

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Bài 2: Java cơ bản

Trinh Thi Van Anh – PTIT

Nội dung

- Giới thiệu về Java
- Định danh
- Các kiểu dữ liệu
- Toán tử
- Cấu trúc điều khiển
- Scanner, Random, Math, Wrapper Class, Regular Expression
- Mảng

Ngôn ngữ lập trình Java

- Ngôn ngữ lập trình Java được phát triển vào năm 1991 bởi Sun Microsystems bởi James Gosling với tên "Oak" (nay là Oracle)
- 1995: Tên chính thức là Java
- Tiêu chí phát triển:
"*Write Once, Run Anywhere*"

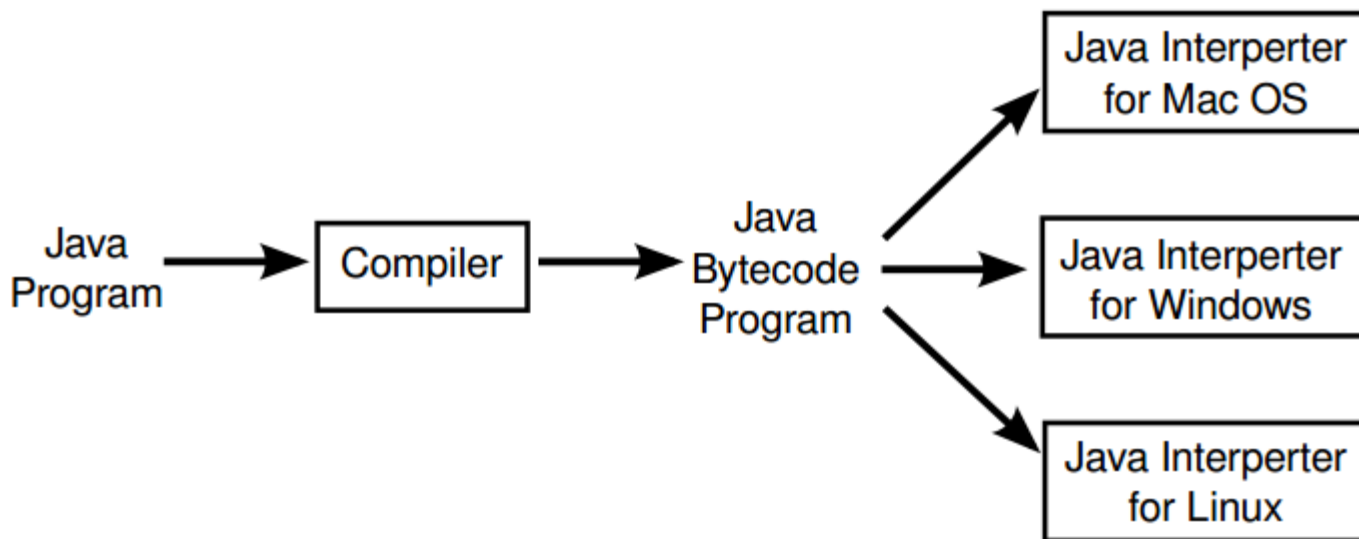


Java platform

- Java Platform – nền tảng Java
 - Được xây dựng để phát triển các ứng dụng và phân phối trên môi trường đa nền (các HĐH, điện thoại, thiết bị nhúng, enterprise server...)
 - Sử dụng ngôn ngữ Java (và một số ngôn ngữ khác)
- Tránh nhầm lẫn với ngôn ngữ lập trình Java
- Các thành phần của Java Platform
 - Các API
 - Java Platform cung cấp các API để lập trình viên không cần phải sử dụng các API của HĐH
 - Java Virtual Machine (JVM)
 - Có thể chạy trên các software platform khác hoặc trực tiếp trên phần cứng
 - Mỗi một platform sử dụng một JVM riêng

Mô hình biên dịch của Java

- Mô hình biên dịch của Java platform
- Mã nguồn được biên dịch thành Java bytecode; sau đó được thông dịch trên JVM thành các mã lệnh thực thi bởi trình thông dịch Just-In-Time (JIT)



Đặc điểm của Java

- Đơn giản
- Hướng đối tượng
- Đa nhiệm
- An toàn
- Garbage Collection
- Máy ảo (biên dịch và thông dịch)
- Khả chuyển (Portability)
- Phân tán

Cú pháp cơ bản

- Là ngôn ngữ lập trình phân biệt chữ hoa, chữ thường (case-sensitive)
- Cú pháp tương tự C/C++

Cài đặt

- Cài Java Development Kit (JDK)
 - <http://www.oracle.com/technetwork/java/javase/downloads>
- Cài IDE
 - Notepad / Notepad++ (<https://notepad-plus-plus.org>)
 - Eclipse (<http://www.eclipse.org>)
 - NetBeans (<http://netbeans.org>)

**Môi trường phát triển tích hợp
IDE (Integrated Development
Environment)**

JDK (Java Development Kit)

- JDK 1.02 (1995)
- JDK 1.1 (1996)
- JDK 1.2 (1998)
- JDK 1.3 (2000)
- JDK 1.4 (2002)
- JDK 1.5 (2004) a. k. a. JDK 5 or Java 5
- JDK 1.6 (2006) a. k. a. JDK 6 or Java 6
- JDK 1.7 (2011) a. k. a. JDK 7 or Java 7
- JDK 1.8 (2014) a. k. a. JDK 8 or Java 8

Bộ công cụ phát triển Java (jdk) gồm trình biên dịch, thông dịch, trợ giúp, soạn tài liệu... và các thư viện chuẩn

