



CƠ SỞ LẬP TRÌNH JAVA (JAVA FUNDAMENTAL)



CƠ SỞ LẬP TRÌNH JAVA

- Hệ thống từ khoá của JAVA (Java keywords)
- Các kiểu dữ liệu, kiểu dữ liệu nguyên thuỷ (Data types, Primitive types)
- Biến và hằng (Variables & Constants)
- Chuyển kiểu dữ liệu (Type conversions).
- Các loại hằng (Constant types)
- Toán tử (Java Operators).
- Biểu thức (Java expression).
- Biểu thức Boolean (Boolean expression).
- Vào ra cơ bản (Basic IO).
- Các cấu trúc điều khiển (Basic flow control).
- Xử lý Mảng (Arrays)
- Xử lý Chuỗi ký tự (Strings)

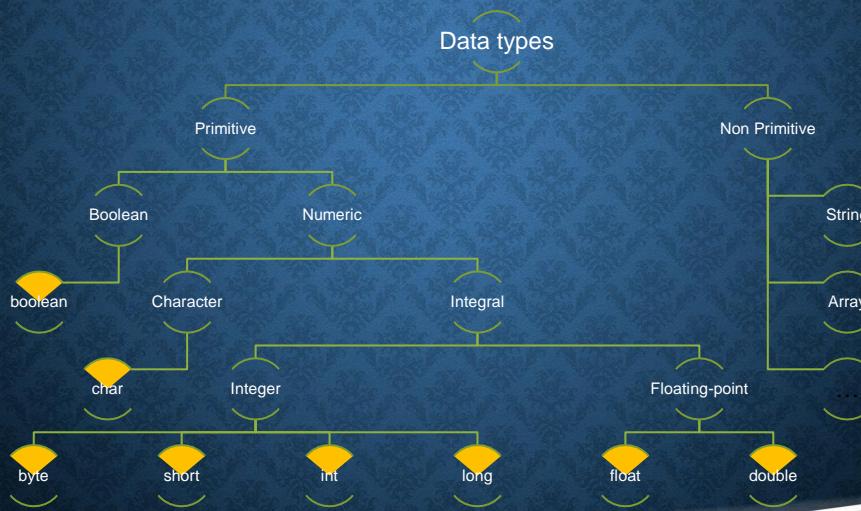


1. HỆ THỐNG TỪ KHOÁ CỦA JAVA (KEYWORDS)

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



2. CÁC KIỂU DỮ LIỆU (DATA TYPES)





2.1 CÁC KIỂU DỮ LIỆU NGUYÊN THỦY (PRIMITIVE DATA TYPES)

Kiểu dữ liệu	Kích thước	Khoảng giá trị	Wrapper
byte	1 byte	-128 → 127	Byte
short	2 byte	-32,768 → 32,767	Short
int	4 byte	-2,147,483,648 → 2,147,483,647 (2^32-1)	Integer
long	8 byte	±9,223,372,036,854,775,808(7) (2^64-1)	Long
float	4 byte	±3.40292347E+38	Float
double	8 byte	±1.7976931348623157E+308	Double
char	2 byte (unicode)	\u0000 → \uffff	Character
boolean	1 bit	true or false	Boolean

Từ phiên bản JDK7, giá trị số có thể được biểu diễn bởi ký tự '_'. VD: 10_000_000



2.2 VÍ DỤ KIỂU DỮ LIỆU NGUYÊN THỦY

- Chuyển đổi kiểu nguyên thủy sang kiểu đối tượng Wrapper
 - Sử dụng constructor: `Integer object = new Integer(10);`
 - Sử dụng phương thức: `Integer anotherObject = Integer.valueOf(10);`
- Chuyển đổi Wrapper trở về nguyên thủy
 - `Integer object = new Integer(10);`
 - `int val = object.intValue(); //wrapper to primitive`
- Từ Java 5.0
 - Autoboxing: `Character ch = 'a';`
 - Auto-Unboxing:

```
int sum = 0;
for (Integer i: integerList) { if (i % 2 == 0) sum += i; }
return sum;
```



3. BIẾN VÀ HẰNG (VARIABLES & CONSTANTS)

- Chương trình sử dụng bộ nhớ cho việc lưu trữ và đọc dữ liệu thông qua việc sử dụng các biến và hằng.
- Biến và hằng là một vùng bộ nhớ được xác định trước dành riêng cho chương trình khi hoạt động.
- Chỉ có biến mới có thể thay đổi giá trị của nó khi chương trình hoạt động.
- Khi một biến mà không thay đổi giá trị trong toàn bộ thời gian hoạt động của chương trình thì nên khai báo nó là hằng.



3. BIẾN VÀ HẰNG (TIẾP)

- Mỗi một biến có một tên, một kiểu dữ liệu.
- Các loại biến:
 - Biến khởi tạo đơn thuần (Instance variables)
 - Biến không thay đổi giá trị (Static variables)
 - Biến cục bộ (Local variables)
 - Các tham số của phương thức (Method arguments)
- Quy ước đặt tên (cho cả hằng):
 - Bắt đầu bằng ký tự chữ, gạch dưới “_”, dấu \$.
 - Không có dấu cách.
 - Sau ký tự đầu có thể dùng các ký tự số, gạch dưới, dấu &.
 - Không trùng với từ khoá.
- Ví dụ:
 - _abc, A_bc, a123bc, A_1bc → đúng
 - 1abc, ab c, A1/bc, → sai



3. BIẾN VÀ HÀNG - KHAI BÁO BIẾN

- Cấu trúc khai báo

<Kiểu dữ liệu> <Tên biến> [=<Giá trị>];

- Ví dụ:

- char ch;
- int x, y, m = 10_100;//jdk7 trả về
- long n = 20l;
- float f = 1.2f;
- double d = 4.12;
- boolean b1 = (3>10);



4. CHUYỂN KIỂU DỮ LIỆU (TYPE CONVERSIONS)

- Chuyển kiểu hẹp (Narrowing):

- int i; float f1=3.2f; **i = (int)f1;**
- long l1; double d1=1.234d; **l1=(long)d1;**

➔ Chuyển kiểu hẹp sẽ mất thông tin về độ rộng cũng như độ chính xác biến số.

