Toán tử và câu điều kiện rẽ nhánh

(Operators & Conditions)

Nội dung

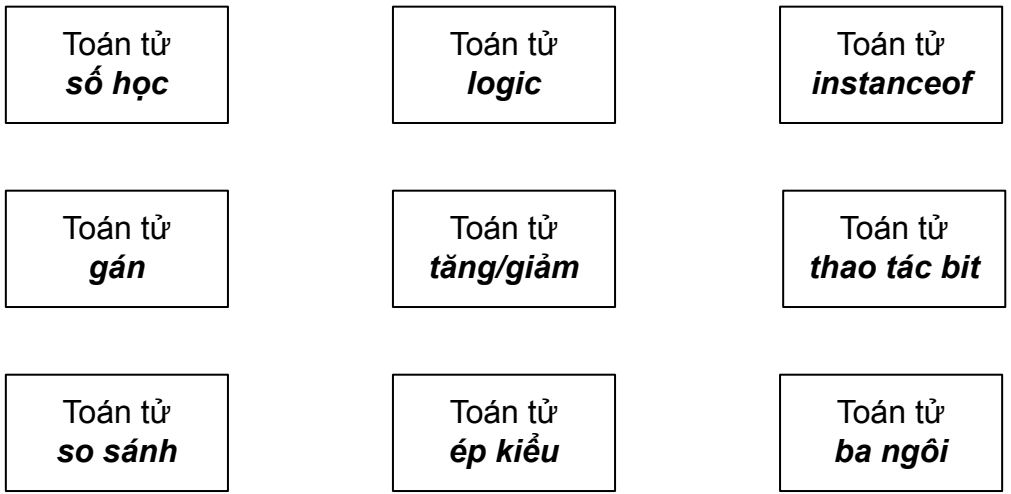
1. **Toán tử (Operators)**
    - Định nghĩa toán tử trong Java
    - Phân loại toán tử trong Java

1. **Câu điều kiện rẽ nhánh (Conditions)**
    - Câu điều kiện rẽ nhánh “if…else”
    - Toán tử ba ngôi
    - Câu điều kiện switch case

1. Toán tử (Operators)

1.1. Định nghĩa và phân loại toán tử

- Toán tử hay còn gọi là *các phép toán*.
- Trong Java có *9 loại* toán tử cơ bản:



1.2. Toán tử số học

Toán tử	Mô tả	Ví dụ
+	Toán tử cộng	1 + 2 a + b
-	Toán tử trừ	1 - 2 a - b
*	Toán tử nhân	1 * 2 a * b
/	Toán tử chia	1 / 2 a / b
%	Toán tử chia dư	10 % 3 = 1 a % b

Ví dụ 1.2

```
// Declare variables
int a = 12;
int b = 5;
// Addition operator
System.out.println("a + b = " + (a + b)); // a + b = 17
// Subtraction operator
System.out.println("a - b = " + (a - b)); // a - b = 7
// Multiplication operator
System.out.println("a * b = " + (a * b)); // a * b = 60
// Division operator
System.out.println("a / b = " + (a / b)); // a / b = 2
// Modulo operator
System.out.println("a % b = " + (a % b)); // a % b = 2
```

1.3. Toán tử gán

Toán tử	Mô tả	Ví dụ	Tương đương
=	Toán tử gán	a = b a = 1	
+=	Toán tử cộng và gán	a += 1 a += b	a = a + 1 a = a + b
-=	Toán tử trừ và gán	a -= 1 a -= b	a = a - 1 a = a - b
/=	Toán tử chia và gán	a /= 1 a /= b	a = a / 1 a = a / b
%=	Toán tử chia dư và gán	a %= 1 a %= b	a = a % 1 a = a % b

Ví dụ 1.3

```
int a = 12;
int b = 5;

System.out.println("Giá trị biến a là: " + a); // Giá trị biến a là: 12
System.out.println("Giá trị biến b là: " + b); // Giá trị biến b là: 5

int c = a;
System.out.println("Giá trị biến c sau khi gán c = a là: " + c);
// Giá trị biến c sau khi gán c = a là: 12

c += b; // c = c + b
System.out.println("Giá trị biến c sau khi cộng và gán c += b là: " + c);
// Giá trị biến c sau khi cộng và gán c += b là: 17
```

1.4. Toán tử so sánh

Toán tử	Mô tả	Ví dụ
==	Toán từ bằng	a = 1 b = 1 a == b ⇒ True
!=	Toán từ khác	a = 1 b = 2 a != b ⇒ True
>	Toán tử lớn hơn	a = 1 b = 2 a > b ⇒ False

