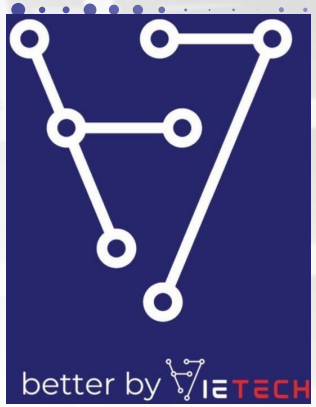


Lập trình Java

Các toán tử và các kiểu dữ liệu cơ bản

Ths. Vũ Duy Khương

Nội dung

1. Tên, biến và hằng
2. Các kiểu dữ liệu cơ sở
3. Biểu thức
4. Các toán tử trong Java
5. Case Studies

Ví dụ lập trình Java

➢ **Ví dụ 2.1**: Tính diện tích hình tròn

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dientichhinhtron;

/**
 *
 * @author KhuongVD1
 */
public class DienTichHinhTron {
    final static double PI = 3.14159;
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        double bankinh,dientich;
        bankinh = 10.6;
        dientich = PI*bankinh*bankinh;
        System.out.println("Diện tích hình tròn " + dientich
                + " với bán kính " + bankinh);
    }
}
```

Tên - Identifiers

- Một tên là một chuỗi các ký tự gồm các chữ, số, dấu gạch dưới (_), và dấu dollar ($).
- Một tên phải bắt đầu bởi một chữ, dấu gạch dưới (_), hoặc dấu dollar ($). Nó không thể bắt đầu bởi một số.
- Một tên không thể là một từ khóa.
- Một tên không thể là true, false, hoặc null.
- Một tên có thể có độ dài bất kỳ.

Biến - Variables

```
// Tinh dien tich hình chu nhat


bankinh = 10.6;
dientich = bankinh*bankinh*3.14159;
System.out.println("Dien tich bang " + dientich + " voi ban kinh la " + bankinh);
```

Khai báo biến

Dạng thức: datatype variableName;

Ví dụ:

```
int x;                  // Khai báo x là một
                        // biến nguyên (integer);

double bankinh

char a;
```

Lệnh gán và biểu thức gán

Dạng thức: variable = expression;

Ví dụ:

```
x = 1;                  // Gán 1 cho x;
bankinh = 10.6;         // Gán 10.6 cho bankinh;
a = 'A';                // Gán 'A' cho a;
x = x + 1;
dttg = Math.sqrt(p*(p-a)*(p-b)*(p-c)) ;
```

Khai báo và khởi tạo trong 1 lệnh

- int i = 1, j = 5;
- double d = 1.4;
- float pi = 3.1416;

Các câu lệnh trên có đúng không?

Hằng - Constants

- Dạng thức:

final datatype CONSTANTNAME = VALUE;

- Ví dụ:

final double PI = 3.14159;

final int SIZE = 3;

Các kiểu dữ liệu số

byte	8 bits
short	16 bits
int	32 bits
long	64 bits
float	32 bits
double	64 bits

Toán tử - Operators

+ - * / %

int i1 = 5/2 ; ⇒ kết quả là số nguyên i1 = 2

float i2 = 5.0/2 ; ⇒ kết quả là số thực i2 = 2.5

byte i3 = 5 % 2; ⇒ i3 = 1 (số dư của phép chia)

CHÚ Ý

- Các phép tính với số dấu chấm động được lấy xấp xỉ vì chúng được lưu trữ không hoàn toàn chính xác. Ví dụ:

System.out.println(1 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);

hiển thị 0.5000000000000001, không phải 0.5

System.out.println(1.0 - 0.9);

hiển thị 0.09999999999999998, không phải 0.1.

- Các số nguyên được lưu trữ chính xác nên các phép tính với chúng cho kết quả chính xác.

Biểu thức toán học

$$\frac{3+4x}{5} - \frac{10(y-5)(a+b+c)}{x} + 9\left(\frac{4}{x} + \frac{9+x}{y}\right)$$

được chuyển thành công thức Java như sau:

(3+4*x)/5 – 10*(y-5)*(a+b+c)/x + 9*(4/x + (9+x)/y)

Các toán tử gán tắt

Operator	*Example*	*Equivalent*
+=	i+=8	i = i+8
-=	f-=8.0	f = f-8.0
*=	i*=8	i = i*8
/=	i/=8	i = i/8
%=	i%=8	i = i%8

Các toán tử tăng và giảm

suffix