- Chuyển kiểu rộng (Widening):

- int i=5; float f1; **f1=i;**
- byte b1=23; double d1; **d1=b1;**

➔ Chuyển kiểu rộng không mất thông tin nhưng độ chính xác cũng bị ảnh hưởng.

- Chuyển kiểu rộng an toàn hơn về mặt giá trị tuyệt đối.



VÍ DỤ CHUYÊN KIỀU

```
public class eg1{
    public static void main(String args[]){
        int i = 5; float f;
        f = i;
        System.out.println("i="+i+", f="+f);
        f = 5.86F; i = (int)f;
        System.out.println("i="+i+", f="+f);
        System.out.println(6/4);
        System.out.println(6/(float)4);
        System.out.println((float)6/4);
        System.out.println(6.0/4);
    }
}
```



5. CÁC LOẠI HÀNG (CONSTANT TYPES)

- Cấu trúc khai báo

final <Kiểu dữ liệu> <TÊN HÀNG> = <Giá trị>;

- Phân loại:

- Hằng số nguyên (Integer constants).
- Hằng số thực (Real number constants).
- Hằng ký tự (Character constants).
- Hằng chuỗi ký tự (String constants).
- Hằng boolean (Boolean constants)



5.1 HÀNG SỐ NGUYÊN

- Số nguyên có thể ở:

- hệ 10 (decimal), (0-9)
- hệ 16 (hexadecimal), (0-9,A,B,C,D,E,F)
- hệ 8 (octal), (0-7)...

- Quy ước viết:

- Đổi với hệ 10: 16, 25, ...
- Đổi với hệ 16: 0x10 hoặc 0X10, 0x19,...
- Đổi với hệ 8: 020, 031,...

- Khai báo:

- final int X=16;
- final int Y=0x10;
- final int Z=020;



5.2 HÀNG SỐ THỰC

- Quy ước viết:

- Số thực kiểu float, có f hoặc F sau cùng:
 - Ví dụ: 0.5f; 1,234.3F; ...
- Số thực kiểu double, có d hoặc D sau cùng:
 - Ví dụ: 0.5d; 1,234D; 4.12d; ...
- Số thực khoa học, có e hoặc E và số mũ hệ 10:
 - Ví dụ: 1.2343e+3f; 2.123E-2D;...

- Khai báo:

- final float F=3.2f , F1=9.123e+2F; → đúng
- final double D1=1.2 , D2=3.123f; → sai



5.3 HÀNG KÝ TỰ

- | | |
|---|--|
| <ul style="list-style-type: none"> • Các loại ký tự đặc biệt: <ul style="list-style-type: none"> • \n – xuống dòng • \t – tab ngang • \b – xoá lùi • \r – enter • \" – nháy kép • \' – nháy đơn • \\ – gạch \ • \f – đầy trang • \uxxxx – ký tự unicode. | <ul style="list-style-type: none"> • Biểu diễn bởi " • Ví dụ: 'A'; 'j';... • 'a' khác với "a". • Khai báo: <ul style="list-style-type: none"> • final char CH='a'; • final char CH1='\n'; |
|---|--|



5.4 HÀNG CHUỖI KÝ TỰ

- Chuỗi ký tự không phải là kiểu dữ liệu nguyên thuỷ của JAVA.
- Viết trong ""
- Khai báo:
 - final String STR1="Hello world";
 - final String STR2="column1\tcolumn2";
 - final String STR3="";



5.5 HÀNG BOOLEAN

- Có hai giá trị: true và false.
- Có thể lấy giá trị từ các phép toán so sánh hay kết quả biểu thức,...
- Khai báo:
 - final boolean BL1=true;
 - final boolean BL2= true && false;
 - final boolean BL3=(2>5)||(4<7); //→true
 - final boolean BL4=(2>5)&&(4<7);//→false



6. TOÁN TỬ (JAVA OPERATORS)

- Toán tử cơ bản.
- Toán tử kết hợp.
- Tiền toán tử và hậu toán tử.
- Toán tử điều kiện
- Một số toán tử mở rộng

<https://howtodoinjava.com/java/basics/operators-in-java/>



6.1 TOÁN TỬ CƠ BẢN (ARITHMETIC OPERATORS)

- Một số toán tử cơ bản: +, -, *, /, %
- Sử dụng:
 - $a=b+c;$
 - $x=c\%b; a=23\%4 \rightarrow a=3$
 - $a=c/b; x=21/5 \rightarrow x=4$
- Thứ tự:
 - *, /, %
 - +, -
- Ví dụ: $a=(5+3)/4+2*(1+2) = 8$



VÍ DỤ TOÁN TỬ CƠ BẢN

```
public class eg2{
    public static void main(String args[]) {
        int num1, num2;
        float fnum1, fnum2;
        num1 = 22/3; fnum1 = 22/3;
        System.out.println("num1=" + num1);
        System.out.println("fnum1=" + fnum1);
        num2 = 22 % 3; fnum2 = 22 % 3;
        System.out.println("num2=" + num2);
        System.out.println("fnum2=" + fnum2);
        System.out.println("Có thể thực hiện= "+22.3%3);
    }
}
```



6.2 CÁC TOÁN TỬ KẾT HỢP (ASSIGNMENT OPERATORS)

- Các toán tử kết hợp: $+=$, $-=$, $/=$, $\%=$, $*=$
- Dạng toán tử đôi, sử dụng:
 - $a+=1 \rightarrow a=a+1$
 - $b\%=2 \rightarrow b=b\%2$



6.3 TIỀN TOÁN TỬ - HẬU TOÁN TỬ

- Các toán tử: $++$, $--$
- Sử dụng:
 - $i-- \rightarrow i=i-1$ ($i-=1$)
 - $--i \rightarrow i=i-1$ ($i-=1$)
 - $++i \rightarrow i=i+1$ ($i+=1$)
 - $i++ \rightarrow i=i+1$ ($i+=1$)
- Tiền toán tử sẽ thực hiện phép toán trước, hậu toán tử thực hiện phép toán sau.
- VD:
 - $i=1; j=++i+1 \rightarrow j=3, i=2$
 - $i=4; j=i--+1 \rightarrow j=5, i=3$