Toán tử	Mô tả	Ví dụ
>=	Toán từ lớn hơn hoặc bằng	a = 2 b = 2 a >= b ⇒True
<=	Toán tử nhỏ hơn hoặc bằng	a = 1 b = 2 a <= b ⇒ True
<	Toán tử nhỏ hơn	a = 1 b = 2 a < b ⇒ True

Ví dụ 1.4

```
int a = 12;
int b = 5;

System.out.printf("\n %d == %d is %b", a, b , a == b); // 12 == 5 is false
System.out.printf("\n %d != %d is %b", a, b , a != b); // 12 != 5 is true
System.out.printf("\n %d < %d is %b", a, b , a < b); // 12 < 5 is false
System.out.printf("\n %d > %d is %b", a, b , a > b); // 12 > 5 is true
System.out.printf("\n %d <= %d is %b", a, b , a <= b); // 12 <= 5 is false
System.out.printf("\n %d >= %d is %b", a, b , a >= b); // 12 >= 5 is true
```

1.5. Toán tử logic

Toán tử	Mô tả	Giải thích	Ví dụ
&&	Toán từ và (AND)	a && b là true khi và chỉ khi a true và b true	boolean a = true boolean b = true ⇒ a && b ⇒ true
||	Toán từ hoặc (OR)	a || b là true khi a true hoặc b true	boolean a = true boolean b = false ⇒ a || b ⇒ true
!	Toán tử phủ định (NOT)	!a là true khi a false !a là false khi a true	boolean a = true ⇒ !a ⇒ false boolean a = false ⇒ !a ⇒ true

Ví dụ 1.5

```
boolean a = true;
boolean b = false;

System.out.printf("\n a && b is %b", a && b); // a && b is false
System.out.printf("\n a || b is %b", a || b); // a || b is true
System.out.printf("\n !a is %b", !a); // !a is false
```

1.6. Toán tử tăng/giảm

Toán tử	Mô tả	Ví dụ	Tương đương
++	Toán từ tăng	a = 1 a ++ *hoặc* ++ a	a = 1 a = a + 1
--	Toán tử giảm	a = 1 a -- *hoặc* -- a	a = 1 a = a - 1

Ví dụ 1.6.1

```
// ------------- Kiểu variable++ or variable-- --------------
int a = 1;
int b = 2;
a++; // <=> a = a + 1
System.out.printf("\n Giá trị của a sau khi thực hiện a++ là: %d", a);
// Giá trị của a sau khi thực hiện a++ là: 2

b--; // <=> b = b - 1
System.out.printf("\n Giá trị của b sau khi thực hiện b-- là: %d", b);
// Giá trị của b sau khi thực hiện b-- là: 1
```

Ví dụ 1.6.2

```
// ------------- Kiểu ++variable or --variable --------------
int c = 1;
int d = 2;
++c; // <=> c = c + 1
System.out.printf("\n Giá trị của c sau khi thực hiện ++c là: %d", c);
// Giá trị của c sau khi thực hiện ++c là: 2

--d; // <=> d = d - 1
System.out.printf("\n Giá trị của d sau khi thực hiện --d là: %d", d);
// Giá trị của d sau khi thực hiện --d là: 1
```

Lưu ý khi sử dụng toán tử tăng/giảm và phép gán

Toán tử	Mô tả	Ví dụ	Tương đương	Giải thích
++	Toán từ tăng	a = 1 b = a ++	a = 1 b = a a = a + 1	Giá trị của a được gán cho b trước. Sau đó giá trị của a mới được tăng lên 1. Kết quả: b = 1, a = 2
		a = 1 b = ++ a	a = 1 a = a + 1 b = a	Giá trị của a được tăng lên 1 trước. Sau đó giá trị của a mới được gán cho b. Kết quả: a = 2, b = 2
--	Toán tử giảm	a = 1 b = a --	a = 1 b = a a = a - 1	Giá trị của a được gán cho b trước. Sau đó giá trị của a mới được giảm đi 1. Kết quả: b = 1, a = 0
		a = 1 b = -- a	a = 1 a = a - 1 b = a	Giá trị của a được giảm đi 1 trước. Sau đó giá trị của a mới được gán cho b. Kết quả: a = 0, b = 0