```
x++; // Same as x = x + 1;
```

prefix

```
++x; // Same as x = x + 1;
```

suffix

```
x--; // Same as x = x - 1;
```

prefix

```
--x; // Same as x = x - 1;
```

Các toán tử tăng và giảm (tiếp)

```
int i=10;
int newNum = 10*i++;
```

Equivalent to

```
int newNum = 10*i;
i = i + 1;
```

```
int i=10;
int newNum = 10*(++i);
```

Equivalent to

```
i = i + 1;
int newNum = 10*i;
```

Các toán tử tăng và giảm (tiếp)

- Sử dụng các toán tử tăng và giảm giúp các biểu thức ngắn gọn hơn, nhưng cũng làm cho chúng phức tạp và khó đọc hơn.
- Nên tránh sử dụng các toán tử này trong những biểu thức làm thay đổi nhiều biến hoặc sử dụng cùng một biến nhiều lần như sau: int k = ++i + i.

Biểu thức gán và Câu lệnh gán

- Trước Java 2, tất cả các biểu thức có thể được sử dụng như câu lệnh. Kể từ Java 2, chỉ những loại biểu thức sau có thể là câu lệnh:
- variable op= expression; // Với op là +, -, *, /, %
- ++variable;
- variable++;
- --variable;
- variable--;

Chuyển đổi dữ liệu kiểu số (Ép kiểu)

Xét các câu lệnh sau đây:

```
byte i = 100;
long k = i*3+4;
double d = i*3.1+k/2;

int x = k;  //(sai, int < long)
long k = x; //(đúng, long > int)
```

Luật chuyển

- Khi thực hiện một phép tính nhị phân chứa 2 toán hạng khác kiểu, Java tự động chuyển kiểu toán hạng theo luật sau:

1. Nếu một toán hạng kiểu double, toán hạng khác được chuyển đổi thành kiểu double.
2. Nếu không thì, nếu một toán hạng kiểu float, toán hạng khác được chuyển đổi thành kiểu float.
3. Nếu không thì, nếu một toán hạng kiểu long, toán hạng khác được chuyển đổi thành kiểu long.
4. Nếu không thì, cả hai toán hạng được chuyển đổi thành kiểu int.

Mức ưu tiên Ép kiểu

- double
- float
- long
- int
- short
- byte

Ép kiểu mở rộng và thu hẹp

Ép kiểu mở rộng

double d = 3; (mở rộng kiểu)

Ép kiểu thu hẹp

int i = (int)3.0; (thu hẹp kiểu)

Có sai không? int x = 5/2.0;

Kiểu dữ liệu ký tự

char letter = 'A'; (ASCII)

char numChar = '4'; (ASCII)

char letter = '\u0041'; (Unicode)

char numChar = '\u0034'; (Unicode)

Với các ký tự đặc biệt:

char tab = ‘\t’;

Các ký tự đặc biệt

Description	*Escape Sequence*	*Unicode*
Backspace	\b	\u0008
Tab	\t	\u0009
Linefeed	\n	\u000a
Carriage return	\r	\u000d
Backslash	\\	\u005C
Single Quote	\'	\u0027
Double Quote	\"	\u0022

Appendix B: ASCII Character Set

ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

TABLE B.1 ASCII Character Set in the Decimal Index

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dcl	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	″	#	$	%	&	’
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	‘	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{	|	}	~	del		

ASCII Character Set, const

ASCII Character Set is a subset of the Unicode from \u0000 to \u007f

TABLE B.2 ASCII Character Set in the Hexadecimal Index

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	nl	vt	ff	cr	so	si
1	dle	dcl	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
2	sp	!	“	#	$	%	&	’	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	del

Ép kiểu giữa kiểu ký tự và kiểu số

```
int i = 'a';                    // tương tự int i = (int)'a';

char c = 97;                    // tương tự char c = (char)97;
```

Kiểu boolean và các toán tử

