóa các đối tượng mà không được tham chiếu, bởi vì những thứ đó được Bộ dọn rác tự động (GarbageCollection) thực hiện trong Java.

Hướng đối tượng

- ❑ Hướng đối tượng nghĩa là chúng ta tổ chức phần mềm dưới dạng một sự kết hợp của nhiều loại đối tượng khác nhau mà kết hợp chặt chẽ cả về dữ liệu lẫn hành vi của chúng.



Đặc điểm và tính năng của Java

Độc lập nền tảng

- ❑ Một Platform là môi trường phần cứng hoặc phần mềm trong đó một chương trình chạy. Có hai loại Platform: một loại dựa trên phần mềm (software-based) và một loại dựa trên phần cứng (hardware-based). Java cung cấp software-based platform. Java Platform khác với nhiều nền tảng khác ở chỗ nó chạy ở trên các nền tảng hardware-based khác.
- ❑ Java code có thể chạy trên nhiều nền tảng như Windows, Linux, Sun Solaris, Mac/OS, ... Java code được biên dịch bởi Bộ biên dịch Compiler và được chuyển đổi thành Bytecode. Bytecode này là một code độc lập nền tảng bởi vì nó có thể chạy trên nhiều nền tảng khác nhau.
- ❑ Viết một lần, Chạy khắp nơi (Write Once and Run Anywhere).



Đặc điểm và tính năng của Java

Hiệu suất cao

- ❑ Với việc sử dụng Just-In-Time compilers, Java giúp nâng cao hiệu năng, giúp việc debug được dễ dàng cũng như nhanh chóng phát hiện lỗi.

Phân tán (Distributed)

- ❑ Có thể tạo các ứng dụng phân tán trong Java. Có thể truy cập các file bằng việc gọi các phương thức từ bất cứ thiết bị nào trên internet.

Đa luồng (Multi-thread)

- ❑ Một Thread là giống như một chương trình riêng rẽ, thực thi một cách đồng thời. Chúng ta có thể viết các chương trình Java mà xử lý nhiều tác vụ cùng một lúc bằng việc định nghĩa nhiều Thread. Lợi thế chính của Multi-thread là nó chia sẻ cùng bộ nhớ. Các Thread là quan trọng cho Multi-media, Web App, ...



Ví dụ đầu tiên In một dòng văn bản

```
// This is a simple program called First.java

class First {
    public static void main (String [] args) {
        System.out.println ("My first program in Java ");
    }
}
```



Phân tích một chương trình Java

- ❑ Ký hiệu // tạo dòng chú thích.
- ❑ Dòng class First mô tả lớp mới có tên **First**.
- ❑ public static void main (String [] args)
 - ✓ **Bất kỳ một chương trình Java phải có dòng này**
 - ✓ "args" có thể thay đổi bởi một tên hay định danh bất kỳ
 - ✓ Đây là phương thức chính của chương trình và nó được thực hiện đầu tiên.
- ❑ System.out.println ("My first program in java");
 - ✓ In dòng *My first program in java* lên màn hình.



Summary

- ❑ Java là một ngôn ngữ lập trình OOP, được phát triển bởi Sun Microsystems
- ❑ Chủ yếu sử dụng cho các ứng dụng web doanh nghiệp
- ❑ Write Once, Run Anywhere