JDK Editions

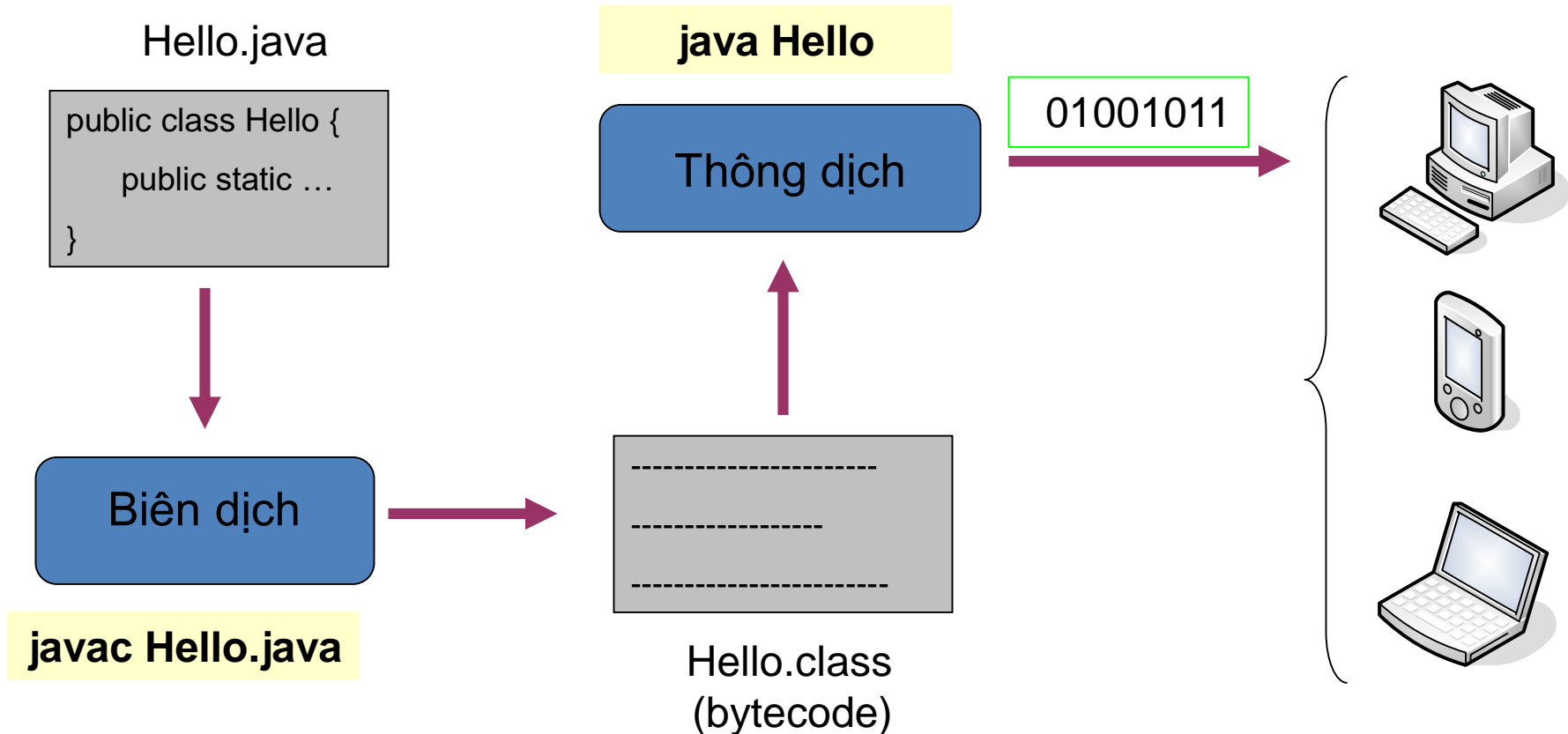
- Java Standard Edition (J2SE)
 - J2SE được dùng để phát triển các ứng dụng đơn người dùng (standalone) hoặc applet.
- Java Enterprise Edition (J2EE)
 - J2EE được dùng để phát triển các ứng dụng trên server như Java servlets hoặc Java ServerPages.
- Java Micro Edition (J2ME).
 - J2ME được dùng để phát triển các ứng dụng cho thiết bị di động.

Links for reading

- Java tutorial
 - <https://docs.oracle.com/javase/tutorial/>
- data type and java platform
 - http://www.tutorialspoint.com/java/java_basic_datatypes.htm
- java.util.Collections
 - http://www.tutorialspoint.com/java/util/java_util_collections.htm
- java.util.ArrayList
 - http://www.tutorialspoint.com/java/util/java_util_arraylist.htm

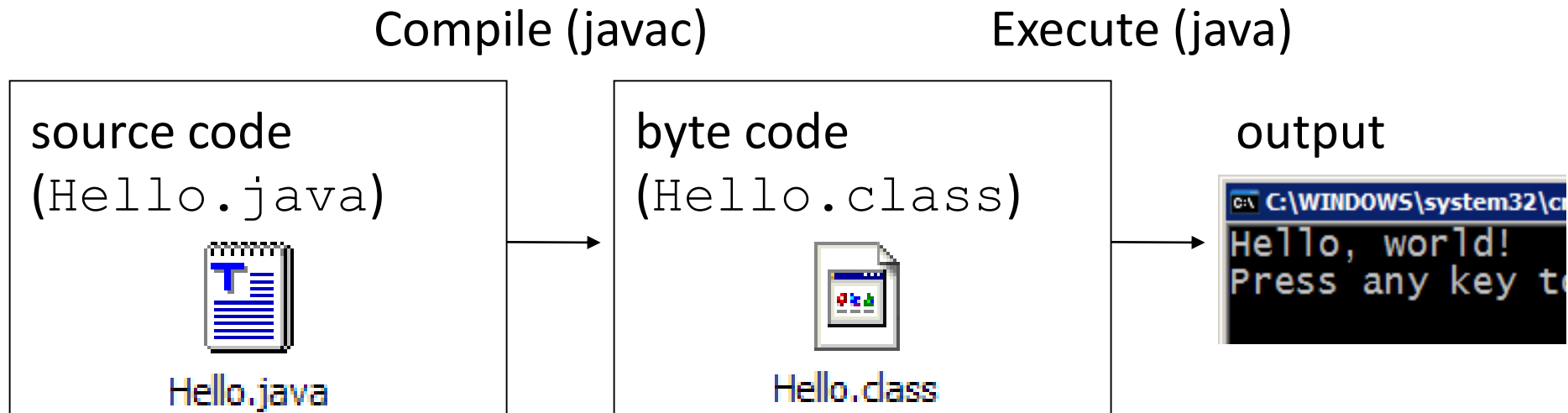
Phát triển ứng dụng Java

- Các bước phát triển



Biên dịch

- Trước khi chạy chương trình cần được biên dịch.
- **Trình biên dịch (compiler)** : Chuyển một chương trình máy tính được viết bởi một ngôn ngữ nào đó (ex., Java) sang một ngôn ngữ khác (i.e., byte code)



Máy ảo Java (Java Virtual Machine – JVM or VM)

- JVM chạy byte code
 - Sử dụng lệnh “java” để chạy
 - Chỉ hiểu byte code (file “.class”)
- JVM làm Java khác so với các ngôn ngữ trước đây (C, C++)
 - Trình biên dịch của các ngôn ngữ khác chuyển trực tiếp sang mã máy. Java cần thêm một bước phụ (byte code).
 - Chậm hơn
 - Linh động hơn: cho phép một byte code có thể chạy trên bất kỳ máy ảo Java nào

Một chương trình Java cơ bản

```
1 // Tên file : Hello.java
2 /* @author AnhTTV */
3
4 public class Hello
5 {
6     // Phương thức main, điểm bắt đầu của chương trình
7     public static void main( String args[ ] )
8     {
9         System.out.println( "Hello World!" );
10    } // Kết thúc phương thức main
11 } // Khai báo lớp
```

Tên lớp chứa hàm main phải giống tên file

Điểm bắt đầu và kết thúc của lớp

Dấu hiệu chú thích =>
Làm cho chương trình dễ hiểu hơn. Trình biên dịch sẽ bỏ qua những dòng có dấu chú thích

Khai báo lớp
Mỗi CT phải có ít nhất một khai báo lớp

Hiện thị dãy ký tự ra màn hình

Phương thức main() sẽ được gọi đầu tiên. Mỗi CT thực thi phải có một phương thức main()

Các câu lệnh phải kết thúc bằng dấu chấm phẩy

Cú pháp

- **Cú pháp:** Tập các cấu trúc và lệnh hợp lệ có thể dùng được.
- Ví dụ:
 - Dấu chấm phẩy sau mỗi câu lệnh.
 - Nội dung của class được chứa trong hai dấu ngoặc { }.
- Lỗi lập trình
 - Lỗi cú pháp: Phát hiện bởi trình biên dịch
 - Lỗi runtime: Chương trình bị thoát
 - Lỗi logic: Kết quả không đúng

Lỗi cú pháp (1)