```
boolean a1 = true;
boolean a2 = false;

boolean b = (1 > 2);

boolean b2 = (1 == 2);
```

- Kết quả của phép so sánh là một giá trị logic Boolean: true hoặc false

Các toán tử so sánh

Operator	*Name*
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to

Các toán tử Boolean

Operator	*Name*	*Example*
!	not	!b
&&	and	(1<x) && (x<100)
||	or	a1 || a2
^	exclusive or	a1 ^ a2

Bảng chân lý của toán tử !

p	!p	Example
true	false	!(1 > 2) là true, vì (1 > 2) là false.
false	true	!(1 > 0) là false, vì (1 > 0) là true.

Bảng chân lý của toán tử &&

p1	p2	p1 && p2
F	F	F
F	T	F
T	F	F
T	T	T

Ví dụ:

(3 > 2) && (5 >= 5) là true, vì cả (3 > 2) và (5 >= 5) đều là true.

(3 > 2) && (5 > 5) là false, vì (5 > 5) là false.

Bảng chân lý của toán tử ||

p1	p2	p1 || p2
F	F	F
F	T	T
T	F	T
T	T	T

Ví dụ:

(2 > 3) || (5 > 5) là false, vì cả (2 > 3) và (5 > 5) đều là false.

(3 > 2) || (5 > 5) là true, vì (3 > 2) là true.

Bảng chân lý của toán tử ^

p1	p2	p1 ^ p2
F	F	F
F	T	T
T	F	T
T	T	F

Ví dụ:

(2 > 3) ^ (5 > 1) là true, vì (2 > 3) là false và (5 > 1) là true.

(3 > 2) ^ (5 > 1) là false, vì cả (3 > 2) và (5 > 1) đều là true.

Ví dụ

- *System.out.println("Is " + num + " divisible by 2 and 3? " + ((num % 2 == 0) && (num % 3 == 0)));*

- *System.out.println("Is " + num + " divisible by 2 or 3? " + ((num % 2 == 0) || (num % 3 == 0)));*

- *System.out.println("Is " + num + " divisible by 2 or 3, but not both? " + ((num % 2 == 0) ^ (num % 3 == 0)));*

Xác định năm nhuận?

- Một năm là năm nhuận nếu nó chia hết cho 4, nhưng không chia hết cho 100 hoặc nó chia hết cho 400. Source code xác định năm nhuận như sau:

```
boolean NamNhuan = ((nam % 4 == 0) && (nam % 100 != 0)) || (nam % 400 == 0);
```

Các toán tử & và |

&&: toán tử AND có điều kiện

&: toán tử AND không có điều kiện

||: toán tử OR có điều kiện

|: toán tử OR không có điều kiện

bt1 && bt2

(1 < x) && (x < 100)

(1 < x) & (x < 100)

Các toán tử & và | (tiếp)

Nếu x bằng 1, x bằng bao nhiêu sau khi thực hiện biểu thức?

(x > 1) & (x++ < 10);

Nếu x bằng 1, x bằng bao nhiêu sau khi thực hiện biểu thức?

(1 > x) && (1 > x++);

Và (1 == x) | (10 > x++);?

(1 == x) || (10 > x++);?

Các toán tử điều kiện

Toán tử điều kiện là một loại toán tử đặc biệt vì nó bao gồm ba thành phần cấu thành biểu thức điều kiện

Cú pháp:

<biểu thức 1> ? <biểu thức 2> : <biểu thức 3>;

biểu thức 1: Biểu thức logic. Trả trả về giá trị True hoặc False
biểu thức 2: Là giá trị trả về nếu xác định là True
biểu thức 3: Là giá trị trả về nếu xác định là False

Các toán tử điều kiện (tiếp)

Ví dụ 1:

```
System.out.println(
  (so % 2 == 0)? so + "la so chan" : so + "la so
 le");
```

- tương đương với:

```
if (so % 2 == 0)
  System.out.println(so+"la so chan");
else
  System.out.println(so + "la so le");
```

Các toán tử điều kiện (tiếp)

Ví dụ 2:

```
public class Test {
    public static void main(String[] args) {
        int a = 20;
        int b = 3;