Ví dụ 1.6.3

```
// ------------- Khi dùng cùng phép gán --------------
// ------------- Kiểu variable++ or variable-- --------------
int e = 1;
int f = 2;
int g = e++; // <=> g = e; e = e + 1
System.out.printf("\n Giá trị của g và e sau khi thực hiện g = e++ là: g : %d, e : %d", g, e);
// Giá trị của g và e sau khi thực hiện g = e++ là: g : 1, e : 2

g = f--; // <=> f = f - 1; g = f
System.out.printf("\n Giá trị của g và f sau khi thực hiện g = f-- là: g : %d, f : %d", g, f);
// Giá trị của g và f sau khi thực hiện g = f-- là: g : 2, f : 1
```

Ví dụ 1.6.4

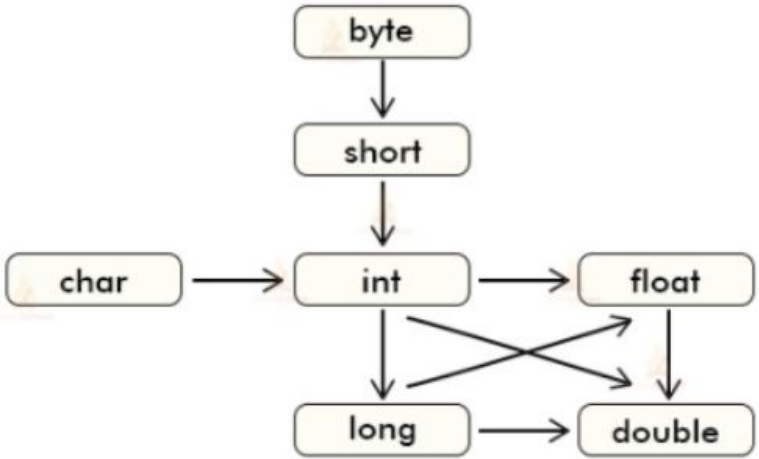
```
// ------------- Khi dùng cùng phép gán --------------
// ------------- Kiểu ++variable or --variable --------------
int i = 1;
int j = 2;
int k = ++i; // <=> i = i + 1; k = i
System.out.printf("\n Giá trị của k và i sau khi thực hiện k = ++i là: k : %d, i : %d", k, i);
// Giá trị của k và i sau khi thực hiện k = ++i là: k : 2, i : 2

k = --j; // <=> j = j - 1; k = j
System.out.printf("\n Giá trị của k và j sau khi thực hiện gk = --j là: k : %d, j : %d", k, j);
// Giá trị của k và j sau khi thực hiện gk = --j là: k : 1, j : 1
```

1.7. Toán tử ép kiểu

- Ép kiểu là việc gán giá trị của một biến có kiểu dữ liệu này sang biến khác có kiểu dữ liệu khác.
- Ép kiểu có thể gây mất thông tin và dữ liệu hiển thị có thể không còn được chính xác.
- Có hai loại ép kiểu dữ liệu:
    - Ép kiểu ngầm định
    - Ép kiểu tường mình

1.7.1. Ép kiểu ngầm định

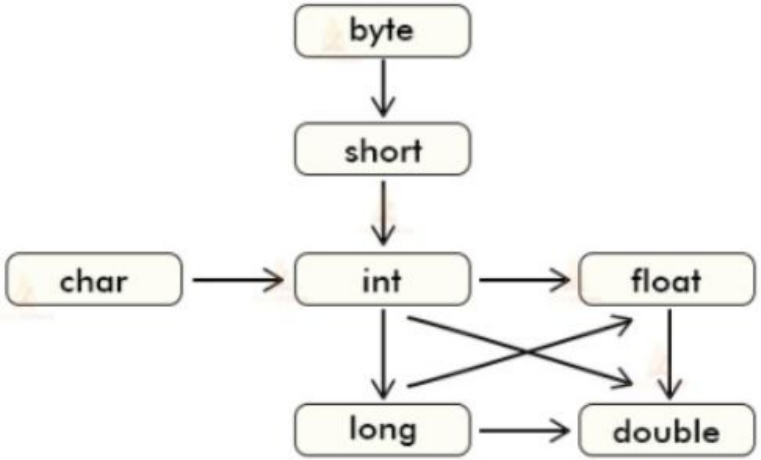
- Ép kiểu ngầm định là quá trình chuyển đổi tự động giữa các kiểu dữ liệu dựa trên quy tắc của ngôn ngữ Java.
- Ép kiểu ngầm định là chuyển đổi kiểu có phạm vi lưu trữ nhỏ hơn, sang phạm vi lưu trữ lớn hơn theo quy tắc của ngôn ngữ Java.