Eclipse: công cụ IDE để lập trình Java

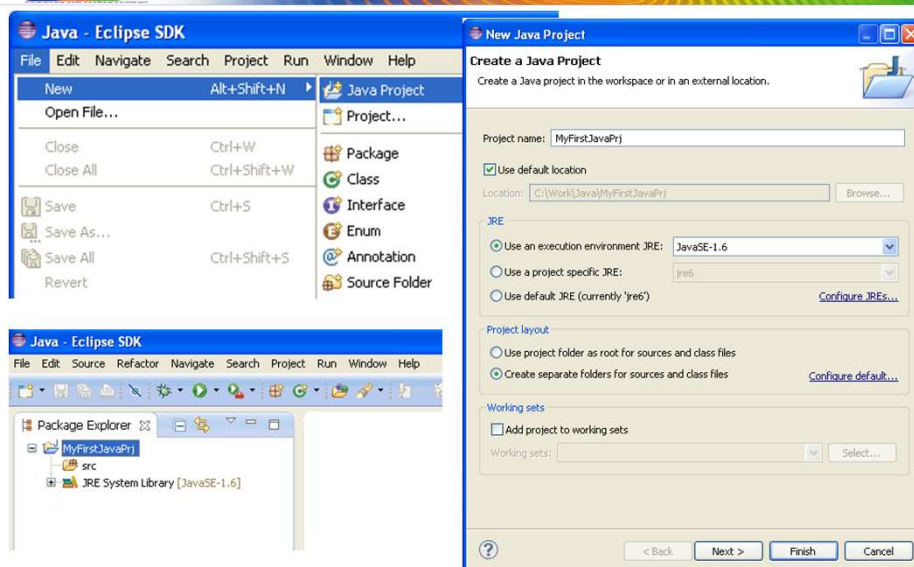



Nội dung

- ❑ Khởi tạo một dự án (Project) mới
- ❑ Chèn mã Java vào các tập tin Java mới
- ❑ Chạy các dự án Java
- ❑ Xem kết quả

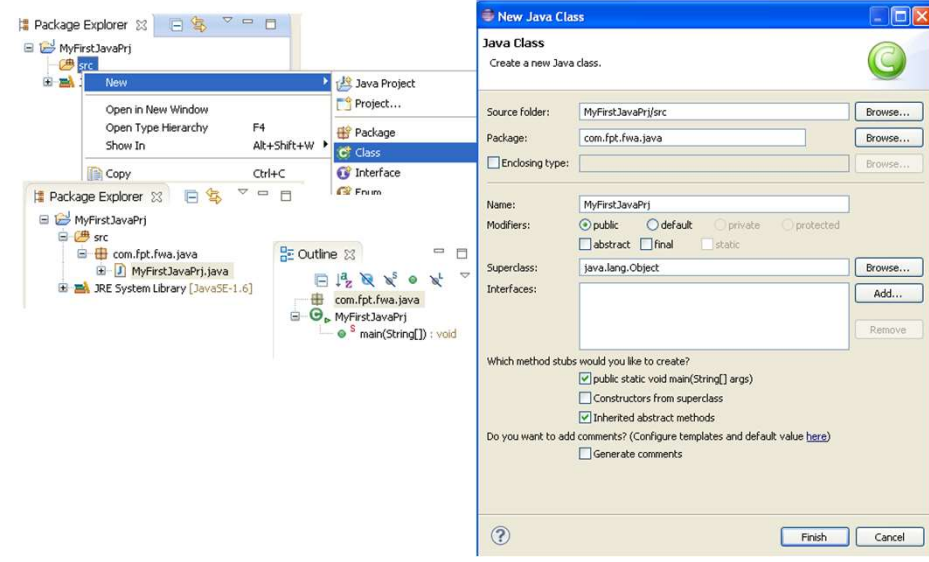


Tạo một dự án (Project) mới






Tạo một lớp (Class) mới

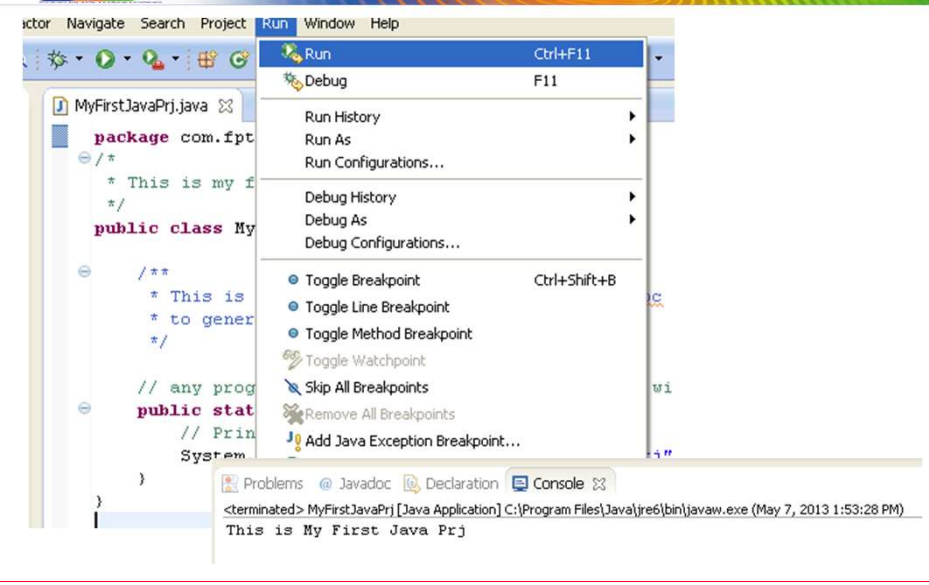


Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Running/Debugging



Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



1.2. Các cú pháp cơ bản của Java

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Dòng chú thích

```
/*  
 * Multi line  
 */  
  
// Single line
```

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Các kiểu tên, định danh trong Java

Trong Java, tên được *phân biệt chữ hoa chữ thường*.