        String s = (a % b == 0) ? "a chia het cho b" : "a khong chia het cho b";
        System.out.println(s);
    }
}
```

Thứ tự ưu tiên các toán hạng

Biểu thức sau được tính như thế nào?

3 + 4 * 4 > 5 * (4 + 3) - ++i

- Tất nhiên phải ưu tiên trong ngoặc trước, ngoài ngoặc sau
- Chỉ dùng ngoặc tròn
- Nhiều tầng ngoặc thì thứ tự ưu tiên ngoặc từ trong ra ngoài

Thứ tự ưu tiên các toán hạng (tiếp)

1. var++, var--
2. +, - (dấu dương, âm), ++var,--var
3. (type) Casting (ép kiểu)
4. ! (Not)
5. *, /, % (nhân, chia thường, chia lấy phần dư)
6. +, - (cộng, trừ)
7. <, <=, >, >= (so sánh)
8. ==, !=; (đẳng thức)
9. & (AND không có điều kiện)
10. ^ (Exclusive OR)

Thứ tự ưu tiên các toán hạng (tiếp)

11. | (OR không có điều kiện)
12. && (AND có điều kiện)
13. || (OR có điều kiện)
14. =, +=, -=, *=, /=, %= (toán tử gán)

Sự kết hợp toán tử - Operator Associativity

- Khi tính toán với 2 toán hạng có cùng mức ưu tiên, sự kết hợp toán tử sẽ xác định thứ tự các phép tính. Tất cả các toán tử nhị phân, ngoại trừ toán tử gán, là kết hợp trái (*left-associative*).

  a – b + c – d là tương đương với **((a – b) + c) – d**

- Các toán tử gán là kết hợp phải. Do đó biểu thức

  a = b += c = 5 tương đương với **a = (b += (c = 5))**

Luật tính biểu thức

- Luật 1: Tính bất kỳ biểu thức con nào có thể tính được từ trái sang phải.
- Luật 2: Các toán hạng được áp dụng theo thứ tự ưu tiên của chúng.
- Luật 3: Luật kết hợp áp dụng cho 2 toán hạng cạnh nhau có cùng mức ưu tiên.

Ví dụ

- 3 + 4 * 4 > 5 * (4 + 3) - 1
- 3 + 4 * 4 > 5 * 7 - 1
- 3 + 16 > 5 * 7 - 1
- 3 + 16 > 35 - 1
- 19 > 35 – 1
- 19 > 34
- false

(1) Trong ngoặc trước

(2) Nhân

(3) Nhân

(4) Cộng

(5) Trừ

(6) Lớn hơn

Thứ tự tính toán toán hạng

Trong Java, các toán hạng được tính từ trái sang phải.

Toán hạng bên trái của một toán tử nhị phân được tính trước bất kỳ phần nào của toán hạng bên phải.

Luật này có quyền ưu tiên hơn các luật đã nêu.

Thứ tự tính toán toán hạng (tiếp)

- Khi các toán hạng có hiệu ứng lề (*side effects*), thứ tự tính toán của các toán hạng rất cần quan tâm.
- Ví dụ, *x* sẽ bằng 1 trong đoạn lệnh sau, vì a được tính bằng 0 trước khi ++*a* tăng nó lên thành 1.

```
int a = 0;
int x = a + (++a);      // 0 + 1
```

- Nhưng *x* sẽ bằng 2 trong đoạn lệnh sau, vì ++*a* tăng nó lên thành 1, rồi cộng với chính nó.