VÍ DỤ TIỀN TOÁN TỪ – HẬU TOÁN TỪ

```
public class eg3 {
    public static void main(String args[]) {
        int i = 5, j = 10;
        int k=0;
        System.out.println("i="+i+", j="+j+", k="+k);
        ++i; j++;
        System.out.println("i="+i+", j="+j+", k="+k);
        k=++i + j++;
        System.out.println("i="+i+", j="+j+", k="+k);
        k=--k;
        System.out.println("i="+i+", j="+j+", k="+k);
    }
}
```



6.4 TOÁN TỪ ĐIỀU KIỆN (TERNARY OR CONDITIONAL OPERATORS)

- Cú pháp:

<Điều kiện>?<Giá trị 1>:<Giá trị 2>

- Điều kiện đúng → biểu thức nhận giá trị 1.
- Ngược lại → biểu thức nhận giá trị 2.

- Ví dụ:

- int m=3, n=8;*
- int x;*
- x=(m>=n)?11.5;*



6.5 MỘT SỐ TOÁN TỬ MỞ RỘNG (BITWISE OPERATORS)

Toán tử	Dạng kết hợp	Ý nghĩa	Ví dụ
<<	<<=	Dịch chuyển bít sang trái	$a=4<<2 \rightarrow a=16$
>>	>>=	Dịch chuyển bít sang phải	$a=4>>2 \rightarrow a=1$
>>>	>>>=	Dịch chuyển bít sang phải và điền vào trước đó các số 0	$a=4>>>2 \rightarrow a=1$



6.6 KIỂU LIỆT KÊ DỮ LIỆU

- Kiểu liệt kê dữ liệu là một tập cố định các thành phần hằng trong JAVA, sử dụng từ khóa enum.
- Phiên bản JDK1.5.
- Cấu trúc:
 - enum <tên kiểu liệt kê> { <CÁC GIÁ TRỊ> };
 - Ví dụ:
 - enum Size { SMALL, MEDIUM, LARGE, EXTRA_LARGE };
 - Tên kiểu liệt kê dạng như lớp đối tượng, các giá trị dạng như các HÀNG.
- Khai báo và sử dụng:
 - Size s = Size.MEDIUM;
 - Mỗi một enum là một thể hiện kiểu liệt kê của bản thân nó



VÍ DỤ KIỂU LIỆT KÊ DỮ LIỆU – NON CONSTRUCTOR

- Khai báo

```
public enum Direction {
    EAST, WEST, NORTH, SOUTH;
}
```

- JVM sẽ chuyển đổi thành cấu trúc sau

```
final class Direction extends Enum<Direction> {
    public final static Direction EAST = new Direction();
    public final static Direction WEST = new Direction();
    public final static Direction NORTH = new Direction();
    public final static Direction SOUTH = new Direction();
}
```

- Sử dụng

```
public class EnumExample {
    public static void main(String[] args) {
        Direction north = Direction.NORTH;
        System.out.println(north); //NORTH
        Direction.EAST.ordinal(); //0
        Direction.NORTH.ordinal(); //2
        Direction[] directions = Direction.values();
        for (Direction d : directions) {
            System.out.println(d); //EAST, WEST, NORTH,
        }
        Direction east = Direction.valueOf("EAST");
        System.out.println(east);
    }
}
```

<https://howtodoinjava.com/java/enum/enum-tutorial/>



VÍ DỤ KIỂU LIỆT KÊ DỮ LIỆU - CONSTRUCTOR

- Khai báo

```
public enum Direction {
    // enum fields
    EAST(0), WEST(180), NORTH(90), SOUTH(270);
    // constructor
    private Direction(final int angle) {
        this.angle = angle;
    }
    // internal state
    private int angle;
    public int getAngle() {
        return angle;
    }
}
```

- Sử dụng

```
Direction north = Direction.NORTH;
System.out.println( north ); //NORTH
System.out.println( north.getAngle() ); //90
System.out.println( Direction.NORTH.getAngle() ); //90
```



VÍ DỤ KIỀU LIỆT KÊ DỮ LIỆU – PHƯƠNG THỨC TRONG ENUM

Phương thức tường minh

```
public enum Direction {
    // enum fields
    EAST, WEST, NORTH, SOUTH;

    protected String printDirection() {
        String message = "You are moving in " + this
+ " direction";
        System.out.println( message );
        return message;
    }
}
```

Phương thức trừu tượng

```
public enum Direction {
    // enum fields
    EAST {
        @Override
        public String printDirection() {
            String message = "ĐÔNG"; return message;
        }
    },
    WEST {
        ...
    },
    NORTH {
        ...
    },
    SOUTH {
        ...
    };
    public abstract String printDirection();
}
```



6.6 KIỀU LIỆT KÊ DỮ LIỆU – LƯU Ý

- Các liệt kê là lớp con của **java.lang.Enum**
- Nếu một enum là thành viên của một lớp, thì enum đó hoàn toàn là static
- Không sử dụng từ khóa **new** để khởi tạo enum
- Các phương thức **name()** và **valueOf()** sử dụng văn bản của các hằng liệt kê, trong khi **toString()** có thể ghi đè để cung cấp các nội dung khác, nếu muốn
- Với enum thì phương thức **equal()** và **==** cho kết quả như nhau
- Các hằng liệt kê phải là **public static final**
- Thứ tự thể hiện trong danh sách các hằng là thứ tự tự nhiên của các hằng đó.
- Các **constructor** trong enum nên khai báo với **private**, vì tránh sử dụng **new** để tạo enum.
- Enum có thể được sử dụng trong **switch**