Nó bao gồm chữ cái, chữ số, dấu \$, dấu _ (gạch dưới)

Một số dạng tên:

- ☞ Tên lớp: CustomerInfo
- ☞ Tên biến, hàm: basicAnnualSalary
- ☞ Tên hằng: MAXIMUM_NUM



Cách đặt tên trong Java

- ❑ Tên phải có nghĩa, không trùng với từ khóa hay tên khác trong cùng một phạm vi.
- ❑ Không nên viết tắt, ngoại trừ một số tên biến tạm: a, i, j
- ❑ Phân biệt: lớp TransferAction và lớp DoTransferAction
- ❑ Tên lớp bắt đầu bằng một danh từ, sử dụng các từ và tránh viết tắt
- ❑ Tên biến bắt đầu bằng một danh từ: numberOfFiles
- ❑ Tên biến không nên bắt đầu bằng '_' hoặc '\$' mặc dù được phép.
- ❑ Phân biệt số ít và số nhiều
- ❑ Tên phương thức (Method) bắt đầu bằng động từ: countNumberOfFiles()
- ❑ Tránh trộn lẫn các ngôn ngữ trong đặt tên.



Cách đặt tên trong Java

- ❑ **Tên gói (package):** bằng chữ thường toàn bộ.
Ví dụ: mypackage
- ❑ **Tên biến và phương thức:** bắt đầu bằng ký tự thường, nếu tên có nhiều từ thì ghép các từ lại, và viết hoa chữ cái đầu tiên.
Ví dụ: number, numberOne, numberTwo, getTotalRows,...
- ❑ **Tên lớp, giao diện:** viết hoa ký tự đầu tiên, nếu tên lớp hoặc giao diện có nhiều từ thì ta cũng nối các từ lại, và viết hoa chữ đầu tiên.
Ví dụ: Students, StudentManager.
- ❑ **Tên hằng:** bao giờ cũng viết hoa, nếu hằng có nhiều từ thì phân cách các từ đó bằng dấu gạch dưới.
Ví dụ: COLOR_RED, CORLOR_BLUE,...

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Một số từ khóa trong Java

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Chuẩn đầu ra trong Java

- ❑ System.out là lớp chuẩn in đầu ra trong Java
- ❑ System.err là lớp chuẩn in lỗi ra trong Java

```
class Output
{
    public static void main( String args[] )
    {
        // Standard out
        System.out.print( "Prints, but no newline" );
        System.out.println( "Prints, adds platforms newline at end" );
        double test = 4.6;
        System.out.println( test );
        System.out.println( "You can use " + "the plus operator on "
            + test + " String mixed with numbers" );
        System.out.println( 5 + "\t" + 7 );
        System.out.println( "trouble" + 5 + 7 ); //Problem here
        System.out.println( "ok" + (5 + 7) );
        System.out.flush(); // flushes output buffer
        System.err.println( "Standard error output" );
    }
}
```



Chuẩn đầu vào trong Java

- ❑ System.in là chuẩn đầu vào trong Java

Chương trình sau thực hiện đọc các ký tự từ bàn phím và in chúng ra màn hình.

```
// Echo.java
class Echo {
    public static void main (String[] args) throws IOException {
        int ch;
        System.out.println("Enter some text: ");
        while ((ch = System.in.read()) != '\n') {
            System.out.print((char)ch);
        }
    }
}
```



Các ký tự điều khiển

- Ngôn ngữ Java hỗ trợ một số dãy thoát đặc biệt cho hàng chuỗi và hằng ký tự như sau:

Escape Sequence	Description
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\r	Insert a carriage return in the text at this point.
\f	Insert a formfeed in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Các kiểu dữ liệu cơ bản

- **byte**: có giá trị từ -128 đến 127
- **short**: có giá trị từ -32,768 đến 32,767
- **int**: có giá trị từ -2,147,483,648 đến 2,147,483,647
- **long**: có giá trị từ 9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Các kiểu dữ liệu cơ bản (tiếp)

- ❑ **float**: có giá trị từ $1.4E^{-45}$ đến $3.4028235E^{38}$
- ❑ **double**: có giá trị từ $4.9E^{-324}$ đến $1.7976931348623157E^{308}$
- ❑ **boolean**: có giá trị **true** và **false**. Giá trị mặc định là **false**
- ❑ **char**: Dùng để lưu dữ liệu kiểu kí tự hoặc số nguyên không âm có kích thước 2 byte (16 bit). Có giá trị từ u0000 (0) đến ufffe (65535). Giá trị mặc định là 0.



