Operators

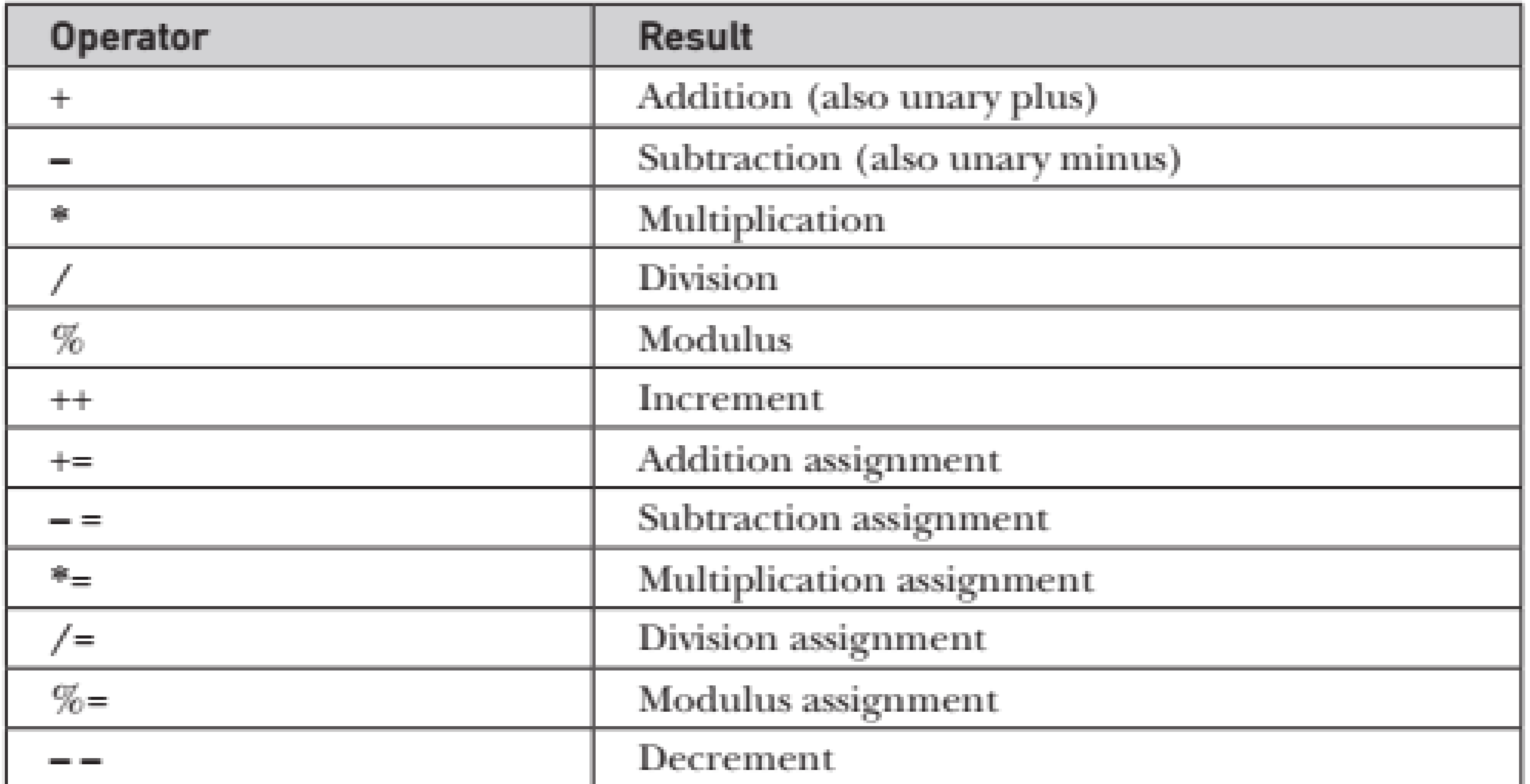
Type of Operator

Broadly can be divided into four groups: arithmetic, bitwise, relational, and logical.

- Arithmetic Operators
- Bitwise Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Ternary Operator

Arithmetic Operators

- Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra.
- The operands of the arithmetic operators must be of a numeric type.
- We cannot use them on boolean types, but we can use char types.