7. BIỂU THỨC (JAVA EXPRESSIONS)

- Biểu thức thể hiện cách tính toán giá trị.
- Biểu thức bao gồm 1 hay nhiều toán hạng được kết hợp bởi các toán tử.
 - $a=b+c*d;$
 - $m=2^g * i * tri(n);$
- Biểu thức có thể bao gồm các biểu thức con, nên xác nhận bởi các dấu ngoặc đơn.
 - $x=(a+2)/4 + c*(b-a);$
- Biểu thức còn được dùng để gán giá trị cho biến.
 - $y=8;$
 - $x=6+(z=5);$



8. BIỂU THỨC BOOLEAN

Các toán tử logic (logical operators):

Toán tử	Mô tả
&	Phép và (and) trên bít
&&	Phép và (and) trên toán hạng
	Phép hoặc (or) trên bít
	Phép hoặc (or) trên toán hạng
!	Phép phủ định (not)
^	Phép xor



8. BIẾU THỨC BOOLEAN (TIẾP)

Các toán tử quan hệ (relational operators):

Toán tử	Mô tả
!=	So sánh khác nhau
==	So sánh bằng nhau
>	So sánh lớn hơn
<	So sánh nhỏ hơn
>=	So sánh lớn hơn hoặc bằng
<=	So sánh nhỏ hơn hoặc bằng



VÍ DỤ VỚI BIẾU THỨC BOOLEAN

```
public class eg4{
    public static void main(String args[]) {
        int i=12; float f=3.2f;
        int m=6, n=4;
        char ch='a';
        System.out.println(i>(int)f);
        System.out.println((int)f <= ch);
        System.out.println((i>(int)ch) && ('A'<=ch));
        System.out.println(m&n);
    }
}
```



9. VÀO RA CƠ BẢN

- Đối với việc nhập dữ liệu, JAVA không cung cấp sẵn các phương thức mà chỉ cung cấp đối tượng *in* thuộc lớp *System* để người lập trình có thể dựa vào đó xây dựng các phương thức của riêng mình.
- Tuỳ vào phiên bản của JDK mà sẽ có những cách xử lý khác nhau.
- Việc xây dựng các phương thức đầu vào phụ thuộc vào kiểu dữ liệu mà người lập trình muốn nhận.



CHƯƠNG TRÌNH ĐỌC MỘT SỐ NGUYỄN TỪ BÀN PHÍM

```
public class inputInt{
    public static void main(String[] args) {
        int n;
        BufferedReader in =
            new BufferedReader(new InputStreamReader(System.in));
        try {
            n = Integer.parseInt(in.readLine());
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.println("Số vừa nhập "+n);
    }
}
```



XÂY DỰNG CHƯƠNG TRÌNH THEO PHIÊN BẢN JDK1.5 TRỞ ĐI

```
public class inputInt{
    public static void main(String[] args) {
        java.util.Scanner in = new java.util.Scanner(System.in);
        int n = in.nextInt();
        System.out.println("So vua nhap "+n);
    }
}
```



9. VÀO RA CƠ BẢN (TIẾP)

- Đối với yêu tố xuất dữ liệu, JAVA đã cung cấp sẵn các phương thức để người lập trình có thể sử dụng, các phương thức đầy gồm *print*, *println* của đối tượng *out* thuộc lớp *System*.
- Cách sử dụng:

System.out.print[In] ("Thông tin đưa ra ở đây.");

- Người lập trình có thể kết hợp các chuỗi đưa ra bằng cách sử dụng dấu cộng “+”.
- Ví dụ:

```
int Age=28;
System.out.println("Tuoi cua ban la "+Age);
```



9. VÀO RA CƠ BẢN (TIẾP)

- Định dạng dữ liệu xuất theo phương thức printf của **System.out**
- Cấu trúc:

```
System.out.printf("% [argument index] [flag] [width] [.precision] opt", value);
```

- Trong đó:
 - % - biểu thị một hướng dẫn định dạng, bắt buộc.
 - [argument index] : xác định chỉ số của các đối số để được định dạng, tùy chọn.
 - [flag] là một hướng dẫn định dạng viết từ bên phải (+), viết từ bên trái (-), tùy chọn.
 - [width] độ rộng tối thiểu các ký tự in ra cho đối số đó.
 - [.precision] xác định độ chính xác sau dấu phân cách thập phân.
 - opt một số tùy chọn khác.
- Ví dụ:

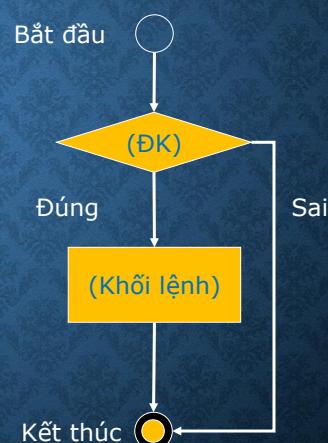
```
double x = 10000.0 / 3.0;
System.out.println(x);
→3333.333333
System.out.printf("%8.2f\n", x);
→3333.33
```



10. CÁU TRÚC ĐIỀU KHIỂN - LẬP

- Lệnh điều khiển if
- Dạng:


```
if (<Điều kiện> {
    //khối lệnh
}
```
- Cặp {} tương tự C/C++.
- Dạng if đơn giản





VÍ DỤ VỚI CÂU LỆNH IF

```
public class egIf1 {
    public static void main(String args[]) {
        int a = (int)(Math.random()*100);
        int b = (int)(Math.random()*100);
        if(a>b) {
            System.out.println("a is larger than b");
        }
        if(a<b) {
            System.out.println("b is larger than a");
        }
        if(a==b) {
            System.out.println("a equals b");
        }
    }
}
```

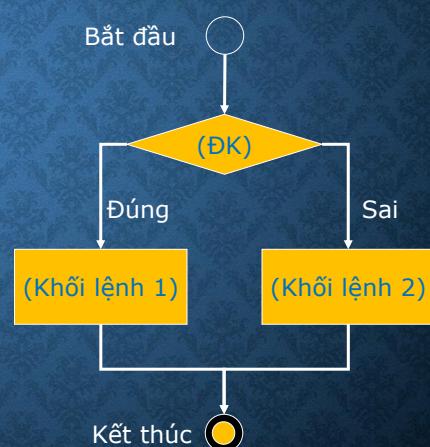


10. CÁU TRÚC ĐIỀU KHIỂN - LẬP (TIẾP)

- Lệnh điều khiển if...else
- Dạng:

```
if (<Điều kiện>) {
    //khối lệnh 1
} else {
    //khối lệnh 2
}
```

- Có thể kết hợp các if...else liên tiếp nhau cho một công việc.