■ Cấu trúc chương trình bị lỗi.

```
1 public class Hello {  
2     pooblic static void main(String[] args) {  
3         System.owt.println("Hello world!")  
4     }  
5 }
```

compiler output:

```
2 errors found:  
File: Hello.java [line: 2]  
Error: Hello.java:2: <identifier> expected  
File: Hello.java [line: 3]  
Error: Hello.java:3: ';' expected
```

Lỗi cú pháp (2)

- Java phân biệt chữ hoa chữ thường - Hello và hello là khác nhau

```
1 Public class Hello {  
2     public static void main(String[] args) {  
3         System.out.println("Hello world!");  
4     }  
5 }
```

compiler output:

1 error found:

File: Hello.java [line: 1]

Error: Hello.java:1: class, interface, or enum expected

Lỗi trong coding

- Thường xuyên gặp !
- Có thể được nhận biết bởi IDE
- Ví dụ

- **Lỗi** khai báo 2 biến cùng tên:

```
int x;
```

```
int x;          // ERROR: x already exists
```

- **Lỗi** đọc giá trị biến trước khi được khởi tạo:

```
int x;
```

```
System.out.println(x); //ERROR: x has no  
value
```

Runtime Errors

```
1 public class ShowRuntimeErrors {  
2     public static void main(String[] args) {  
3         System.out.println(1/0);  
4     }  
5 }
```

```
no-one:src ntkhanh$ java ptit.oop.s1.ShowRuntimeErrors  
Exception in thread "main" java.lang.ArithmeticException: / by zero  
    at ptit.oop.s1.ShowRuntimeErrors.main(ShowRuntimeErrors.java:5)
```

System.out.println

- `System.out.println()` : Lệnh in dòng trên console
- 2 cách sử dụng
 - `System.out.println();`
`System.out.println("<message>");`
 - In thông điệp trên console.
`System.out.println();`
 - In dòng trống.

Chú thích (Comment)

- Chú thích: Ghi chép trong code -> giải thích code để hiểu hơn.
 - Trình biên dịch bỏ qua.
 - Màu khác với code.

- Cú pháp:

`/* <comment text; may span multiple lines> */`

or,

`// <comment text, on one line>`

- Ví dụ:

```
/** A comment goes here. */
```

```
/* It can even span  
multiple lines. */
```



```
// This is a one-line comment.
```

Quy ước viết mã

- Programming style?
 - Sự thụt dòng
 - Viết hoa
 - Định dạng/ khoảng trống
 - Cấu trúc code
 - Không dư thừa
 - ...
- Why is it important?

Định danh (Identifier)

hất, dù trạng thái của nó
ợng

 An_Identifier a_2nd_Identifier Go2 \$10	 An-Identifier 2nd_Identifier goto 10\$
---	--

cá
:
ữ số

- Bắt đầu bởi một chữ số
- Trùng với từ khóa
- Phân biệt chữ hoa chữ thường
 - Yourname, yourname, YourName và yourName là 4 định danh khác nhau

Quy ước đặt tên

- Quy ước với định danh (naming convention):
 - Bắt đầu bằng chữ cái
 - Gói (package): tất cả sử dụng chữ thường
 - theexample
 - Lớp (Class): viết hoa chữ cái đầu tiên trong các từ ghép lại
 - TheExample
 - Phương thức/thuộc tính (method/field): Bắt đầu bằng chữ thường, viết hoa chữ cái đầu tiên trong các từ còn lại
 - theExample
 - Hằng (constants): Tất cả viết hoa
 - THE_EXAMPLE

Các từ khóa

- Literals
 - null true false
- Từ khóa (keyword)

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

- Class, cLaSs thì thế nào ? Được nhưng không khuyến khích !

Các kiểu dữ liệu

Trong Java kiểu dữ liệu được chia thành hai loại:

- Kiểu dữ liệu nguyên thủy (primitive)
 - Số nguyên (integer)
 - Số thực (float)
 - Ký tự (char)
 - Giá trị logic (boolean)
- Kiểu dữ liệu tham chiếu (reference)
 - Mảng (array)
 - Đối tượng (object)

Kiểu dữ liệu nguyên thủy

- Mọi biến đều phải khai báo một kiểu dữ liệu
 - Các kiểu dữ liệu cơ bản chứa một giá trị đơn
 - Kích thước và định dạng phải phù hợp với kiểu của nó
- Java phân loại thành 4 kiểu dữ liệu nguyên thủy

Categories:

a. integer

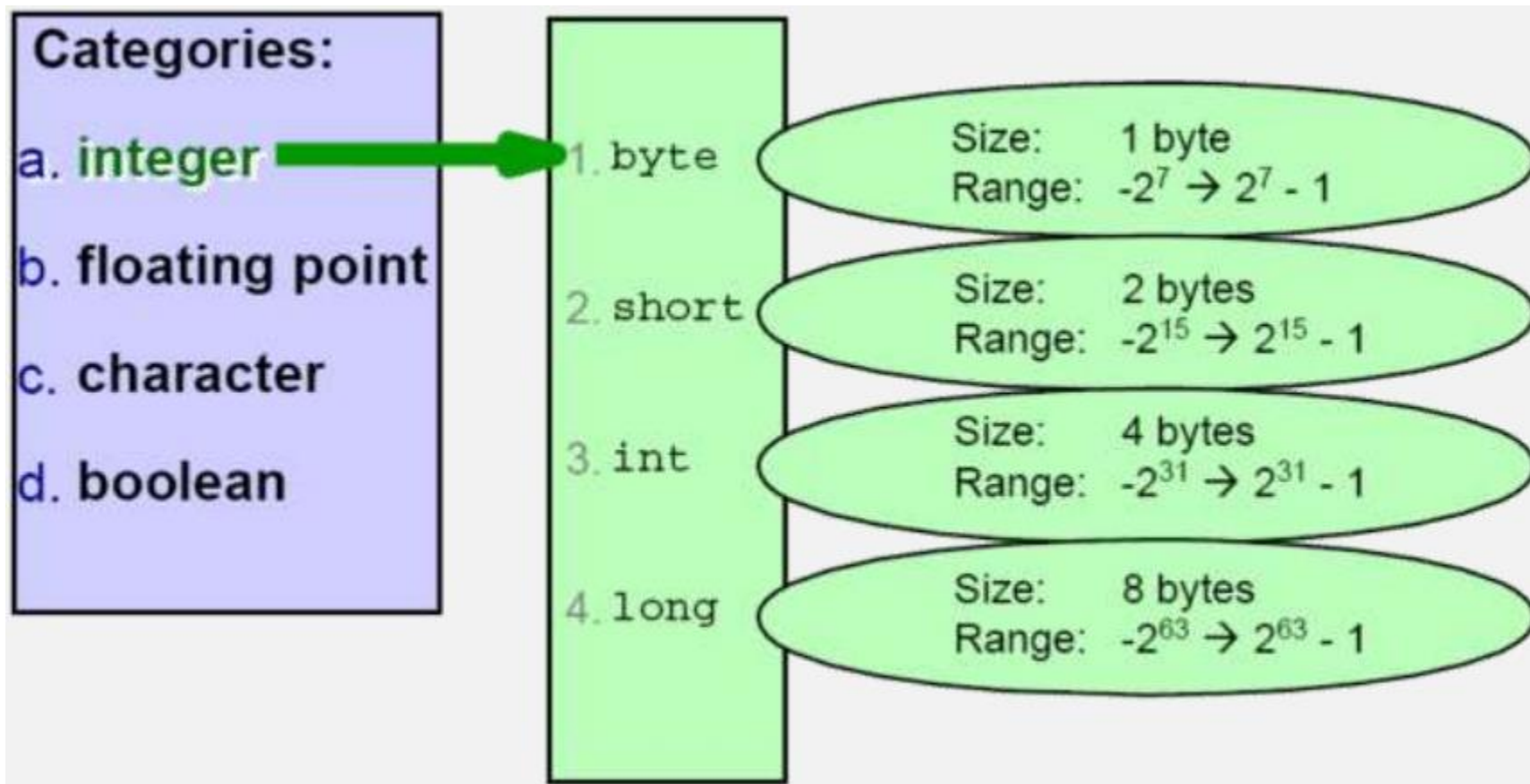
b. floating point

c. character

d. boolean

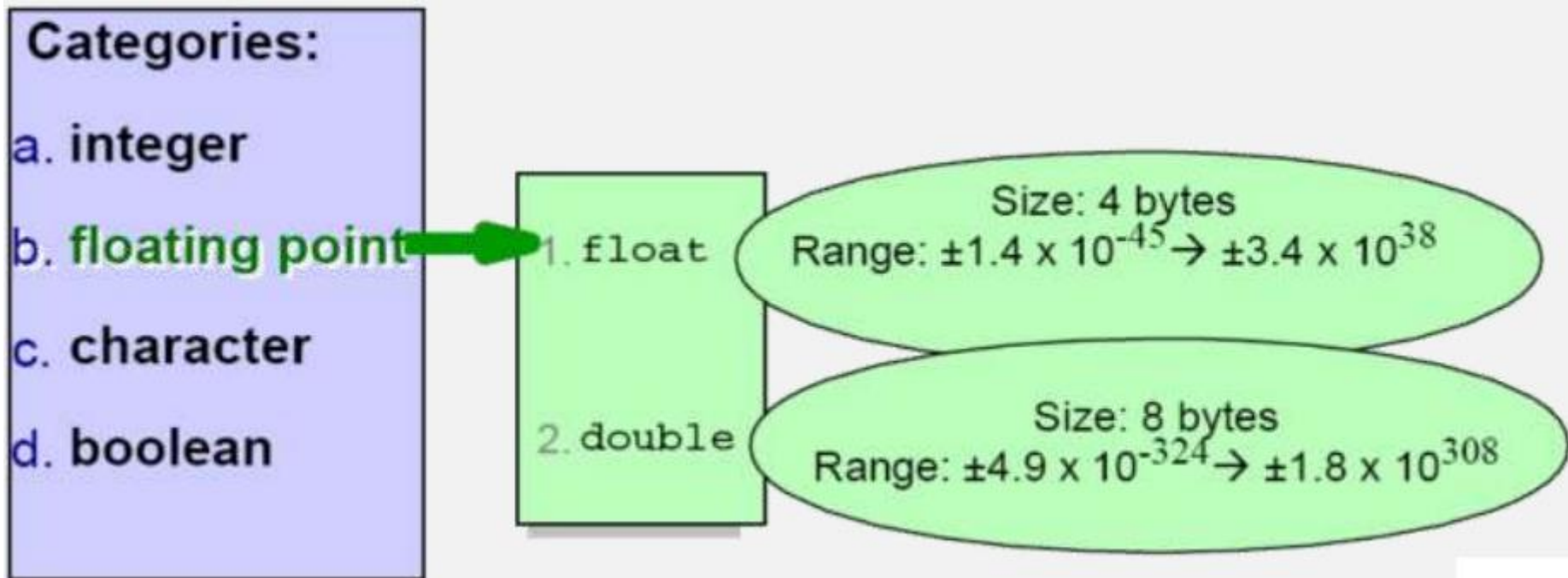
Số nguyên

- Số nguyên có dấu
 - Giá trị mặc định: 0



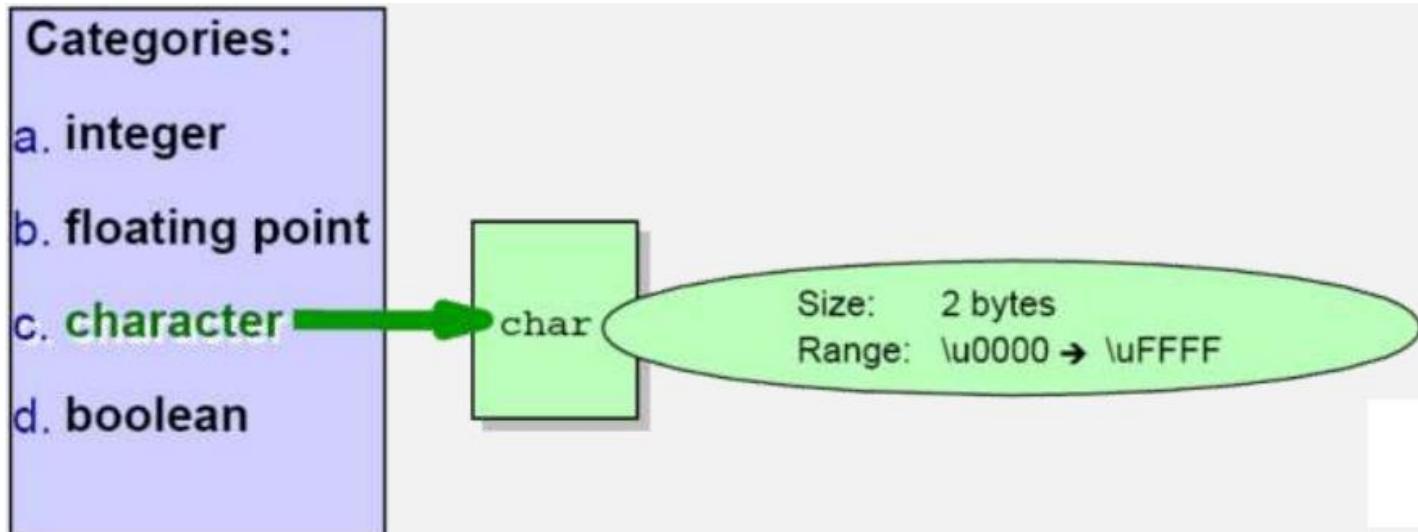
Số thực

- Số thực dấu phẩy động
 - Giá trị mặc định: 0.0



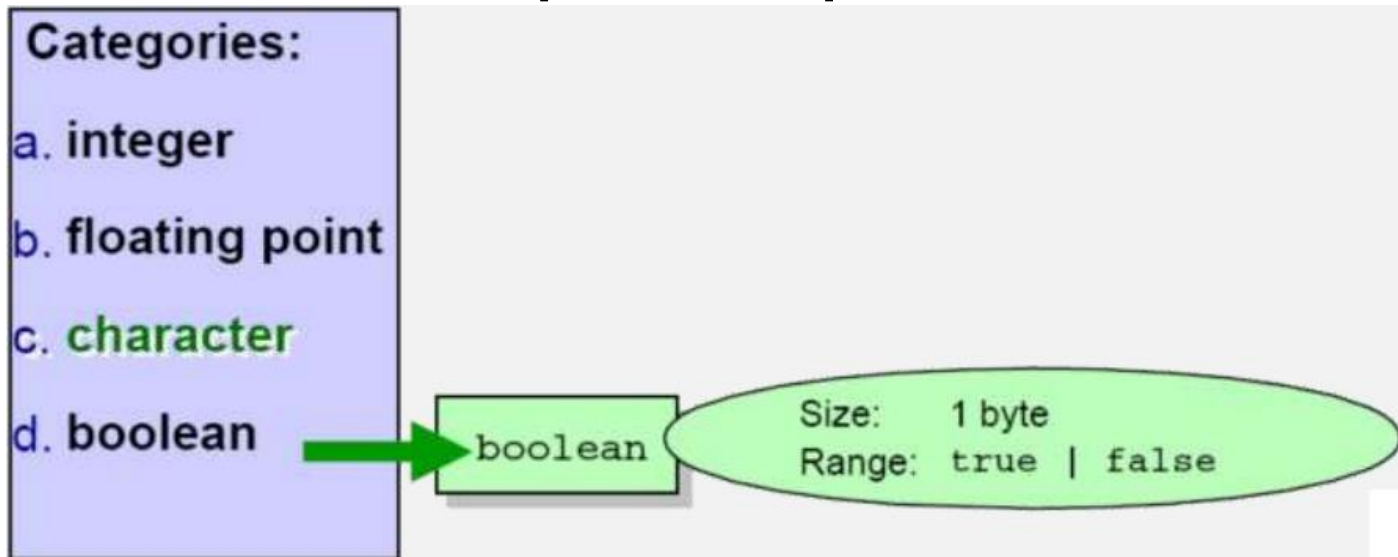
Ký tự

- Ký tự Unicode không dấu, được đặt giữa hai dấu nháy đơn
- 2 cách gán giá trị:
 - Sử dụng các chữ số trong hệ 16: `char uni = '\u05D0';`
 - Sử dụng ký tự: `char a = 'A';`
 - Giá trị mặc định là giá trị zero (`\u0000`)



Nguyên dạng

- Giá trị boolean được xác định rõ ràng trong Java
 - Một giá trị int không thể sử dụng thay cho giá trị boolean
 - Có thể lưu trữ giá trị hoặc true hoặc false
- Biến boolean được khởi tạo là false



Nguyên dạng

- Literal là một giá trị của các kiểu dữ liệu nguyên thủy và xâu ký tự.
- Gồm 5 loại:
 - Integer
 - floating point
 - Boolean
 - Character
 - String

Literals

integer.....	7
floating point...	7.0f
boolean.....	true
character.....	'A'
string.....	"A"

Số nguyên

- Hệ cơ số 8 (Octals) bắt đầu với chữ số 0
 - $032 = 011\ 010(2) = 16 + 8 + 2 = 26(10)$
- Hệ cơ số 16 (Hexadecimals) bắt đầu với 0 và ký tự x
 - $0x1A = 0001\ 1010(2) = 16 + 8 + 2 = 26(10)$