Các toán tử (phép toán)

- ❑ **Toán tử Gán**
 - = toán tử gán đơn giản
- ❑ **Các toán tử toán học**
 - + cộng
 - trừ
 - * nhân
 - / chia
 - % lấy dư
- ❑ **Các toán tử 1 ngôi**
 - + cộng một ngôi
 - trừ một ngôi
 - ++ tăng 1
 - giảm 1
 - ! đảo ngược giá trị logic



Các toán tử (phép toán)

□ Các toán tử so sánh

==	so sánh bằng
!=	so sánh không bằng
>	lớn hơn
>=	lớn hơn hoặc bằng
<	nhỏ hơn
<=	nhỏ hơn hoặc bằng

□ Các toán tử điều kiện

&&	AND
	OR
?:	viết tắt của lệnh if_then_else



Các toán tử (phép toán)

□ Các toán tử trên bit

~	Unary bitwise complement
<<	Signed left shift
>>	Signed right shift
>>>	Unsigned right shift
&	Bitwise AND
^	Bitwise exclusive OR
	Bitwise inclusive OR



Phép GÁN trong Java

- ❑ Trong Java, phép gán “=” là sao chép con trỏ chứ không phải giá trị.
- ❑ Trong Java, chúng ta còn có các toán tử gán như sau:
 - “+=”: đầu tiên là cộng sau đó là gán.
 - “-=”: đầu tiên là trừ sau đó là gán.
 - “*=”: đầu tiên là nhân sau đó là gán.
 - “/=”: đầu tiên là chia sau đó là gán.
- ❑ Toán tử gán “=” được sử dụng để khởi tạo biến và gán giá trị mới cho biến đó.
 - Không sử dụng toán tử gán để gán giá trị **boolean** cho một biến có kiểu dữ liệu **char**, **byte**, **short**, **int**, **long**, **float**, **double** và ngược lại.
 - Không gán một biến mà kiểu dữ liệu của nó có khoảng giá trị lớn hơn sang một biến có kiểu giá trị có khoảng giá trị nhỏ hơn.



Chuyển đổi kiểu

- ❑ Trong chuyển đổi kiểu, một kiểu dữ liệu sẽ được chuyển đổi đến một kiểu dữ liệu khác.

Ví dụ:

```
float c = 34.89675f;
int b = (int)c + 10;
```




Biến (Variable)

- ❑ Có 3 thành phần trong khai báo biến:

- ✓ Tên kiểu dữ liệu
- ✓ Tên biến/định danh
- ✓ Giá trị khởi tạo

- ❑ Cú pháp khai báo

`datatype identifier1 [= initial_value1][, identifier2 [= initial_value2]...];`



Phạm vi và thời gian tồn tại của biến

- ❑ Các biến có thể được khai báo bên trong một khối.
- ❑ Các khối bắt đầu bởi “{” và kết thúc bởi “}”.
- ❑ Một khối định nghĩa một phạm vi.
- ❑ Một phạm vi mới được tạo ra mỗi khi một khối mới được tạo ra.
- ❑ Nó cũng xác định thời gian tồn tại của một đối tượng.



Ví dụ

```
class ScopeVar {
    public static void main(String[] args) {
        int num = 10;
        if ( num == 10) {
            // num is available in inner scope
            int num1 = num * num;
            System.out.println("Value of num and num1 are "
                               + num + " " + num1);
        }
        //num1 = 10;  ERROR ! num1 is not known
        System.out.println("Value of num is " + num);
    }
}
```



Summary

- ❑ Cú pháp của Java bắt nguồn từ C++
- ❑ Tất cả các code được viết trong một lớp, mọi thứ đều là một đối tượng.
- ❑ Các từ khóa
- ❑ Các kiểu dữ liệu cơ bản
- ❑ Các toán tử
- ❑ Biến và phạm vi