VÍ DỤ CÂU LỆNH IF...ELSE

```
public class egIf2 {
    public static void main(String args[]) {
        int a = (int) (Math.random()*100);
        int b = (int) (Math.random()*100);
        if (a>b) {
            System.out.println("a is larger than b");
        }else {
            System.out.println("b is larger than a");
            System.out.println("OR b is equal to a");
        }
    }
}
```



QUAN HỆ TOÁN TỬ ĐIỀU KIỆN – LỆNH ĐIỀU KHIỂN IF...ELSE

- Toán tử: <điều kiện>?<bt1>:<bt2>
- Lệnh:


```
if (<điều kiện>){
    //khởi lệnh 1
} else{
    //khởi lệnh 2
}
```
- Cùng trạng thái lựa chọn, thành phần giá trị tương ứng.



VÍ DỤ CHUYÊN ĐỔI CÁCH VIẾT

```

public class CondOper{
    public static void main(String args[]){
        char ch = '0';
        System.out.println((ch>='0' && ch <='9') ? "là số":"không phải số");
    }
}

public class CondOper {
    public static void main(String args[]) {
        char ch = '0';
        if (ch >= '0' && ch <= '9'){System.out.println("là số");}
        else {System.out.println("không phải số");}
    }
}

```



10. CẤU TRÚC ĐIỀU KHIỂN - LẬP (TIẾP)

- Lệnh điều khiển switch - nhiều khả năng lựa chọn.
- Dạng:

```

switch ( <bíểu thức> ) {
    case <GT1>: //khối lệnh 1;
        break;//lệnh thoát
    ...
    case <GTn>: //khối lệnh n;
        break;
    default: //khối lệnh ngầm định;
        break;
}

```

- Mô tả: là dạng if lồng nhau, nếu biểu thức đúng với 1 trong các trường hợp (case) thì khối lệnh tương ứng được thực hiện, giá trị của case không được trùng.
- Sử dụng từ khoá break → ngừng thực hiện khi hết khối lệnh. Thành phần default của toàn switch.
- Switch làm việc với các kiểu nguyên thủy, kiểu đối tượng Wrapper, enum (java 5), String (java 7).



VÍ DỤ VỚI CÂU LỆNH SWITCH

```

public class egSwitch {
    public static void main(String args[]) {
        int n = new java.util.Scanner(System.in).nextInt();
        switch(n) {
            case 1: System.out.println("One");break;
            case 2: System.out.println("Two");break;
            case 3: System.out.println("Three");break;
            case 4: System.out.println("Four");break;
            default: System.out.println("default");break;
        }
        System.out.println("End of Switch");
    }
}

```



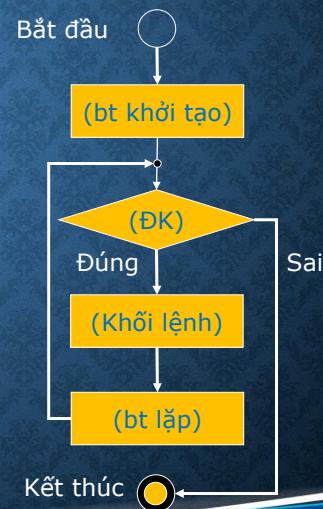
10. CÁU TRÚC ĐIỀU KHIỂN - LẶP (TIẾP)

- Vòng lặp for - lặp xác định.
- Dạng:

```

for (<biểu thức khởi tạo>; <điều
    kiện lặp>; <biểu thức lặp>) {
    //khởi lệnh
}
    
```

- Thường biểu thức lặp là bước nhảy biến số.
- Các biểu thức và điều kiện có thể có nhiều hơn 1.
- Có thể bỏ trống cả 3 thành phần trên.





VÍ DỤ VỚI VÒNG LẬP FOR

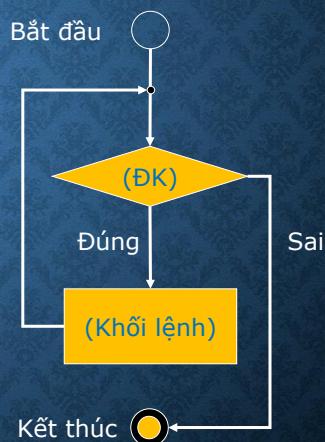
```
public class egFor {
    public static void main(String args[]) {
        int num, total;
        for (num = 1; num <= 5; num++) {
            System.out.println("Num = " + num);
        }
        for (num = 0; num <= 100; num += 10) {
            System.out.println("Num = " + num);
        }
        for (num = 1, total = 0; num <= 10; total += num, ++num) {
            System.out.println("Num = "+num+", total = "+total);
        }
        for (num = 10; num != 0; num -= 3) {
            System.out.println("Num = "+ num);
        }
    }
}
```



10. CÁU TRÚC ĐIỀU KHIỂN - LẬP (TIẾP)

- Vòng lặp while - lặp không xác định.
- Dạng:


```
while (<Điều kiện>){
    //khởi lệnh
}
```
- Phải có câu lệnh để thay đổi biểu thức điều kiện hoặc để thoát khỏi while.
- Chú ý với {}.