- Kết thúc bởi ký tự "L" thể hiện kiểu dữ liệu long
 - 26L
- Ký tự hoa, thường cho giá trị bằng nhau
 - 0x1a , 0x1A , 0X1a , 0X1A đều có giá trị 26 trong hệ decimal

Số thực

- float kết thúc bằng ký tự f (hoặc F)
 - 7.1f
- double kết thúc bằng ký tự d (hoặc D)
 - 7.1D
- e (hoặc E) được sử dụng trong dạng biểu diễn khoa học:
 - 7.1e2
- Một giá trị thực mà không có ký tự kết thúc đi kèm sẽ có kiểu là double
 - 7.1 giống như 7.1d

boolean, ký tự và xâu ký tự

- boolean:
 - True
 - False
- Ký tự:
 - Được đặt giữa 2 dấu nháy đơn
 - Ví dụ: 'a', 'A' hoặc '\uffff'
- Xâu ký tự:
 - Được đặt giữa hai dấu nháy kép
 - Ví dụ: "Hello world", "Xin chào bạn",...

Escape sequence

- Các ký tự điều khiển nhấn phím
 - \b backspace
 - \f form feed
 - \n newline
 - \r return (về đầu dòng)
 - \t tab
- Hiển thị các ký tự đặc biệt trong xâu
 - \" quotation mark
 - \' apostrophe
 - \\ backslash

Chuyển đổi kiểu dữ liệu (casting) (1)

- Java là ngôn ngữ định kiểu chặt
 - Gán sai kiểu giá trị cho một biến có thể dẫn đến các lỗi biên dịch hoặc các ngoại lệ của JVM
- JVM có thể ngầm định chuyển từ một kiểu dữ liệu hẹp sang một kiểu rộng hơn
- Để chuyển sang một kiểu dữ liệu hẹp hơn, cần phải định kiểu rõ ràng.

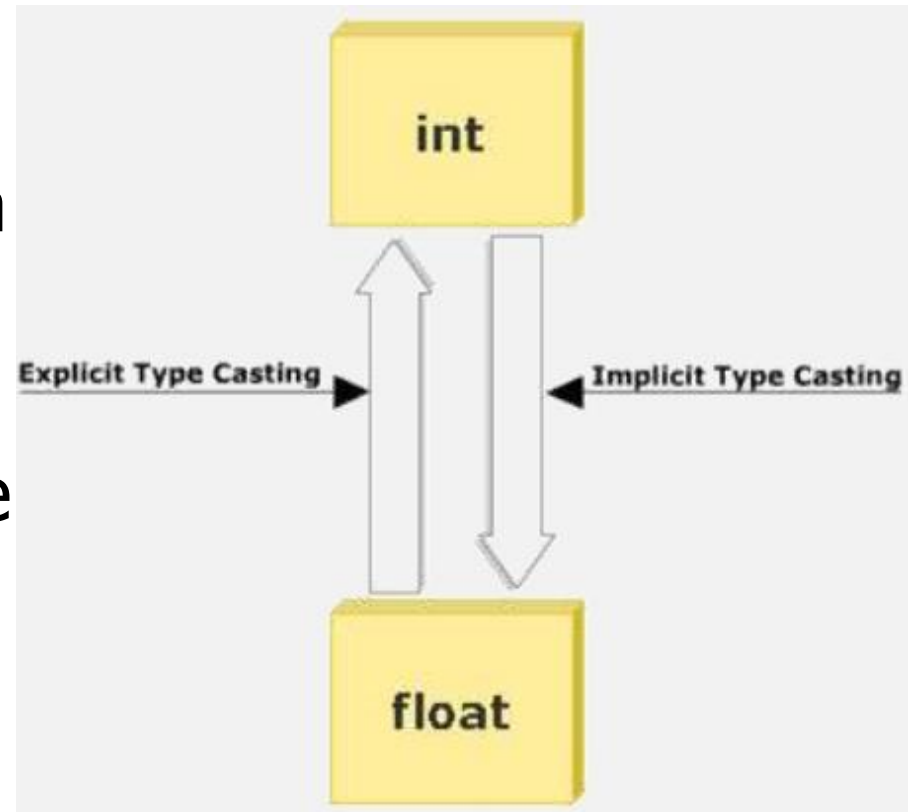
```
int a, b;  
short c;  
a = b + (int)c;
```