Type	Size (in bits)	Range
byte	8	-128 to 127
short	16	-32,768 to 32,767
int	32	-2^{31} to $2^{31}-1$
long	64	-2^{63} to $2^{63}-1$
float	32	1.4e-045 to 3.4e+038
double	64	4.9e-324 to 1.8e+308
char	16	0 to 65,535

1.7.1. Ép kiểu ngầm định

- Ép kiểu ngầm định là quá trình chuyển đổi tự động giữa các kiểu dữ liệu dựa trên quy tắc của ngôn ngữ Java.
- Ép kiểu ngầm định là chuyển đổi kiểu có phạm vi lưu trữ nhỏ hơn, sang phạm vi lưu trữ lớn hơn theo quy tắc của ngôn ngữ Java.



- float ⇒ double
- long ⇒ float, double
- int ⇒ long, float, double
- char ⇒ int, long, float, double
- short ⇒ int, long, float, double
- byte ⇒ short, int, long, float, double

Ví dụ 1.7.1

```
int intVariable = 25;

long longVariable = intVariable;

float floatVariable = longVariable;

double doubleVariable = floatVariable;

System.out.println("Integer value is: " +intVariable); // Integer value is: 25

System.out.println("Long value is: " +longVariable); // Long value is: 25

System.out.println("Float value is: " +floatVariable); // Float value is: 25.0

System.out.println("Double value is: " +doubleVariable); // Double value is: 25.0
```

1.7.2. Ép kiểu tường minh

- Ép kiểu tường minh là quá trình chuyển đổi kiểu dữ liệu một cách rõ ràng bằng cách sử dụng cú pháp đặc biệt của Java.
- **Cú pháp:** variable1 = (Data type) variable2
- *Chú ý: Khi ép kiểu từ kiểu dữ liệu có kích thước lớn sang kiểu dữ liệu có kích thước nhỏ, để phòng bị mất mát dữ liệu, gây ra hiển thị không chính xác.*

Type	Size (in bits)	Range
byte	8	-128 to 127
short	16	-32,768 to 32,767
int	32	-2^{31} to $2^{31}-1$
long	64	-2^{63} to $2^{63}-1$
float	32	1.4e-045 to 3.4e+038
double	64	4.9e-324 to 1.8e+308
char	16	0 to 65,535

1.7.3. Một số ví dụ về ép kiểu

- Ép kiểu từ số nguyên sang số thực

```
int a = 2;
double b = (double) a;
System.out.println("Giá trị của a = " + a); // Giá trị của a = 2
System.out.println("Giá trị của b = " + b); // Giá trị của b = 2.0
```

- Ép kiểu từ số thực sang số nguyên

```
double c = 4.14156;
int d = (int) c;
System.out.println("Giá trị của c = " + c); // Giá trị của c = 4.14156
System.out.println("Giá trị của d = " + d); // Giá trị của d = 4
```

1.7.3. Một số ví dụ về ép kiểu

- Ép kiểu từ kiểu dữ liệu char sang kiểu int

```
char e = 'e';
int f = (int) e; // Lấy giá trị mã ASCII của ký tự 'e'
System.out.println("Giá trị của e = " + e); // Giá trị của e = e
System.out.println("Giá trị của f = " + f); // Giá trị của f = 101
```

- Ép kiểu từ dạng int sang dạng char

```
int g = 65;
char h = (char) g; // Lấy ký tự có mã ASCII là 65
System.out.println("Giá trị của g = " + g); // Giá trị của g = 65
System.out.println("Giá trị của h = " + h); // Giá trị của h = A
```

Decimal - Binary - Octal - Hex – ASCII Conversion Chart

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(	72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29	)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[	123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	|
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D	]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

1.8. Toán tử Instanceof

- Toán tử instanceof dùng để kiểm tra 1 biến là một instance của một class nào đó, sẽ được tìm hiểu và để cập trong bài lập trình hướng đối tượng.

- Ví dụ:

```
String str = "hello world";
System.out.println("\n Biến str là instance của String là: " + (str instanceof String));
// Biến str là instance của String là: true
```

1.9. Toán tử thao tác với bit

- Toán từ thao tác với bit dụng để thực hiện các phép toán trên các bit riêng lẻ, các toán tử này thường được sử dụng để thực hiện các phép toán logic và máy tính, xử lý dữ liệu nén, masking, và thực hiện một số phép toán khác liên quan đến các bit.