```
int a = 0;
int x = ++a + a;        // 1 + 1
```

Ví dụ

- 3 + 4 * 4 > 5 * (4 + 3) - 1
- 3 + 16 > 5 * (4+3) - 1
- 3 + 16 > 5 * (4+3) - 1
- 19 > 5 * 7 - 1
- 19 > 35 – 1
- 19 > 34
- false

(1) Biểu thức con đầu tiên có thể được tính từ bên trái

(2) Cộng

(3) Cộng trong ngoặc

(4) Nhân

(5) Trừ

(6) Lớn hơn

Kiểu chuỗi ký tự

- Kiểu char chỉ biểu diễn 1 ký tự. Để biểu diễn một chuỗi ký tự, sử dụng kiểu dữ liệu String. Ví dụ:

  String message = "Welcome to Java";

- String là một lớp được định nghĩa trước trong thư viện Java giống như System class và JOptionPane class.
- Kiểu String không phải là kiểu cơ sở mà là một kiểu tham chiếu (*reference type*). Bất kỳ lớp Java nào cũng có thể được sử dụng như một kiểu tham chiếu thay cho một biến.
- Hiện tại, bạn chỉ cần hiểu cách khai báo một biến String, cách gán một chuỗi ký tự cho một biến, và cách ghép các chuỗi.

Ghép chuỗi

- String message = "Welcome " + "to " + "Java";

  // ⇒ message = "Welcome to Java"

- String s = "Chuong" + 2;

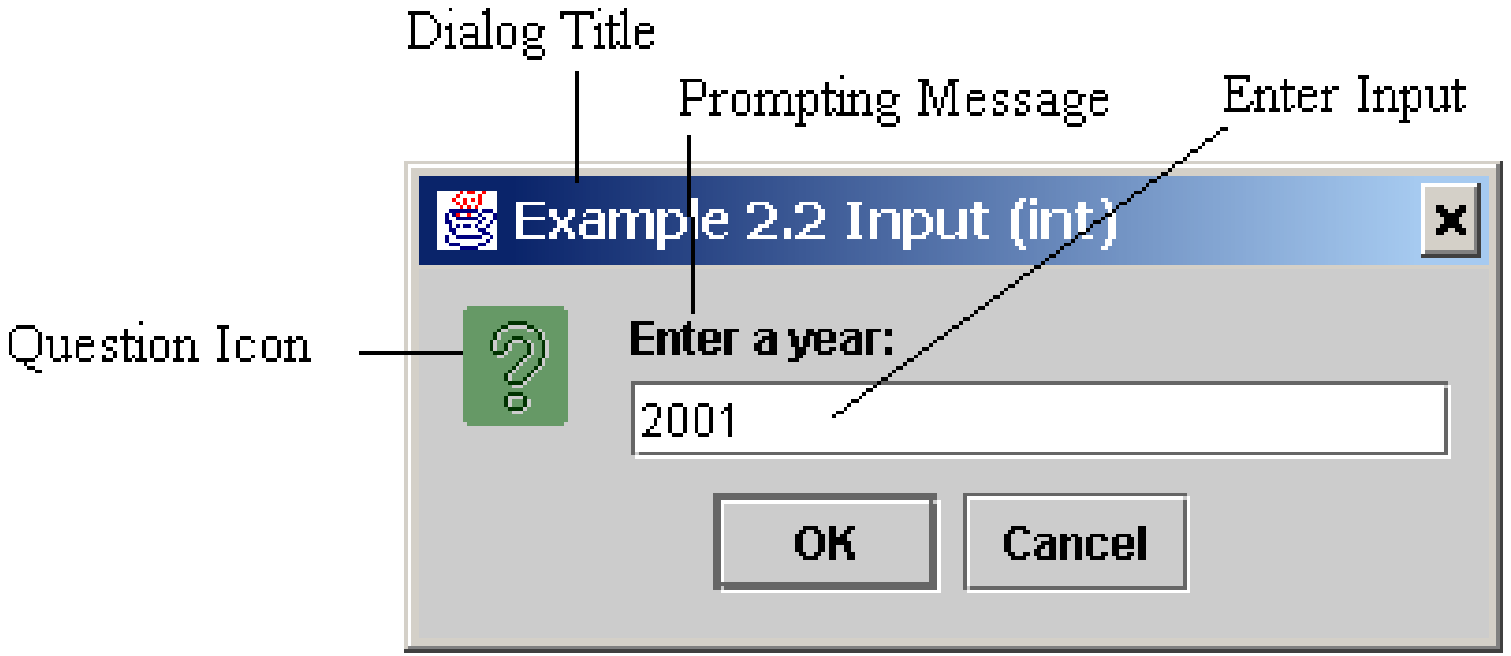
  // ⇒ s trở thành Chuong2

- String s1 = "Hello" + 'B';

  // s1 trở thành HelloB

Nhận dữ liệu từ Input Dialog Box

String string = JOptionPane.showInputDialog(
null, "Prompt Message", "Dialog Title",
JOptionPane.QUESTION_MESSAGE));



Chuyển đổi chuỗi ký tự thành số nguyên

- Dữ liệu trả về từ input dialog box là một chuỗi ký tự. Nếu bạn nhập vào một giá trị số 123, nó trả về chuỗi "123". Để nhận được dữ liệu là một số, bạn phải chuyển đổi.

- Để chuyển đổi một chuỗi ký tự thành một giá trị int, bạn có thể sử dụng phương thức tĩnh parseInt trong lớp Integer như sau:

int intValue = Integer.parseInt(intString);

trong đó intString là một chuỗi số nguyên như "123".

Chuyển đổi chuỗi ký tự thành số thực

Để chuyển đổi một chuỗi ký tự thành một giá trị double, bạn có thể sử dụng phương thức tĩnh parseDouble trong lớp Double như sau:

double doubleValue = Double.parseDouble(doubleString);

trong đó doubleString là một chuỗi số thực như “123.45”.

Ví dụ: Nhập dữ liệu từ Dialog Boxes

Chương trình cho phép người sử dụng nhập vào một năm và kiểm tra đó có phải năm nhuận hay không. Sau đó nhập vào một số thực và kiểm tra có phải số dương hay không.

Năm nhuận là năm chia hết cho 4 nhưng không chia hết cho 100, hoặc là năm chia hết cho 400.

Nhập dữ liệu từ Command Prompt

- Sử dụng MyInput.java
- hoặc sử dụng lớp Scanner (JDK 1.5)