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

```
double f;  
long g;  
f = g;  
g = f; //error
```

Casting (2)

- Chuyển đổi kiểu sẽ được thực hiện tự động nếu không xảy ra mất mát thông tin
 - byte • short • int • long • float • double
- Ép kiểu trực tiếp (explicit cast) được yêu cầu nếu có “nguy cơ” giảm độ chính xác



Ví dụ - chuyển đổi kiểu

- `long p = (long) 12345.56; // p == 12345`
- `int g = p; // không hợp lệ dù kiểu int`
`//có thể lưu giá trị 12345`
- `char c = 't';`
- `int j = c; // tự động chuyển đổi`
- `short k = c; // không hợp lệ`
- `short k = (short) c; // ép kiểu trực tiếp`
- `float f = 12.35; // không hợp lệ`

Khai báo và khởi tạo biến

- Các biến đơn (biến không phải là mảng) cần phải được khởi tạo trước khi sử dụng trong các biểu thức
 - Có thể kết hợp khai báo và khởi tạo cùng một lúc.
 - Sử dụng = để gán (bao gồm cả khởi tạo)
 - Ví dụ:

```
int i, j;           // Khai báo biến
```

```
i = 0;
```

```
int k = i + 1;
```

```
float x = 1.0f, y = 2.0f;
```

```
System.out.println(i); // In ra 0
```

```
System.out.println(k); // In ra 1
```

```
System.out.println(j); // Lỗi biên dịch
```

Câu lệnh

- Các câu lệnh kết thúc bởi dấu ;
- Nhiều lệnh có thể viết trên một dòng
- Một câu lệnh có thể viết trên nhiều dòng
 - Ví dụ:

```
System.out.println("This is  
part of the same line");
```

```
a=0; b=1; c=2;
```

Toán tử (Operators)

- Kết hợp các giá trị đơn hoặc các biểu thức con thành những biểu thức mới, phức tạp hơn và có thể trả về giá trị.
- Java cung cấp nhiều dạng toán tử sau:
 - Toán tử số học
 - Toán tử bit, toán tử quan hệ
 - Toán tử logic
 - Toán tử gán
 - Toán tử một ngôi

Toán tử

- Toán tử số học
 - `+`, `-`, `*`, `/`, `%`
- Toán tử bit
 - AND: `&`, OR: `|`, XOR: `^`, NOT: `~`, Dịch bit: `<<`, `>>`
- Toán tử quan hệ
 - `==`, `!=`, `>`, `<`, `>=`, `<=`, `instanceof`
- Toán tử logic
 - `&&`, `||`, `!`
- Toán tử một ngôi
 - Đảo dấu: `+`, `-` ; Tăng giảm 1 đơn vị: `++`, `--`
 - Phủ định một biểu thức logic: `!`
- Toán tử gán
 - `=`, `+=`, `-=`, `%=` tương tự với `>>`, `<<`, `&`, `|`, `^`

Thứ tự ưu tiên của toán tử

Operators	Precedence
postfix	expr++ expr--
unary	++expr --expr +expr -expr ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	
logical AND	&&
logical OR	
ternary	? :
assignment	= += -= *= /= %= &= ^= = <<= >>= >>>=

Lớp Scanner

- Tạo đối tượng Scanner để đọc console:

```
Scanner <name> = new Scanner(System.in);
```

- Ví dụ:

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

- Chuẩn vào từ bàn phím [standard input]: dùng Scanner ánh xạ với **System.in**
 - hasNext() //is something available to read? Returns true or false.
 - nextInt()//get an int
 - nextDouble()//get a double
 - next()//get a String (delimited by whitespace)
 - nextLine() //get the rest of the line as a String
- Note:There is no method to read a character! Read a string instead.

Ví dụ

```
int a;  
float u;  
String ten;  
Scanner in=new Scanner(System.in);  
System.out.println("\n name: ");  
ten=in.nextLine();  
System.out.print("\n a=");  
a=in.nextInt();  
System.out.println("\n u=");  
u=in.nextFloat();  
String tt=a+", "+u+", "+ten;  
System.out.println("Cac thu da nhap:"+tt);
```

Cấu trúc điều khiển

- Lệnh if – else
- Lệnh switch – case
- Vòng lặp for
- Vòng lặp while và do while
- Các lệnh thay đổi cấu trúc điều khiển

Lệnh if - else

- **if** (condition) {
 //one or more statements;
}
- **if** (condition) {
 //one or more statements;
} **else** {
 //one or more statements;
}
- **if** (condition) {
 //one or more statements;
} **else if** (condition) {
 //one or more statements;
...
} **else** {
 //one or more statements;
}

Ví dụ: Kiểm tra số chẵn – lẻ

```
class CheckNumber{  
    public static void main(String args[]) {  
        int num =10;  
        if (num %2 == 0)  
            System.out.println(num+" So chan");  
        else  
            System.out.println (num+"So le");  
        }  
    }
```

Lệnh switch - case

- Kiểm tra một biến đơn với nhiều giá trị khác nhau và thực hiện trường hợp tương ứng
 - break: Thoát khỏi lệnh switchcase
 - default kiểm soát các giá trị nằm ngoài các giá trị case:

```
switch (expression) {  
    case value1:  
        //statement;  
        break;  
    case value2:  
        //statement;  
        break;  
    ...  
    case valueN:  
        //statement;  
        break;  
    default:  
        //default statement;  
}
```

```
public class SwitchDemo {  
    public static void main(String[] args) {  
        int m = 8;  
        String mst;  
        switch (m) {  
            case 1: mst = "thang gieng"; break;  
            case 2: mst = "thang hai"; break;  
            case 3: mst = "thang ba"; break;  
            case 4: mst = "thang tu"; break;  
            case 5: mst = "thang nam"; break;  
            case 6: mst = "thang sau"; break;  
            case 7: mst = "thang bay"; break;  
            case 8: mst = "thang tam"; break;  
            case 9: mst = "thang chin"; break;  
        }  
    }  
}
```

```
    case 10: mst = " tháng mười"; break;
    case 11: mst = "tháng mười một"; break;
    case 12: mst = " tháng mười hai"; break;
    default: mst = "không có tháng này";
              break;
}
System.out.println(mst);
}
}
```

```
public class TheSwitch {  
    public static void main(String[] args) {  
        String c="blue" ;  
        String mg="";  
        switch(c) {  
            case "blue": mg = "mau xanh duong"; break;  
            case "red": mg = "mau do"; break;  
            case "green": mg = "mau xanh luc"; break;  
            case "yellow": mg = "mau vang"; break;  
            default: mg = "mau khac";  
        }  
        System.out.println(mg);  
    }  
}
```

Vòng lặp for

- **Cú pháp:**

```
for (start_expr; test_expr; increment_expr) {  
    // code to execute repeatedly  
}
```

- **3 biểu thức đều có thể vắng mặt**

- **Có thể khai báo biến trong câu lệnh for**

- Thường sử dụng để khai báo một biến đếm
- Thường khai báo trong biểu thức “start”
- Phạm vi của biến giới hạn trong vòng lặp

- **Ví dụ:**

```
for (int index = 0; index < 10; index++) {  
    System.out.println(index);  
}
```

Biến đổi vòng lặp for

- Biến khởi tạo và cuối cùng có thể bất kỳ biểu thức nào :
- Ví dụ :