<table>
<tr><th>Toán tử</th><th>Mô tả</th><th>Ví dụ</th></tr>
<tr><td>&</td><td>AND bit (&): Bằng 1 ⇔x = y = 1, còn lại bằng 0
<table><tr><td>x & y</td><td>0</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr></table></td><td>int a = 5 ⇔ 0101
int b = 3 ⇔ 0011
int result = a & b = 0001 (1)</td></tr>
<tr><td>|</td><td>OR bit (|): Bằng 1 ⇔ x = 1 hoặc y = 1, còn lại bằng 0
<table><tr><td>x | y</td><td>0</td><td>1</td></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td><td>int a = 5 ⇔ 0101
int b = 3 ⇔ 0011
int result = a | b = 0111 (7)</td></tr>
</table>

<table>
<tr><th>Toán tử</th><th>Mô tả</th><th>Ví dụ</th></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>^</td><td>XOR bit (^): Bằng 1 khi x khác y, còn lại bằng 0
<table><tr><th>x ^ y</th><th>0</th><th>1</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></td><td>int a = 5 ⇔0101
int b = 3 ⇔0011
int result = a ^ b ⇔ 0110 (6)</td></tr>
<tr><td>~</td><td>NOT bit (~): Đảo ngược từng bít, 0 → 1, 1 → 0</td><td>int a = 5 ⇔0101
int result = ~a ⇔ 1010 (-6)</td></tr>
<tr><td><<</td><td>Dịch trái bit</td><td>a = 5 ⇔ 00000101
a << 2 ⇔00010100 (20)</td></tr>
<tr><td>>></td><td>Dịch phải bit.
Nếu số ban đầu là số âm, thì sau dịch, các bit trái sẽ được điền bằng 1.
Nếu số ban đầu là số dương, thì sau dịch, các bit trái sẽ</td><td>a = 5 ⇔ 00000101
a >> 2 ⇔00000001 (1)</td></tr>
</table>

1.10. Toán tử ba ngôi

- Toán tử ba ngôi là cách viết thay thế một phần của câu lệnh if-else ngắn gọn và đơn giản, toán từ này sẽ được giới thiệu chi tiết ở phần 2.

2. Câu điều kiện rẽ nhánh (Conditions)

2.1. Khái niệm và phân loại câu điều kiện rẽ nhánh

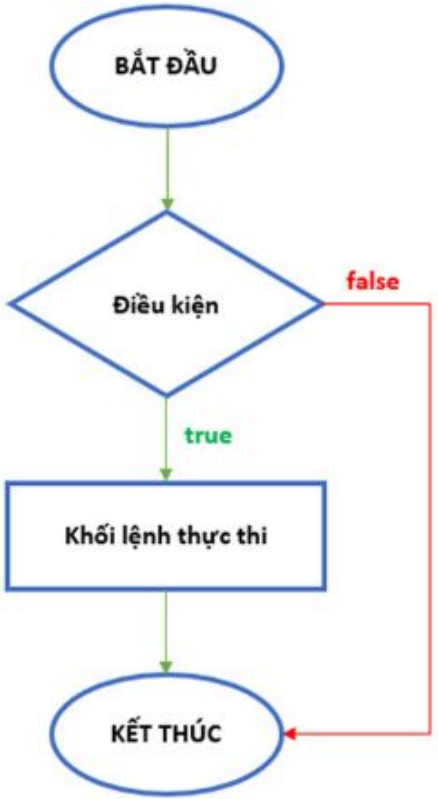
- Câu điều kiện rẽ nhánh thường được sử dụng kết hợp với các ***toán tử so sánh*** để đưa ra quyết định thực thi trong chương trình.
- Câu điều kiện rẽ nhánh có 5 loại:
  - Câu điều kiện rẽ nhánh “if”
  - Câu điều kiện rẽ nhánh “if…else”
  - Câu điều kiện rẽ nhánh “if … else if … else”
  - Toán tử ba ngôi
  - Câu điều kiện switch case

2.2. Câu điều kiện rẽ nhánh “if”

- Câu điều kiện rẽ nhánh if là câu điều kiện mà nếu điều kiện đúng thì mới thực thi khối lệnh.

- **Cú pháp**

```
if(điều kiện) {
    Các câu lệnh cần thực thi nếu điều kiện đúng
}
```



Ví dụ 2.2

Viết một chương trình sử dụng câu điều kiện rẽ nhánh if để kiểm tra một số có phải là số lớn hơn 0 không? Nếu đúng là số lớn hơn 0, thì in ra “YES".