VÍ DỤ VỚI VÒNG LẶP WHILE

```
public class egWhile {
    public static void main(String args[]) {
        int num;
        num = 1;
        while (num <= 5) {
            System.out.println("Num = " + num);
            num++; //lệnh thay đổi giá trị num → điều kiện
        }
    }
}
```



10. CÁU TRÚC ĐIỀU KHIỂN - LẶP (TIẾP)

- Vòng lặp do...while - lặp không xác định.
- Dạng:


```
do {
    //khởi lệnh
}while(<Điều kiện>);
```
- Tương tự while, cần có câu lệnh để thay đổi biểu thức điều kiện, để kết thúc lặp
- Có ít nhất 1 lần thực hiện khối lệnh.





VÍ DỤ VỚI VÒNG LẶP DO...WHILE

```
public class egDoWhile {
    public static void main(String args[]) {
        int num , digit = 0;
        System.out.println("Nhập vào một số :");
        num = new java.util.Scanner(System.in).nextInt();
        do {
            digit = num % 10; num /= 10;
            System.out.println(digit);
        } while (num != 0);
    }
}
```



11. XỬ LÝ MẢNG

- Giới thiệu mảng (array).
- Mảng một chiều
 - Khai báo và sử dụng.
 - Các thao tác trên dữ liệu.
 - Tìm kiếm một giá trị.
 - Tìm giá trị lớn nhất, nhỏ nhất
 - Sắp xếp tăng dần, giảm dần theo giá trị...
- Mảng đa chiều (>=2 chiều)
- Mảng đối tượng
- Duyệt mảng từ phiên bản Java 8



11.1 GIỚI THIỆU MẢNG

- Mảng (array) được dùng để lưu trữ các thành phần dữ liệu cùng kiểu, liên tiếp nhau trong bộ nhớ.
- Các thành phần trong mảng được xác định bởi tên mảng (array name) và chỉ số (array index number).
- Quá trình khai báo mảng, tùy thuộc vào kiểu dữ liệu và số lượng phần tử tối đa mà ta có kích thước sử dụng bộ nhớ của mảng.
- VD: một mảng tối đa 30 phần tử có kiểu dữ liệu là số nguyên (int), thì kích thước bộ nhớ sử dụng $\approx 30 \times 4$ byte = 120 byte.
- Kích thước khi đã được khai báo thì không thể thay đổi lại, ngoại trừ thành phần cấp phát động.
- Chỉ số của mảng bắt đầu từ 0, phân biệt với số thứ tự phần tử mảng.



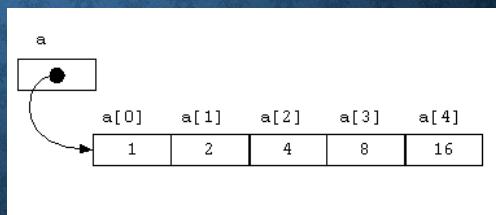
11.2 MẢNG MỘT CHIỀU - KHAI BÁO VÀ SỬ DỤNG

- Dạng khai báo mảng 1 chiều:

```
<kiểu dữ liệu>[] <tên mảng> = new <kiểu dữ liệu>[<kích thước>];
<kiểu dữ liệu> <tên mảng>[] = new <kiểu dữ liệu>[<kích thước>];
```

- Ví dụ:

- int[] a = new int[5], a2;
- int a[] = new int[5], a2;





11.2 MẢNG MỘT CHIỀU - KHAI BÁO VÀ SỬ DỤNG (TIẾP)

- Một số chú ý:

- Cặp ngoặc []
- Mảng có thể không được xác định kích thước khi khai báo. VD:

```
String[] str;
```

- Mảng có thể nhận trực tiếp các giá trị. VD

```
int[] arrInt1 = {2,45,9,13,4,81};  
String[] arrStr = {"a", "ab", "abc", "abcd"};
```

- Truy cập:

- < tên mảng >[chỉ số] (Chỉ số < Kích thước mảng)
- VD: arrInt1[3] → 13, arrStr[1] → ab



11.2 MẢNG MỘT CHIỀU - KHAI BÁO VÀ SỬ DỤNG (TIẾP)

- Một số thuộc tính, phương thức làm việc với mảng
 - length → lấy về độ dài (kích thước) mảng
 - clone() → tạo bản sao cho một mảng
- Ví dụ:

```
int[] arrInt1=new int[10], arrInt2;  
for(int i=0;i<arrInt1.length;i++){  
    arrInt1[i]=i+1;  
}  
arrInt2 = arrInt1.clone();  
for(int value:arrInt2){ //JDK5  
    System.out.println(value);  
}
```



11.2 MÀNG MỘT CHIỀU – THAO TÁC DỮ LIỆU

- Tìm kiếm một giá trị trong mảng

```
int key = new java.util.Scanner(System.in).nextInt();
boolean flag = false;
for(int value:arrInt){
    if(value==key){flag=true; break;}
}
if(flag){
    System.out.print("Đã tìm thấy giá trị nhập vào trong mảng");
} else{
    System.out.print("Không tìm thấy giá trị này");
}
```



11.2 MÀNG MỘT CHIỀU – THAO TÁC DỮ LIỆU

- Tìm giá trị lớn nhất (nhỏ nhất)

```
int max = arrInt[0]; //min cũng tương tự
for(int i=1;i<arrInt.length;i++){
    if(max < arrInt[i]){
        max = arrInt[i];
    }
}
```

- Chú ý:

- Ban đầu, giá trị được gán bởi phần tử đầu tiên của mảng
- Duyệt từ phần tử thứ 2 của mảng (chỉ số = 1)



11.2 MÀNG MỘT CHIỀU – THAO TÁC DỮ LIỆU

- Sắp xếp tăng dần (giảm dần)

```
int tmp;

for(int i=0;i<arrInt.length-1;i++){
    for(int j=i+1; j<arrInt.length; j++){
        if(arrInt[i]>arrInt[j]){
            tmp = arrInt[i];
            arrInt[i] = arrInt[j];
            arrInt[j] = tmp;
        }
    }
}
```

- Sử dụng Arrays.sort()

```
//Mảng chưa sắp xếp
Integer[] numbers = new Integer[] { 15, 11, 9, 55, 47, 18, 520, 1123, 366, 420 };
//Sắp xếp tăng dần (ngầm định)
Arrays.sort(numbers);
//Sắp xếp giảm dần
Arrays.sort(numbers,
Collections.reverseOrder());
//In ra
System.out.println(Arrays.toString(numbers));
```