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

Output:

```
-3  
-2  
-1  
0  
1  
2
```


■ Ví dụ :

```
for (int i = 1 + 3 * 4; i <= 5248 %  
    100; i++) {  
    System.out.println(i + "=" + (i * i));  
}  
System.out.println("T-minus");  
  
for (int i = 3; i >= 1; i--) {  
    System.out.println(i);  
}  
System.out.println("Blastoff!");
```

Vòng lặp for each

- Trong Java 5, vòng lặp for each (hay enhanced for) đã được giới thiệu. Nó được sử dụng chủ yếu với các mảng. Cú pháp

```
for (declaration : expression) {  
    //Statements }  
}
```

- Declaration - Khai báo: Biến khối được khai báo mới, mà là một kiểu tương thích với các phần tử của mảng bạn đang truy cập. Biến này sẽ có sẵn trong khối for và giá trị của nó sẽ là giống như phần tử mảng hiện tại.
- Expression này có thể là một biến mảng hoặc gọi phương thức mà trả về một mảng.

```
public class Example{
    public static void main(String args[]){
        int [] numbers = {10, 20, 30, 40, 50};
        for(int x : numbers ){
            System.out.print( x );// numbers[i]
            System.out.print(",");
        }
        System.out.print("\n");
        String [] names ={"James", "Larry", "Tom",
"Lacy"};
        for( String name : names ) {
            System.out.print( name );
            System.out.print(",");
        }
    }
}
```

■ Output:

10,20,30,40,50,
James,Larry,Tom,Lacy,

Vòng lặp while

- **Vòng lặp while** : Câu trúc điều khiển thực hiện lặp một phép kiểm tra và thực thi một nhóm lệnh, nếu phép kiểm tra cho kết quả đúng.

- Cú pháp

```
while ( <test> ) {  
    <statement(s)>;  
}
```

- Ví dụ :

```
int number = 1;  
while (number <= 200) {  
    System.out.print(number + " ");  
    number *= 2;  
}
```

Output:

1 2 4 8 16 32 64 128

do/while

- **do/while** : Cấu trúc điều khiển thực thi lặp lại một số lệnh khi điều kiện kiểm tra vẫn đúng; điều kiện kiểm tra được thực hiện sau mỗi vòng lặp.

- cú pháp:

```
do {  
    <statement(s)>;  
} while ( <test> );
```

- Ví dụ:

```
// roll until we get a number other than 3  
Random rand = new Random();  
int die;  
do {  
    die = rand.nextInt();  
} while (die == 3);
```

Menu

```
while(true) {  
    System.out.print("\n 1.Lua  chon 1");  
    System.out.print("\n 2.Lua  chon 2");  
    System.out.print("\n 3.Lua  chon 3'");  
    System.out.print("\n 0.Thoat");  
    System.out.print("\n HAY CHON 1,2,3 or  
0: ");  
    Scanner in = new Scanner(System.in);  
    int c = in.nextInt();  
    System.out.print("\n");  
}
```

```
switch(c) {  
    case 1: //thuc hien lua chon 1  
        break;  
    case 2:// thuc hien lua chon 2  
        break;  
    case 3: // thuc hien lua chon 3  
        break;  
    case 0: System.exit(0);  
        break;  
}  
}
```

Các lệnh thay đổi cấu trúc điều khiển

Break


- Có thể được sử dụng để thoát ra ngoài câu lệnh switch
- Kết thúc vòng lặp for, while hoặc do...while
- Có hai dạng:
 - Gắn nhãn: Tiếp tục thực hiện câu lệnh tiếp theo sau vòng lặp được gắn nhãn
 - Không gắn nhãn: Thực hiện câu lệnh tiếp theo bên ngoài vòng lặp

continue

- Có thể được sử dụng cho vòng lặp for, while hoặc do...while
- Bỏ qua các câu lệnh còn lại của vòng lặp hiện thời và chuyển sang thực hiện vòng lặp tiếp theo.


Ví dụ break

```
int i = 1;  
while (true) {  
    if (i == 3)  
        break;  
    System.out.println("This is a " + i + " iteration");  
    ++i;  
}
```

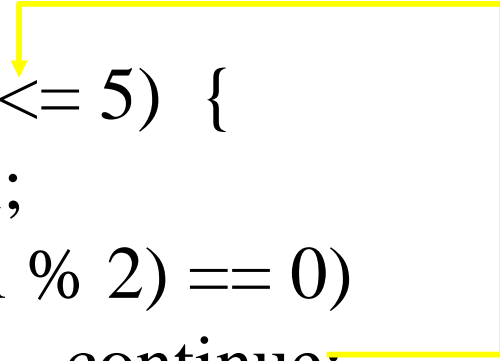


Ví dụ continue

```
for (i=0; i<=5; ++i) {  
    if (i % 2 == 0)  
        continue;  
    System.out.println("This is a " + i + " iteration");  
}
```



```
i = 0;  
while (i <= 5) {  
    ++i;  
    if (i % 2) == 0)  
        continue;  
    System.out.println("This is a odd iteration - " + i);  
}
```



Lớp Random

- Lớp Random là một phần của gói java.util
 - Dùng để phát sinh các số ngẫu nhiên
 - Tạo đối tượng:
- Phát sinh số ngẫu nhiên có miền giá trị từ 0 đến (n-1)

```
Random rd=new Random();
```

```
int a=rd.nextInt(n); //nextInt()  
//nextFloat()
```

- Phát sinh số ngẫu nhiên có miền giá trị từ a đến b

```
int t = rd.nextInt(b-a+1) +a;
```

Lớp Math

- Là một phần của gói java.lang
- Chứa các phương thức có chức năng tính toán về toán học:
 - Lũy thừa (pow)
 - Căn (sqrt)
 - Trị tuyệt đối (abs)
 - ...
- Các phương thức trong lớp Math là những phương thức tĩnh (static methods). Vì vậy, ta gọi trực tiếp thông qua tên lớp mà không cần tạo đối tượng.

```
value = Math.sqrt(delta);
```

Lớp bao kiểu dữ liệu cơ sở (Wrapper Class)

- Trong gói `java.lang` có chứa các lớp wrapper tương ứng cho từng kiểu dữ liệu cơ sở.

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

Wrapper Class

- Trong các lớp wrapper còn chứa các phương thức tĩnh để thao tác trên từng kiểu dữ liệu.
- Ví dụ trong lớp Integer có chứa một phương thức để chuyển từ một chuỗi số nguyên thành giá trị số nguyên.

```
num = Integer.parseInt(str);
```

- Hay chuyển một ký tự của chuỗi word thành ký tự hoa:

```
ch=Character.toUpperCase(st.trim().charAt  
(0));
```

- Ngoài ra, các lớp wrapper còn chứa các hằng số.
- Ví dụ trong lớp Integer chứa hằng MIN_VALUE và
- MAX_VALUE là giá trị nhỏ nhất và lớn nhất của số int.