1.3. Mảng trong Java

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Mảng trong Java

- Mảng
 - ✓ Là một cấu trúc dữ liệu
 - ✓ Tập các phần tử cùng kiểu dữ liệu
 - ✓ Kích thước được khai báo 1 lần

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Cấu trúc của Mảng

Name of array (note that all elements of this array have the same name, c)	c[0]	-45
	c[1]	6
	c[2]	0
	c[3]	72
	c[4]	1543
Value of each element	c[5]	-89
	c[6]	0
	c[7]	62
	c[8]	-3
	c[9]	1
Index (or subscript) of the element in array c, begin from 0	c[10]	6453
	c[11]	78

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Chỉ số của Mảng

- ❑ Còn được gọi là chỉ số dưới
- ❑ Đặt trong cặp dấu []
- ❑ Bắt đầu từ 0
- ❑ Phải ≥ 0 và $<$ độ dài mảng

a = 5;

b = 6;

c[a + b] += 2;

Cộng 2 cho c[11]

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Xem xét một Mảng

- Ta xem xét đối với mảng `c`

- ✓ `c` là tên của mảng

- ✓ `c.length` lấy độ dài của mảng

- ✓ `c` có 12 phần tử (`c[0]`, `c[1]`, ... `c[11]`)

- Giá trị của phần tử `c[0]` là `-45`

<code>c[0]</code>	-45
<code>c[1]</code>	6
<code>c[2]</code>	0
<code>c[3]</code>	72
<code>c[4]</code>	1543
<code>c[5]</code>	-89
<code>c[6]</code>	0
<code>c[7]</code>	62
<code>c[8]</code>	-3
<code>c[9]</code>	1
<code>c[10]</code>	6453
<code>c[11]</code>	78



Khai báo Mảng

- Có 3 cách để khai báo một mảng:

```
datatype[] identifier;
```

```
datatype[] identifier = new datatype[size];
```

```
datatype[] identifier = {value1,value2,...valueN};
```

- Có thể khai báo như sau:

```
datatype identifier[]; // không khuyến khích
```



Mảng nhiều chiều

❑ Mảng nhiều chiều

✓ Là bảng gồm các hàng và các cột

▪ Mảng 2 chiều

▪ Khai báo mảng 2 chiều `b[2][2]`

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

– 1 và 2 khởi tạo cho `b[0][0]` và `b[0][1]`

– 3 và 4 khởi tạo cho `b[1][0]` và `b[1][1]`

▪ Mảng 3 hàng 4 cột

```
int[][] b;  
b = new int[3][4];
```



Cấu trúc của mảng 2 chiều

	Column 0	Column 1	Column 2	Column 3
Row 0	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][3]</code>
Row 1	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][3]</code>
Row 2	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][3]</code>

Diagram illustrating the structure of a 2D array. The array is represented as a table with rows and columns. The first column is labeled 'Column 0', 'Column 1', 'Column 2', and 'Column 3'. The first row is labeled 'Row 0', 'Row 1', and 'Row 2'. The elements are represented as `a[row][column]`. Arrows point from the labels to the corresponding elements in the table:

- Column index: Points to the column part of the index (e.g., 0, 1, 2, 3).
- Row index: Points to the row part of the index (e.g., 0, 1, 2).
- Array name: Points to the variable name 'a'.



Ví dụ

```
public class MultidimensionArrayDemo {
    public static void main(String[] args){
        String[][] names = {{"Mr. ", "Mrs. ", "Ms. "},
                             {"Smith", "Jones"}};
        System.out.println(names[0][0] + names[1][0]);
        System.out.println(names[0][2] + names[1][1]);
    }
}
```

Kết quả:

```
Mr. Smith
Ms. Jones
```



Summary

- ❑ Một mảng là một đối tượng chứa cố định một số phần tử có cùng chung một kiểu.
- ❑ Độ dài của mảng được thiết lập khi khởi tạo.
- ❑ Sau khi khởi tạo, độ dài của mảng là cố định.



1.4. Tham số

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Các tham số

- ❑ Các tham số là các biến được khai báo trong định nghĩa phương thức.
- ❑ Các tham số được phân loại như là “các biến” chứ không phải “các thuộc tính”.
- ❑ Còn được gọi là đối số.

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Ví dụ

- ❑ Truyền tham trị
- ❑ Truyền tham chiếu



Summary

- ❑ Các tham số là các biến được khai báo trong định nghĩa phương thức.
- ❑ Kiểu cơ sở được truyền bởi giá trị.
- ❑ Đối tượng được truyền bởi tham chiếu.