Operator	Result
+	Addition (also unary plus)
–	Subtraction (also unary minus)
*	Multiplication
/	Division
%	Modulus
++	Increment
+=	Addition assignment
–=	Subtraction assignment
*=	Multiplication assignment
/=	Division assignment
%=	Modulus assignment
– –	Decrement

Arithmetic Operators

- ➢Basic Arithmetic Operators:
- addition, subtraction, multiplication, and division: all work with all numeric types.
- ➢The Modulus Operator (%):
- It returns the remainder of a division operation.
- It can be applied to floating-point types as well as integer types.
- ➢Arithmetic Compound Assignment Operators:
- Java provides special operators that can be used to combine an arithmetic operation with an assignment.
- There are compound assignment operators for all of the arithmetic, binary operators.

```
        var = var op expression;                        var op= expression;
         a=a+10;                                        a+=10;
```

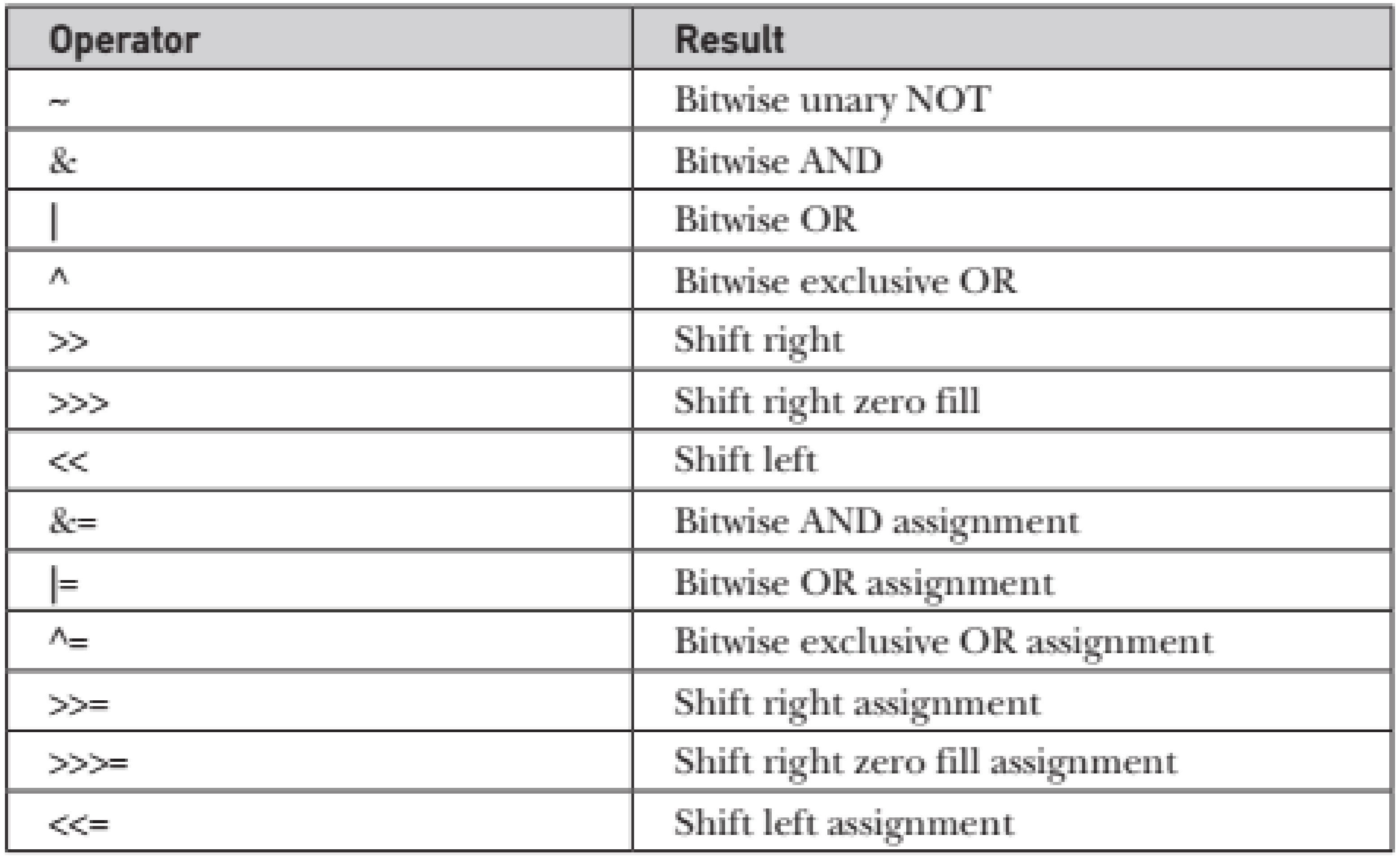
Arithmetic Operators

➢Increment and Decrement :

- The ++ and the – – are Java’s increment and decrement operators.
- The increment operator increases its operand by one.
- The decrement operator decreases its operand by one.
- For example, this statement: x = x + 1;
- can be rewritten by use of the increment operator: x++;
- Similarly, this statement: x = x - 1;
- is equivalent to x--;.
- These operators are unique and can be used as postfix and prefix form
- Postfix: operator follows operand, eg. x++;
- Prefix: operator precede the operand, eg. ++x;

Bitwise Operators

- Java defines several bitwise operators that can be applied to the integer types: long, int, short, char, and byte.
- These operators act upon the individual bits of their operands.