Xử lý trôi lênh khi dùng Scanner nhập dữ liệu

- Chương trình khi sử dụng **Scanner** nó sẽ không cho bạn nhập vào một chuỗi bằng `next()` hoặc `nextLine()` khi trước đó bạn dùng `nextInt()`, `nextFloat()`, `nextDouble()`,... để nhập vào một số.
- Có 2 cách:
- **Cách 1:** Thêm `in.nextLine()` vào sau `in.nextInt()`,....
- **Cách 2:**

```
System.out.print("Nhập tên: ");  
String name = in.nextLine();  
System.out.print("Nhập tuổi: ");  
int age = Integer.parseInt(in.nextLine());  
System.out.print("Nhập địa chỉ: ");  
String address = in.nextLine();
```

Regular Expression trong Java

- Java cung cấp `java.util.regex` package cho pattern so khớp với các Regular Expression.
- Gói **`java.util.regex`** chủ yếu chứa 3 lớp sau:
 - **Lớp Pattern:** Một đối tượng Pattern là một phép biểu diễn được biên dịch của một Regular Expression.
 - **Lớp Matcher:** Một đối tượng Matcher là phương tiện mà thông dịch pattern và thực hiện so khớp các hoạt động với một chuỗi đầu vào.
 - **PatternSyntaxException:** Một đối tượng PatternSyntaxException là một exception (ngoại lệ)

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match
gi	Perform a global case-insensitive match
^	Get a match at the beginning of a string
\$	Get a match at the end of a string
[xyz]	Find any character in the specified character set
[^xyz]	Find any character not in the specified character set
\w	Find any Alphanumeric character including the underscore
\d	Find any single digit
\s	Find any single space character
?	Find zero or one occurrence of the regular expression
*	Find zero or more occurrence of the regular expression
+	Find one or more occurrence of the regular expression
()	Find the group of characters inside the parentheses & stores the matches string
X{n}	Matches any string that contains a sequence of <i>n</i> X's
X{n,m}	Matches any string that contains a sequence of X to Y <i>n</i> 's

"^\\((?\\d{3}\\)?)?-?\\s*\\d{3}\\s*-?\\d{4}\$"

```
public class MatchPhoneNumber {  
    public static boolean isPhoneValid(String  
    phone) {  
        boolean retval = false;  
        String regex =  
        "^\\(\\d{3}\\) ?-?\\s*\\d{3}\\s*-?\\d{4}$";  
        retval = phone.matches(regex);  
        if (retval)  
            System.out.println("MATCH "+phone + "\\n");  
        return retval; }  
    public static void main(String args[]) {  
        isPhoneValid("(234)- 765 -8765");  
        isPhoneValid("999-585-4009");  
        isPhoneValid("1-585-4009");    }  
}
```

```
public class MatchId {  
    public static boolean isId(String id) {  
        boolean retval = false;  
        String regex =  
            "^\\[Bb]{1}\\d{2}\\w{4}\\d{3}$"  
        retval = id.matches(regex);  
        if (retval)  
            System.out.println("MATCH "+id+ "\n");  
        return retval; }  
    public static void main(String args[]) {  
        isPhoneValid("B13DCCN765");  
        isPhoneValid(" B13DCCN8584");  
        isPhoneValid("b12dcat321");    }  
}
```

Xóa khoảng trống

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class Main {
    public static String removeWhitespace(String s) {
        String reg = "\\s+";
        String sp = " ";
        Pattern pattern = Pattern.compile(reg);
        Matcher matcher = pattern.matcher(s);
        return matcher.replaceAll(sp);
    }
    public static void main(String[] args) throws
        Exception {
        String st = "This is a   Java   program. This is
        another Java Program.";
        String result=removeWhitespace(st);
        System.out.println(result); }
}
```

Mảng (array)

- Tập hợp hữu hạn các phần tử cùng kiểu
- Phải được khai báo trước khi sử dụng
- Khai báo:
 - Cú pháp:

```
kieu_dl[] ten_mang = new kieu_dl[KT_MANG];  
kieu_dl ten_mang[] = new kieu_dl[KT_MANG];
```

- Ví dụ:

```
char c[] = new char[12];
```

Khởi tạo mảng

- Khai báo, khởi tạo giá trị ban đầu:

- Cú pháp:

```
kieu_dl[] ten_mang = {ds_gia_tri_cac_ptu};
```

- Ví dụ:

```
int[] number = {10, 9, 8, 7, 6};
```

- Nếu không khởi tạo, nhận giá trị mặc định tùy thuộc vào kiểu dữ liệu.
- Luôn bắt đầu từ phần tử có chỉ số 0

Ví dụ - mảng

Tên của mảng (tất cả các thành phần trong mảng có cùng tên, c)

c.length: cho biết độ dài của mảng c

Chỉ số (truy nhập đến các thành phần của mảng thông qua chỉ số)

c[0]

c[1]

c[2]

c[3]

c[4]

c[5]

c[6]

c[7]

c[8]

c[9]

c[10]

c[11]

-45

6

0

72

1543

-89

0

62

-3

1

6453

78

Khai báo và khởi tạo mảng

- Ví dụ:

- `int MAX = 5;`

- `boolean bit[] = new boolean[MAX];`

- `float[] value = new float[2*3];`

- `int[] number = {10, 9, 8, 7, 6};`

- `// prints "false"`

- `System.out.println(bit[0]);`

- `// prints "0.0"`

- `System.out.println(value[3]);`

- `// prints "9"`

- `System.out.println(number[1]);`

Ví dụ mảng 1 chiều

```
public class DaySo {  
    int calSum(int... a) {  
        int t=0;  
        for(int x:a)  
            t+=x;  
        return t;  
    }  
    int calMin(int... a) {  
        int t=a[0];  
        for(int x:a)  
            if(t>x)  
                t=x;  
        return t;  
    }  
}
```

```
int calMax(int... a) {  
    int t=a[0];  
    for(int x:a)  
        if(t<x)  
            t=x;  
    return t;  
}  
  
int[] sort(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;            }  
    return a;  
}
```

```
int[] input(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();  
    }  
    return b; }  
  
void toString(int... a) {  
    System.out.print("\n Day so:  
"+Arrays.toString(a));  
}
```

Mảng nhiều chiều

Bảng với các dòng và cột

- Thường sử dụng mảng hai chiều
- Ví dụ khai báo mảng hai chiều `b[2][2]`
- `int a[][] = { { 1, 2, 7 }, { 3, 4, 90 } };`
 - 1 và 2 được khởi tạo cho `b[0][0]` và `b[0][1]`
 - 3 và 4 được khởi tạo cho `b[1][0]` và `b[1][1]`
- `int b[3][4];`
- `int row = b.length; // số hàng`
- `int col = b[0].length; // số cột`

Cộng 2 mảng

```
public double[][] add(double[][] a,  
    double[][] 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];  
    return c;  
}
```

Trừ 2 mảng

```
public double[][] subtract(double[][] a,  
    double[][] 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];  
    return c;  
}
```

Tích 2 mảng

```
public double[][] multiply(double[][] a,  
    double[][] b) {  
    int m1 = a.length;  
    int n1 = a[0].length;  
    int m2 = b.length;  
    int n2 = b[0].length;  
    if (n1 != m2) throw new  
        RuntimeException("Illegal matrix dimensions.");  
    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];  
    return c;  
}
```

Chuyển vị

```
public double[][]  
    transpose(double[][] a) {  
    int m = a.length;  
    int n = a[0].length;  
    double[][] b = new double[n][m];  
    for (int i = 0; i < m; i++)  
        for (int j = 0; j < n; j++)  
            b[j][i] = a[i][j];  
    return b;  
}
```