1.5. Chuỗi trong Java

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Chuỗi trong Java

- ❑ Chuỗi là một lớp
- ❑ Các chuỗi là các hằng
- ❑ Khai báo:

```
String greeting = "Hello world!";  
char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.' };  
String helloString = new String(helloArray);
```

Viện Kỹ Thuật & Công Nghệ

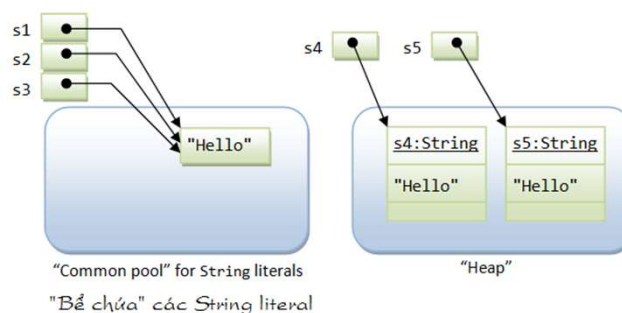
GV: Trần Xuân Hào



So sánh chuỗi

- Việc so sánh chuỗi được thực hiện bởi phương thức **equals()**, không nên sử dụng toán tử “==”

```
String s1 = "Hello";           // String literal
String s2 = "Hello";          // String literal
String s3 = s1;                // Cùng tham chiếu (trỏ tới cùng vị trí)
String s4 = new String("Hello"); // Tạo mới một đối tượng String
String s5 = new String("Hello"); // Tạo mới một đối tượng String
```



Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



So sánh chuỗi

- equals()** với ==

Phương thức **equals()** sử dụng để so sánh 2 đối tượng, với String nó có ý nghĩa là so sánh nội dung của 2 chuỗi. Toán tử “==” có ý nghĩa là so sánh địa chỉ vùng bộ nhớ lưu trữ của đối tượng.

```
String s1 = "Hello";           // String literal
String s2 = "Hello";          // String literal
String s3 = s1;                // Cùng tham chiếu (trỏ tới cùng một vị trí)
String s4 = new String("Hello"); // Tạo mới một đối tượng String
String s5 = new String("Hello"); // Tạo mới một đối tượng String

s1 == s1;                      // true, cùng trỏ vào một vị trí
s1 == s2;                      // true, s1 và s2 cùng trỏ tới 1 vị trí trong "bể chứa" (common pool)
s1 == s3;                      // true, s3 được gán bởi s1, nó sẽ trỏ tới vị trí s1 trỏ tới.
s1 == s4;                      // false, trỏ tới khác vị trí.
s4 == s5;                      // false, trỏ tới khác vị trí trên heap

s1.equals(s3);                 // true, cùng nội dung
s1.equals(s4);                 // true, cùng nội dung
s4.equals(s5);                 // true, cùng nội dung
```

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Ghép nối các chuỗi

- Sử dụng phương thức trong lớp String để ghép 2 chuỗi:

`string1.concat(string2) ;`

Phương thức trả về chuỗi được ghép string2 vào cuối string1.

- Có thể dùng toán tử "+" để ghép chuỗi

`"Hello," + " world" + "!"`

kết quả

`"Hello, world!"`



Một số phương thức xử lý chuỗi String

- **`toUpperCase()`** và **`toLowerCase()`**
- **`trim()`**: loại bỏ các khoảng trống trắng ở trước và sau chuỗi
- **`charAt()`**: trả về ký tự tại chỉ mục đã cho
- **`length()`**: trả về độ dài của chuỗi
- **`concat(String str)`**: ghép nối chuỗi
- **`isEmpty()`**: trả về **true** nếu chuỗi rỗng (`length() = 0`), ngược lại trả về **false**
- **`contains(Char s)`**: trả về **true** nếu chuỗi chứa **s**, ngược lại trả về **false**
- **`substring(int beginIndex, int endIndex)`**: Phương thức này trả về chuỗi bắt đầu từ vị trí *beginIndex* đến vị trí trước *endIndex*
- **`toString()`**: trả về biểu diễn chuỗi của đối tượng



Ví dụ

```
public class Test {
    public static void main(String [] args) {
        String str = new String("Xin chao cac ban");
        System.out.println("Gia tri tra ve la: ");
        System.out.println(str.toString());
    }
}
```