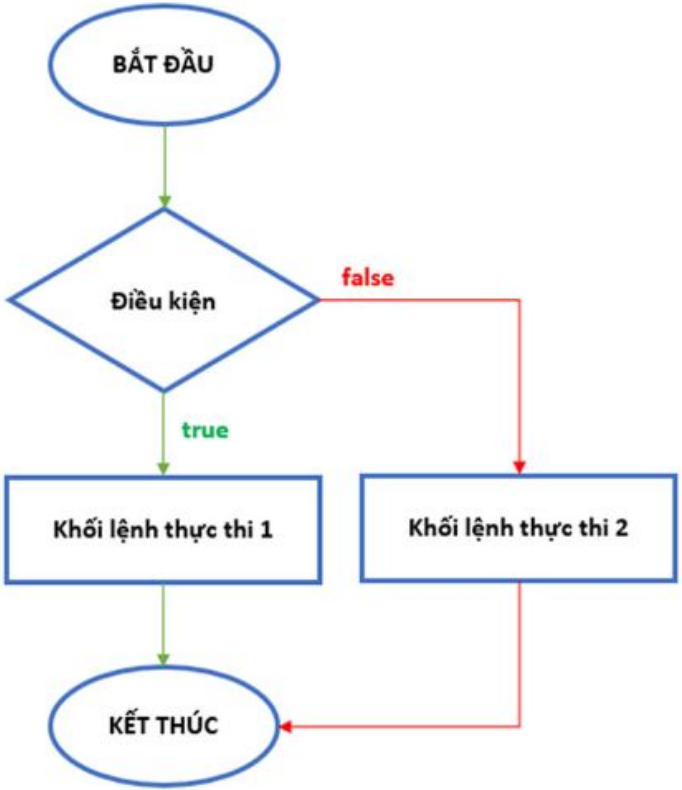
```
public static void main(String[] args) {
    int a = 1;
    if(a > 0) {
        System.out.println("YES"); // YES vì a = 1 > 0
    }
}
  public static void main(String[] args) {
      int b = -1;
      if(b > 0) {
          System.out.println("YES"); // Không in ra gì cả, vì b không lớn hơn 0
      }
  }
```

2.3. Câu điều kiện rẽ nhánh “if…else”

- Câu điều kiện rẽ nhánh if … else là câu điều kiện mà nếu điều kiện đúng thì thực thi khối lệnh 1, nếu sai thì thực khi khối lệnh 2.

- **Cú pháp:**

```
if(điều kiện) {
   Các câu lệnh cần thực hiện khi điều kiện
đúng
} else {
   Các câu lệnh cần thực hiện khi điều kiện
sai
}
```



Ví dụ 2.3

- Viết một chương trình sử dụng câu điều kiện rẽ nhánh if … else để kiểm tra một số có phải là số lớn hơn 0 không? Nếu đúng là số lớn hơn 0, thì in ra “YES", ngược lại in ra “NO”.

```
int a = 1;
if(a > 0) {
   System.out.println("YES"); // YES vì a = 1 > 0
} else {
   System.out.println("NO");
}
-----------------------------------------------
int b = -1;
if(b > 0) {
   System.out.println("YES");
} else {
   System.out.println("NO"); // NO vì b = -1 < 0
}
```

2.4. Câu điều kiện rẽ nhánh “if…else if…else”

- Câu điều kiện rẽ nhánh “if … else if … else” là câu điều kiện mà nếu điều kiện 1 đúng thì làm việc 1, còn không kiểm tra điều kiện 2, nếu điều kiện 2 đúng thì làm việc 2. Nếu điều kiện 1 và điều kiện 2 đều không đúng thì làm việc 3.

Cú pháp:

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

Ví dụ 2.4

Viết một chương trình sử dụng câu điều kiện rẽ nhánh if … else if … else để kiểm tra một số có phải là số lớn hơn 0 không? Nếu đúng là số lớn hơn 0, thì in ra “YES", nếu bằng 0 thì in ra “EQUALS”, nếu nhỏ hơn 0 thì in ra “NO”.

```
int a = 1;
if(a > 0) {
    System.out.println("YES"); // YES, vì a > 0
} else if(a == 0) {
    System.out.println("EQUALS");
}
else {
    System.out.println("NO");
}
```

2.5. Toán tử ba ngôi

- Toán tử 3 ngôi là một loại toán tử đặc biệt, toán từ ba ngôi hoạt động tương đương câu ***điều kiện if else***, và thường được sử dụng cùng phép gán.
- **Cú pháp**