```
import java.util.*;
public class readint{
    static Scanner s=new Scanner(System.in);
    public static void main(String[] abc){
        System.out.print("Doc vao mot so nguyen: ");
        int a=readInt();
        System.out.println("So nguyen la: " + a);
    }
   public static int readInt(){
        return s.nextInt();
    }
}
```

Programming Style

- Chú thích
- Quy ước đặt tên
- Thụt đầu dòng và khoảng cách dòng
- Khối

Quy ước đặt tên

- Chọn các tên mô tả và có ý nghĩa.
- Tên biến và phương thức:
  - Sử dụng chữ thường. Nếu tên có chứa một vài từ, hãy viết liền nhau, sử dụng chữ thường ở từ thứ nhất và viết hoa ký tự đầu tiên của các từ tiếp theo.
  - Ví dụ, các biến radius và area, phương thức computeArea.

Quy ước đặt tên (tiếp)

- **Tên lớp**:
  - Viết hoa ký tự đầu tiên của mỗi từ trong tên. Ví dụ ComputeArea.

- **Tên hằng**:
  - Viết hoa tất cả các ký tự. Ví dụ hằng PI.

Thụt đầu dòng và khoảng cách dòng

- **Thụt đầu dòng**
  - Thụt vào 2 khoảng trống.
  - Cả 2 phía của mỗi toán tử nên có 1 khoảng trống
  - boolean b = 3 + 4 * 4 > 5 * (4 + 3) - ++i;

- **Khoảng cách dòng**
  - Sử dụng dòng trống để ngăn cách các đoạn code.

Block Styles

Sử dụng end-of-line style cho các dấu ngoặc nhọn.

Next-line style

```
public class Test
{
  public static void main(String[] args)
  {
    System.out.println("Block Styles");
  }
}
```

End-of-line style

```
public class Test {
  public static void main(String[] args) {
    System.out.println("Block Styles");
  }
}
```

Programming Errors

- **Syntax Errors**
  - Do trình biên dịch phát hiện
- **Runtime Errors**
  - Gây ra chương trình bị bỏ qua
- **Logic Errors**
  - Tạo ra kết quả sai

Compilation Errors

```
public class ShowSyntaxErrors {
  public static void main(String[] args) {
        i = 30;
        System.out.println(i+4);
  }
}
```

Runtime Errors

```
public class ShowRuntimeErrors {
  public static void main(String[] args) {
        int i = 1 / 0;
  }
}
```

Logic Errors

```
public class ShowLogicErrors {
  public static void main(String[] args) {
        // Cong so1 voi so2
        int so1 = 3;
        int so2 = 5;
        so2 += so1 + so2;
        System.out.println("so2 bang " + so2);
  }
}
```

Q&A

THANK YOU