Lớp StringBuffer

- ❑ Lớp **StringBuffer** trong Java là lớp giống như lớp **String**, chỉ khác là được sử dụng để tạo chuỗi có thể sửa đổi.
- ❑ Các cấu tử khởi tạo của lớp **StringBuffer**:
 - ✓ **StringBuffer()**: tạo một bộ đệm chuỗi trống với dung lượng độ dài ban đầu là 16.
 - ✓ **StringBuffer(String str)**: tạo một bộ đệm chuỗi với chuỗi đã xác định.
 - ✓ **StringBuffer(int capacity)**: tạo một bộ đệm chuỗi trống với dung lượng *capacity* đã cho.

StringBuffer st = new StringBuffer(...)



Một số phương thức lớp StringBuffer

- **append(String s):** bổ sung vào cuối chuỗi đã cho chuỗi **s**.
- **insert(int offset, String s):** chèn chuỗi đã cho vào chuỗi **s** tại vị trí **offset**.
- **replace(int startIndex, int endIndex, String str):** thay thế chuỗi đã cho bằng **str** từ chỉ mục ban đầu **startIndex** và chỉ mục kết thúc **endIndex**.
- **delete(int startIndex, int endIndex):** xóa chuỗi từ chỉ mục **startIndex** và **endIndex**.
- **reverse():** đảo ngược chuỗi.
- **substring(int beginIndex):** trả về chuỗi con từ chỉ mục bắt đầu **beginIndex** đã cho.
- **substring(int beginIndex, int endIndex):** trả về chuỗi con từ **beginIndex** đến **endIndex** đã cho.
- **length():** để trả về độ dài của chuỗi.

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Ví dụ về lớp StringBuffer

- Ví dụ tại lớp đối với cách sử dụng lớp StringBuffer

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Summary

- ❑ Chuỗi là tập các ký tự được sử dụng rộng rãi trong lập trình Java.
- ❑ Trong ngôn ngữ lập trình Java, chuỗi là các đối tượng. Nó bao gồm hơn 60 phương thức và 13 hàm khởi tạo.
- ❑ Nên sử dụng phương thức **equals()**, không nên sử dụng “==” khi so sánh chuỗi.
- ❑ Có nhiều cách để ghép chuỗi.
- ❑ Lớp StringBuffer (*xem thêm lớp StringBuilder*)



1.6. Điều khiển luồng (Flow Control)



Các cấu trúc điều khiển luồng

- ❑ **Cấu trúc rẽ nhánh**
 - ✓ Câu lệnh **if-else**
 - ✓ Câu lệnh **switch-case**
- ❑ **Cấu trúc lặp**
 - ✓ Vòng lặp **while**
 - ✓ Vòng lặp **do-while**
 - ✓ Vòng lặp **for**
- ❑ **Các lệnh điều khiển**
 - ✓ **break**
 - ✓ **continue**
 - ✓ **return**

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Câu lệnh if-else

Cú pháp

```

if (điều kiện 1) {
    các câu lệnh 1;
} else if (điều kiện 2){
    các câu lệnh 2;
...
} else {
    các câu lệnh n;
}

```

Có thể dùng ?:

```

x = (biểu_thức) ? (giatri1 nếu true) : (giatri2 nếu true);
return (biểu_thức) ? (giatri1 nếu true) : (giatri2 nếu false);

```

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Ví dụ

```
class CheckNum {
    public static void main(String[] args) {
        int num = 10;
        if (num % 2 == 0) {
            System.out.println(num + " is an even number");
        } else {
            System.out.println(num + " is an odd number");
        }
    }
}
```



Câu lệnh switch – case

```
switch (biến kiểm tra) {
    case value1:
        // các lệnh 1;
        break;
    case value2:
        // các lệnh 2;
        break;
    default:
        // các lệnh đối với trường hợp còn lại;
}
```



Ví dụ

```
public class SwitchDemo {
    public static void main(String[] args) {
        int month = 8;
        String monthString;
        switch (month) {
            case 1: monthString = "January";    break;
            case 2: monthString = "February";   break;
            case 3: monthString = "March";      break;
            case 4: monthString = "April";      break;
            case 5: monthString = "May";        break;
            case 6: monthString = "June";       break;
            case 7: monthString = "July";       break;
            case 8: monthString = "August";     break;
            case 9: monthString = "September";  break;
            case 10: monthString = "October";   break;
            case 11: monthString = "November";  break;
            case 12: monthString = "December";  break;
            default: monthString = "Invalid month"; break;
        }
        System.out.println(monthString);
    }
}
```