Toán từ ba ngôi	Điều kiện if … else

biến = điều kiện
? biến được gán bằng giá trị 1, nếu điều kiện đúng
: biến được gán bằng giá trị 2, nếu điều kiện sai
	if(điều kiện) {
	biến được gán bằng giá trị 1, nếu điều kiện đúng
	} else {
	biến được gán bằng giá trị 2, nếu điều kiện sai
	}

Ví dụ 2.5

Viết một chương trình để gán giá trị cho biến result dựa vào giá trị của một số. Nếu số đó lớn hơn 0, thì biến result sẽ có giá trị “YES", ngược lại sẽ có giá trị “NO”.

Toán từ ba ngôi

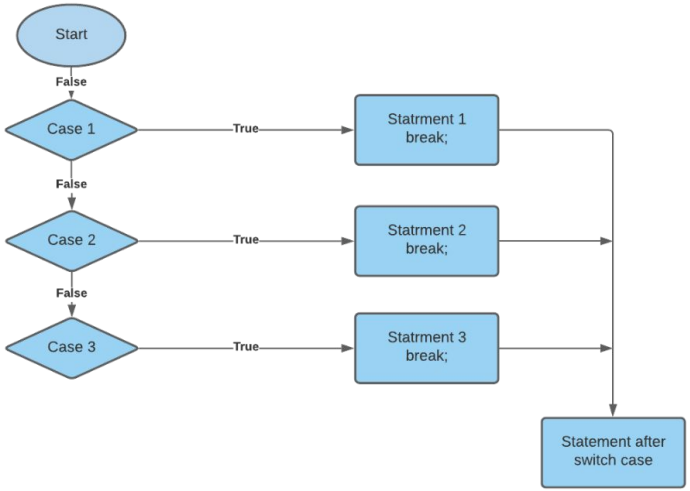
```
// Cách 1: Khai báo và gán tách rời nhau
int a = 1;
String result;
result = a > 0 ? "YES" : "NO";

System.out.println("Giá trị của biến result là:
" + result); // Giá trị của biến result là:
YES, vì a = 1 > 0

// Cách 2: Khai báo kết hợp gán theo điều kiện
int a = 1;
String result = a > 0 ? "YES" : "NO";

System.out.println("Giá trị của biến result là:
" + result); // Giá trị của biến result là:
YES, vì a = 1 > 0
```

Điều kiện if … else

```
int a = 1;
String result;
if(a > 0) {
   result = "YES";
} else {
   result = "NO";
}

System.out.println("Giá trị của biến result
là: " + result); // Giá trị của biến result
là: YES, vì a = 1 > 0
```

2.6. Câu điều kiện Switch Case

- Mệnh đề switch case được sử dụng khi mà một điều kiện có thể có nhiều hơn 1 trường hợp có thể xảy ra.
- Ví dụ, ta xét một ngày trong tuần, có có 7 trường hợp có thể xảy ra (từ thứ 2 đến thứ chủ nhật).



Cú pháp

```
switch(biến) {
   case biến có giá trị bằng x: {
       // Đoạn code cần thực hiện, nếu biến bằng x
       break; // Lệnh break thoát khỏi switch
   }
   case biến có giá trị bằng y: {
       // Đoạn code cần thực hiện, nếu biến bằng y
       break;
   }
   ...
   default: { // Khối default có thể có hoặc không
       // Đoạn code cần thực hiện, nếu biến biến khác tất cả các giá trị trên
   }
}
```

Một số lưu ý khi sử dụng Switch Case

- Switch chỉ hỗ trợ các kiểu dữ liệu:
  - byte, short, char, int và các Wrapper class tương ứng.
  - enum, String và var (tương ứng với các type trên).
- Các giá trị trong câu lệnh case phải là một hằng số.
- Lệnh break làm chương trình thoát ra khỏi switch
- Khối lệnh default không nhất thiết phải có
- Khi tìm thấy một trường hợp đúng, khối lệnh của trường hợp đó sẽ được thực thi. Nếu không bắt gặp lệnh break trong khối lệnh này, chương trình sẽ thực hiện tiếp các khối lệnh bên dưới cho tới khi nó bắt gặp lệnh break, hoặc không còn khối lệnh nào để thực thi.