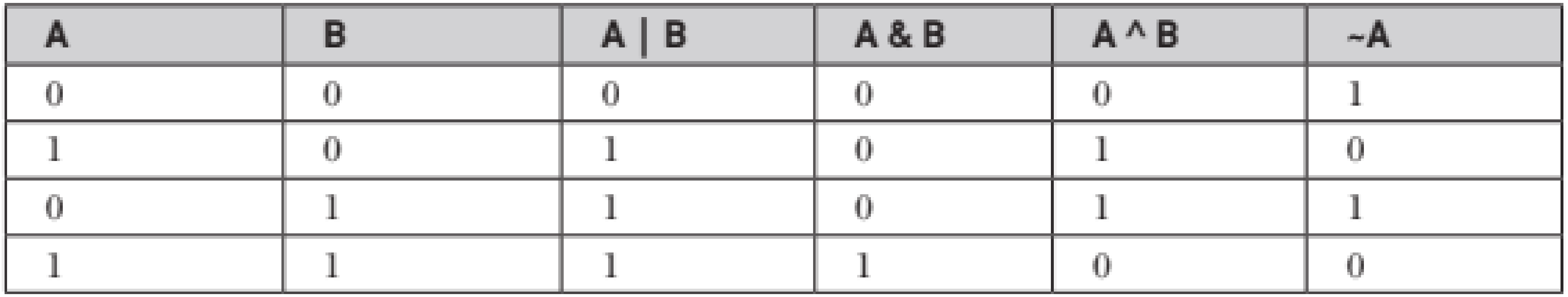


Operator	Result
~	Bitwise unary NOT
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
>>	Shift right
>>>	Shift right zero fill
<<	Shift left
&=	Bitwise AND assignment
|=	Bitwise OR assignment
^=	Bitwise exclusive OR assignment
>>=	Shift right assignment
>>>=	Shift right zero fill assignment
<<=	Shift left assignment

Bitwise Operators

➢Bitwise Logical Operators:

- The bitwise logical operators are &, |, ^, and ~.



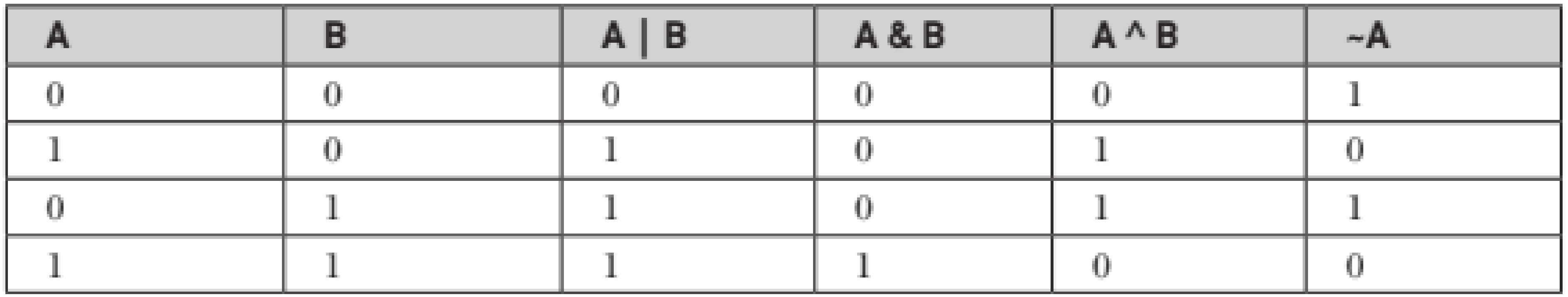
A	B	A | B	A & B	A ^ B	~A
0	0	0	0	0	1
1	0	1	0	1	0
0	1	1	0	1	1
1	1	1	1	0	0

- Bitwise NOT (~):
- Also called the bitwise complement, the unary NOT operator
- inverts all of the bits of its operand.
- Bitwise AND (&):
- The AND operator, produces a 1 bit if both operands are also 1.
- A zero is produced in all other cases.