11.3 MÀNG ĐA CHIỀU (>=2) - KHAI BÁO VÀ SỬ DỤNG

- Dạng khai báo

```
<kiểu dữ liệu>[][] <tên mảng> = new
<kiểu dữ liệu>[<Kthuoccl>]...[<KthuocN>];
<kiểu dữ liệu> <tên mảng>[][] = new
<kiểu dữ liệu>[<Kthuoccl>]...[<KthuocN>];
```

- Ví dụ:

- int[][] arrInt = new int[3][4];
- float[][] arrFloat = {{2.3, 4.12}, {3.1, 9.27}};

- Truy cập:

<tên mảng>[chỉ số 1]...[chỉ số N]



11.3 MÀNG ĐA CHIỀU (>=2) – MỘT SÓ CHÚ Ý

- Duyệt và xử lý phần tử

```
for(int i=0;i<arrInt.length;i++) {
    for(int j=0;j<arrInt[i].length;j++) { //Xử lý phần tử arrInt[i][j]
    }
}
//JDK5

for(int[] row: arrInt) {
    for(int value: row){//Xử lý giá trị value}
}
```

- Tuỳ theo số chiều của mảng:

- Làm việc theo từng chiều và coi như những chiều khác không đổi
- Thao tác trên dữ liệu thực chất là làm việc trên mảng một chiều.



MỘT SÓ VÍ DỤ KHÁC VỀ MÀNG

- In các phần tử mảng 1 chiều

```
String[] array = new String[] {"First", "Second", "Third", "Fourth"};
System.out.println(array.toString());//Không nên dùng
System.out.println(Arrays.toString(array));
```

- Ghép 2 mảng một chiều thành 2 chiều

```
String[] arr1 = new String[] { "Fifth", "Sixth" };
String[] arr2 = new String[] { "Seventh", "Eighth" };
String[][] arrayOfArray = new String[][] { arr1, arr2 };
System.out.println(arrayOfArray);
System.out.println(Arrays.toString(arrayOfArray));//Không dùng được
System.out.println(Arrays.deepToString(arrayOfArray));
```



MỘT SỐ VÍ DỤ KHÁC VỀ MẢNG (TIẾP)

```
String[] names = {"Alex", "Brian", "Charles", "David"};
```

- Sao chép mảng sử dụng clone()

```
String[] cloneOfNames = names.clone();
```

- Sử dụng Arrays.copyOf() – đọc nhanh nhất

```
String[] copyOfNames = Arrays.copyOf(names, names.length);
```

- Sử dụng System.arraycopy() – đọc chậm nhất

```
String[] copyOfNames2 = new String[names.length];
```

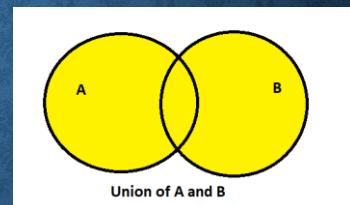
- System.arraycopy(names, 0, copyOfNames2, 0, copyOfNames2.length);



MỘT SỐ VÍ DỤ KHÁC VỀ MẢNG (TIẾP) – HỢP 2 MẢNG

```
Integer[] firstArray = {0,2,4,6,8};
Integer[] secondArray = {1,3,5,7,9};
HashSet<Integer> set = new HashSet<>();
set.addAll(Arrays.asList(firstArray));
set.addAll(Arrays.asList(secondArray));
System.out.println(set); // 0 -> 9

// chuyển đổi sang mảng
Integer[] union = {};
union = set.toArray(union);
System.out.println(Arrays.toString(union)); // 0 -> 9
```



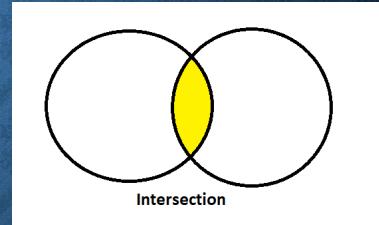


MỘT SỐ VÍ DỤ KHÁC VỀ MÃNG (TIẾP) – GIAO 2 MÃNG

```
Integer[] firstArray = {0,1,2,3,4,5,6,7,8,9};
Integer[] secondArray = {1,3,5,7,9};

HashSet<Integer> set = new HashSet<>();
set.addAll(Arrays.asList(firstArray));
set.retainAll(Arrays.asList(secondArray));
System.out.println(set); //1,3,5,7,9

//convert to array
Integer[] intersection = {};
intersection = set.toArray(intersection);
System.out.println(Arrays.toString(intersection)); //1,3,5,7,9
```



11.4 MÃNG ĐÓI TƯỢNG

- Mảng đối tượng Person

```
Person[] arrPerson = new Person[50];
```

- Gán, đặt giá trị

```
Person aPerson = new Person("EASTER", "BUNNY");
arrPerson[0] = aPerson;
arrPerson[1] = new Person();
arrPerson[1].setFirstName(aPerson.getFirstName());
arrPerson[1].setLastName(aPerson.getLastName());
arrPerson[2] = new Person(aPerson);
```



11.4 MÀNG ĐỔI TƯỢNG (TIẾP)

- Mảng đối tượng lớp học

```
Person[][] studentList = new Person[10][30];
```

- Duyệt và xử lý

```
for(Person[] class:studentList) {
    for(Person student:class) {
        //Xử lý với student
    }
}
```



11.5 DUYỆT MÀNG VỚI JAVA 8

- Phiên bản JDK5

- Cấu trúc: `for(T <Phân tử> : <một tập hợp, một mảng kiểu T>) {...}`
- Ví dụ:

```
int[] numArray = {10, 20, 30, 40};
for(int num : numArray) { System.out.println(num); }
```

- Phiên bản JDK8

```
List<Integer> numList = new ArrayList<Integer>();
numList.add(10);
numList.add(20);
numList.add(30);
numList.add(40);
//lặp với lambda
numList.forEach( item -> System.out.println(item) );
//hoặc tương đương
numList.forEach( System.out::println );
```



12. XỬ LÝ CHUỖI KÝ TỰ

- Một dãy các ký tự và khoảng trắng được xác định bởi cặp nháy kép “”.
- Không phải là kiểu dữ liệu nguyên thuỷ.
- Dạng khai báo và sử dụng

```
String <tên chuỗi> [=<Giá trị khởi tạo>];
```

- VD:

```
String tmp="Đây là chuỗi trung gian";
String strValue = String.valueOf(arrChar)
```

- Xử lý chuỗi là xử lý các ký tự nằm trong chuỗi thông qua các phương thức của đối tượng String cung cấp.