Ví dụ 2.6

- Viết một chương trình Java để xác định và in ra màn hình ngày trong tuần dựa trên một số nguyên từ 1 đến 7, trong đó 1 đại diện cho Chủ Nhật và 7 đại diện cho Thứ Bảy. Nếu ngày không phải là một số trong khoảng [1, 7] thì in ra màn hình dòng chữ “Ngày không hợp lệ”.

```
int day = 5;
if(day == 1) {
   System.out.println("Chủ nhật");
} else if(day == 2) {
   System.out.printf("\n Thứ %d", day);
} else if(day == 3) {
   System.out.printf("\n Thứ %d", day);
} else if(day == 4) {
   System.out.printf("\n Thứ %d", day);
} else if(day == 5) { // Khối if này được
thực hiện vì day = 5
   System.out.printf("\n Thứ %d", day);
} else if(day == 6) {
   System.out.printf("\n Thứ %d", day);
} else if(day == 7) {
   System.out.printf("\n Thứ %d", day);
} else {
   System.out.println("Ngày không hợp lệ");
```

```java
int day = 5;
switch (day) {
    case 1: {
        System.out.println("Chủ nhật");
        break;
    }
    case 2: {
        System.out.printf("\n Thứ %d", day);
        break;
    }
    case 3: {
        System.out.printf("\n Thứ %d", day);
        break;
    }
    case 4: {
        System.out.printf("\n Thứ %d", day);
        break;
    }
    case 5: { // Case này được thực hiện vì day = 5
        System.out.printf("\n Thứ %d", day);
        break;
    }
    case 6: {
        System.out.printf("\n Thứ %d", day);
        break;
    }
    case 7: {
        System.out.printf("\n Thứ %d", day);
        break;
    }
    default: {
        System.out.println("Ngày không hợp lệ");
    }
}
```

Trong trường hợp nếu các khối case có logic thực hiện xử lý giống nhau, ta có thể viết gộp như dưới đây.

```
int day = 5;
switch (day) {
    case 1: {
        System.out.println("Chủ nhật");
        break;
    }
    case 2:
    case 3:
    case 4:
    case 5:
    case 6:
    case 7: {
        System.out.printf("\n Thứ %d", day); // Thứ 5. vì day = 5
        break;
    }
    default: {
        System.out.println("Ngày không hợp lệ");
    }
}
```

So sánh mệnh đề Switch Case và điều kiện If…else if…else

Điều kiện if … else if … else	Mệnh đề switch case
Mỗi if có biểu thức logic bên trong nó để định giá trị là đúng hay sai	Mỗi case trong switch phải là một giá trị cụ thể, không có biểu thức logic bên trong.
Các biến trong biểu thức là bất kỳ kiểu giá trị nào	Biểu thức phải xác định giá trị là byte (Byte), short (Short), char (Character), int (Integer), enum, String, var (tương ứng với các type trên).
Chỉ một khối lệnh được thực thi	Nếu câu lệnh break bị bỏ qua, thì các câu lệnh từ case đúng trở về sau sẽ được thực hiện.

Tổng Kết

- Toán tử là gì? Các loại toán tử trong Java.
- Câu điều kiện rẽ nhánh
- Mệnh đề switch case

Homework

1. Khai báo và gán giá trị cho một biến có kiểu dữ liệu là String và in ra màn hình kết quả của biến đó.
2. Khai báo và gán giá trị cho một biến có kiểu dữ liệu là int. Viết chương trình kiểm tra xem biến đó chẵn hay lẻ, âm hay dương, in kết quả kiểm tra ra màn hình. Mỗi kết luận trên một dòng.
3. Cho hai số nguyên a, b. In ra màn hình tổng, hiệu, tích, thương của chúng. Lưu ý khi xử lý phép chia sẽ cần ép kiểu, kiểm tra mẫu khác 0. Xuất kết quả ra màn hình.
4. Cho hai số nguyên a, b. So sánh xem số nào lớn hơn, số nào nhỏ hơn hay hai số bằng nhau. In kết quả ra màn hình.
5. Cho a và b là hai cạnh của hình chữ nhật và tính chu vi và diện tích của hình chữ nhật đó. Hiển thị kết quả lên màn hình.
6. Cho a, b và c là 3 số thực bất kỳ. Kiểm tra xem 3 số đó có thể là 3 cạnh của một tam giác không. Nếu có in ra màn hình là “YES", nếu không thì in “NO”.

Tài liệu tham khảo

[1] IntroductIon to Java ProgrammIng and data Structures (Eleventh Edition - Global Edition - Y. daniel Liang)

Thank you 🙏