Output will be: August

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Vòng lặp while

□ Cú pháp:

```
while (điều kiện) {
    các câu lệnh;
}
```

- ✓ Trong khi điều kiện còn đúng thì còn thực hiện các câu lệnh.
- ✓ Vòng lặp while là một cấu trúc điều khiển cho phép bạn lặp đi lặp lại một tác vụ một số lần nào đó. Với kiểu vòng lặp này, chương trình sẽ kiểm tra điều kiện trước khi thực thi phần thân vòng lặp.

Viện Kỹ Thuật & Công Nghệ

GV: Trần Xuân Hào



Ví dụ

```
class FactDemo {
    public static void main(String[] args) {
        int num = 5, fact = 1;
        while (num >= 1) {
            fact *= num;
            num--;
        }
        System.out.println("The factorial of 5 is : " + fact);
    }
}
```



Vòng lặp do – while

- ❑ Thực hiện các câu lệnh khi điều kiện còn True.
- ❑ Vòng lặp này là một dạng của **while**, chỉ khác các câu lệnh được thực hiện ít nhất 1 lần.
- ❑ Cú pháp:

```
do {
    các câu lệnh;
} while (điều kiện);
```



Ví dụ

```
class DoWhileDemo {
    public static void main(String[] args) {
        int count = 1, sum = 0;
        do {
            sum += count;
            count++;
        } while (count <= 100);
        System.out.println("The sum of first 100 numbers
is : " + sum);
    }
}
```

The sum of first 100 numbers is : 5050



Vòng lặp for

Lặp với số lần lặp cụ thể (được biết trước) nào đó

Cú pháp

```
for (khởi tạo; điều kiện tiếp tục lặp; giá trị tăng) {  
    các câu lệnh;  
}
```



Ví dụ

```
class ForDemo {
    public static void main(String[] args) {
        int count = 1, sum = 0;
        for (count = 1; count <= 10; count += 2) {
            sum += count;
        }
        System.out.println("The sum of first 5 odd
numbers is : " + sum);
    }
}
```

The sum of first 5 odd numbers is : 25



Câu lệnh break

- Câu lệnh *break* được sử dụng để dừng toàn bộ vòng lặp. Từ khóa *break* phải được sử dụng bên trong bất kỳ vòng lặp nào hoặc một lệnh switch.

```
class BreakDemo {
    public static void main(String [] args) {
        for (int count = 1; count <= 100; count++) {
            if (count == 10) {
                break;
            }
            System.out.println("The value of num is : " +
count);
        }
        System.out.println("The loop is over");
    }
}
```



Câu lệnh *continue*

- Câu lệnh *continue* có thể được sử dụng trong bất kỳ cấu trúc điều khiển vòng lặp nào. Nó làm cho vòng lặp ngay lập tức tiếp tục tiến trình lặp tiếp theo của vòng lặp.



Ví dụ

```
class ContinueDemo {
    public static void main(String[] args) {
        String searchMe = "peter piper picked a peck of pickled peppers";
        int max = searchMe.length();
        int numPs = 0;
        for (int i = 0; i < max; i++) {
            //interested only in p's
            if (searchMe.charAt(i) != 'p') {
                continue;
            }
            //process p's
            numPs++;
        }
        System.out.println("Found " + numPs + " p's in the string.");
    }
}
```

Here is the output of this program: Found 9 p's in the string.



Câu lệnh return

- ❑ Câu lệnh return thoát khỏi phương thức hiện tại, trở về nơi mà phương thức được gọi.
- ❑ Câu lệnh return có 2 dạng:
 - ✓ Trả về một giá trị: `return ++count;`
 - ✓ Không trả về giá trị: `return;`
- ❑ Kiểu dữ liệu trả về phải trùng với kiểu dữ liệu của phương thức khi khai báo.



Summary

- ❑ **Cấu trúc rẽ nhánh**
 - ✓ Câu lệnh **if-else**
 - ✓ Câu lệnh **switch-case**
- ❑ **Cấu trúc lặp**
 - ✓ Vòng lặp **while**
 - ✓ Vòng lặp **do-while**
 - ✓ Vòng lặp **for**
- ❑ **Các lệnh điều khiển**
 - ✓ **break**
 - ✓ **continue**
 - ✓ **return**