Bitwise Operators

➢Bitwise Logical Operators:

- The bitwise logical operators are &, |, ^, and ~.



A	B	A | B	A & B	A ^ B	~A
0	0	0	0	0	1
1	0	1	0	1	0
0	1	1	0	1	1
1	1	1	1	0	0

- Bitwise OR (|)
- The OR operator, combines bits such that if either of the bits in the operands is a 1, then the resultant bit is a 1.
- Bitwise XOR (^):
- The XOR operator, combines bits such that if exactly one operand is 1, then the result is 1. Otherwise, the result is zero.

Bitwise Operators

➢Bitwise Logical Operators:

- Left Shift (<<):
- The left shift operator, shifts all of the bits in a value to the left a specified number of times.

```
value << num;
```

- Right Shift (>>):
- The right shift operator, shifts all of the bits in a value to the right a specified number of times.

```
value >> num;
```

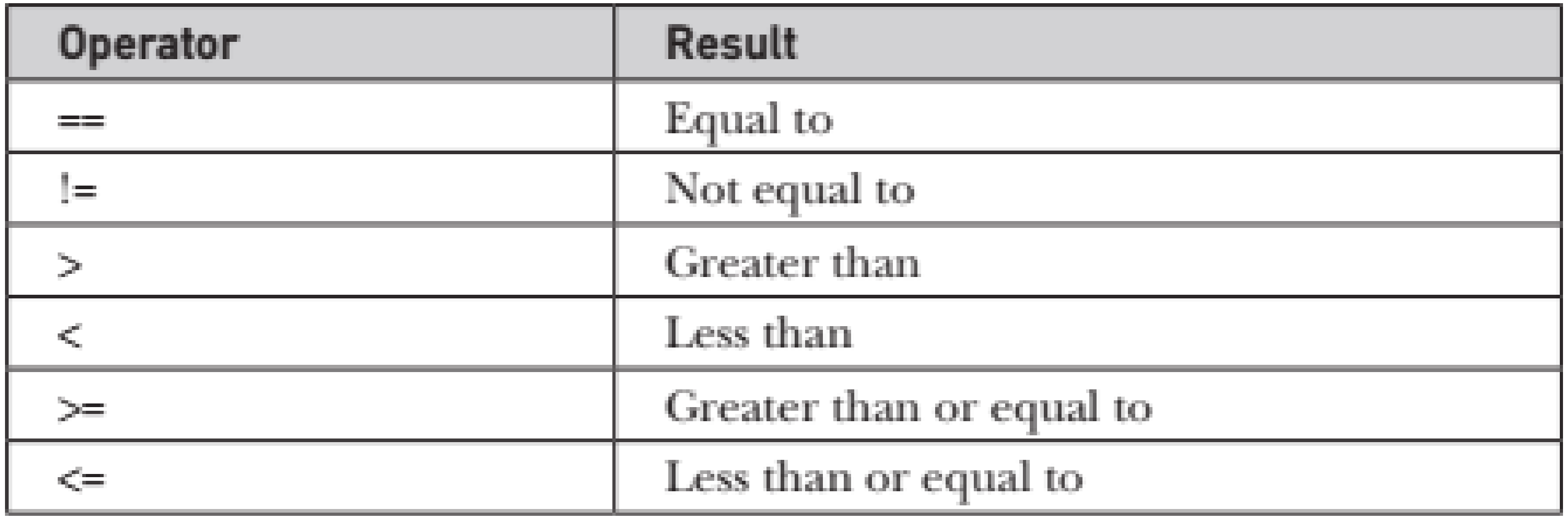
- Unsigned Right Shift (>>>):
- This is known as an unsigned shift.
- always shifts zeros into the high-order bit.

Bitwise Operators

- Bitwise Operator Compound Assignments:
- All of the binary bitwise operators have a compound form similar to that of the algebraic operators

Relational Operators

- The relational operators determine the relationship that one operand has to the other.