12.1 MỘT SỐ API TRONG XỬ LÝ CHUỖI KÝ TỰ

<code>charAt(int)</code>	Lấy ký tự tại vị trí xác định
<code>endsWith(String)</code>	Kết thúc với chuỗi nào đó
<code>equals(Object)</code>	So sánh với đối tượng xác định
<code>indexOf(int)</code>	Lấy ra vị trí đầu tiên khi gặp ký tự
<code>lastIndexOf(int)</code>	Vị trí sau cùng khi gặp ký tự
<code>length()</code>	Lấy về chiều dài chuỗi
<code>replace(char, char)</code>	Thay thế ký tự với ký tự khác
<code>startsWith(String)</code>	Bắt đầu với chuỗi nào đó
<code>substring(int)</code>	Chuỗi con từ vị trí xác định
<code>trim()</code>	Cắt bỏ khoảng trắng trước, sau.
<code>equalsIgnoreCase(String)</code>	So sánh bỏ qua kiểu chữ



VÍ DỤ 1 - ĐÉM SỐ KÝ TỰ TRONG MỘT CHUỖI

```
public int getCharCount(String str, char ch) {
    int count=0;
    for(int i=0;i<str.length();i++) {
        if(str.charAt(i)==ch) {
            count++;
        }
    }
    return count;
}
```



VÍ DỤ 2 - CHUẨN HÓA MỘT CHUỖI

```
public String getFullscreenFormat(String fullname) {
    String tmp = fullname.trim();
    int blankCount=0;
    for(int i=0; i<tmp.length();i++) {
        if(tmp.charAt(i)==' ') {
            for (int j=i; j < tmp.length(); j++) {
                if(tmp.charAt(j)==' ') {
                    blankCount++;
                }else{break;}
            }
            tmp = tmp.substring(0,i)+ tmp.subSequence(i+blankCount-1,tmp.length());
            blankCount=0;
        }
    }
    return tmp;
}
```



12.2 MỘT SỐ VÍ DỤ KHÁC TRONG XỬ LÝ CHUỖI

- Chuyển một String sang String[]

```
String blogName = "how to do in java";
String[] words = null;
    • Cách 1 sử dụng String.split()
words = blogName.split(" "); // [how, to, do, in, java]
    • Cách 2 sử dụng Pattern.split()
Pattern pattern = Pattern.compile(" ");
words = pattern.split(blogName); // [how, to, do, in, java]
```

- Chuyển String[] về String

```
String newName = String.join(" ", words); // "how to do in java"
```



12.2 MỘT SỐ VÍ DỤ KHÁC TRONG XỬ LÝ CHUỖI (TIẾP)

```
String str = "Cộng hòa xã hội chủ nghĩa Việt Nam";
StringTokenizer defaultTokenizer = new StringTokenizer(str);
System.out.println("Tổng số từ : " + defaultTokenizer.countTokens());
while (defaultTokenizer.hasMoreTokens()) {
    System.out.println(defaultTokenizer.nextToken());
}
System.out.println("Tổng số từ: " + defaultTokenizer.countTokens());
```

Kết quả chạy

- Tổng số từ: 8
- Cộng
- hòa
- ...
- Nam
- Tổng số từ: 0



BÀI TẬP – CƠ SỞ LẬP TRÌNH

- Cài đặt và vẽ lưu đồ thuật toán cho chương trình kiểm tra một số n có phải nguyên tố hay không?
- Cài đặt và vẽ lưu đồ thuật toán cho chương trình giải phương trình bậc 2, có tính tới nghiệm phức
- Viết chương trình tính n!
- Viết chương trình tính $C(m,n)$

$$C(m,n) = \frac{m!}{(m-n)!n!}$$

- Viết chương trình tìm UCLN của 2 số
- Viết chương trình kiểm tra một số có phải là nguyên tố.



BÀI TẬP – MẢNG

- **Mảng 1 chiều:**
 - Viết chương trình tìm kiếm một giá trị (theo thuật toán tìm kiếm nhị phân) trong mảng một chiều.
 - Cho một mảng đã được sắp xếp, viết chương trình bổ sung một giá trị mới vào mảng theo đúng thứ tự.
 - Tìm tất cả các mảng nguyên (int) có tổng giá trị các phần tử bằng giá trị đã cho bất kỳ.
 - Tìm tất cả các số nguyên tố trong một mảng nguyên đặt lên đầu và có sắp xếp.
- **Mảng đa chiều:**
 - Viết chương trình tìm số nguyên tố lớn nhất trong mảng hai chiều.
 - Viết chương trình nhân hai mảng hai chiều
 - Viết chương trình tạo một bản sao của một mảng n chiều, n là bất kỳ.



BÀI TẬP – CHUỖI KÝ TỰ

- Viết chương trình đếm số lượng các ký tự khác nhau có trong một chuỗi.
- Tìm ra những ký tự có tần suất xuất hiện lớn nhất trong một chuỗi. “everybody” → e,y
- Đếm xem trong một chuỗi xuất hiện bao nhiêu từ. “Hello world” → 2
- Viết hoa chỉ các ký tự đầu từ trong một chuỗi. “Nguyễn Văn Abc”
- Viết chương trình so sánh sự giống nhau của hai chuỗi ký tự bất kỳ.
- Viết chương trình cắt ra một số lượng từ nhất định trong một chuỗi đã cho.



End !