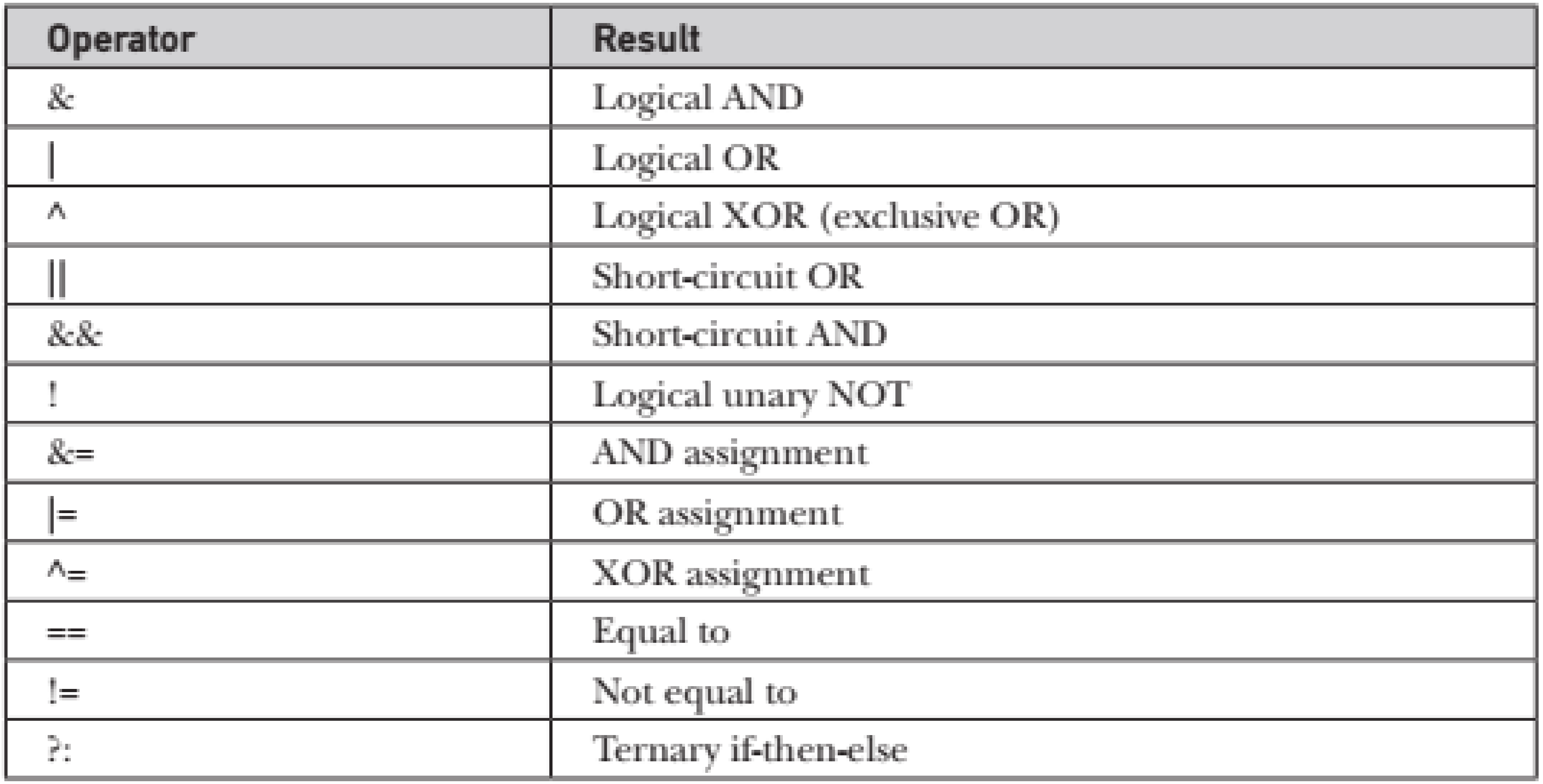


Operator	Result
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

- The outcome of these operations is a boolean value.
- The relational operators are most frequently used in the expressions that control the if statement and the various loop statements.
- Any type, including integers, floating-point numbers, characters, and Booleans can be compared using the equality test, ==, and the inequality test, !=.

Logical Operators

- Boolean Logical Operators



Operator	Result
&	Logical AND
|	Logical OR
^	Logical XOR (exclusive OR)
||	Short-circuit OR
&&	Short-circuit AND
!	Logical unary NOT
&=	AND assignment
|=	OR assignment
^=	XOR assignment
==	Equal to
!=	Not equal to
?:	Ternary if-then-else

Assignment Operator

- Assignment operators are used to assign values to variables.
- The assignment operator is the single equal sign, =.

```
var = expression;
```

Ternary Operator

- Java includes a special ternary (three-way) operator that can replace certain types of if-then-else statements.
- expression1 ? expression2 : expression3
- use of the ternary operator makes our code more readable and clean.
- ternary operator can also be nested.