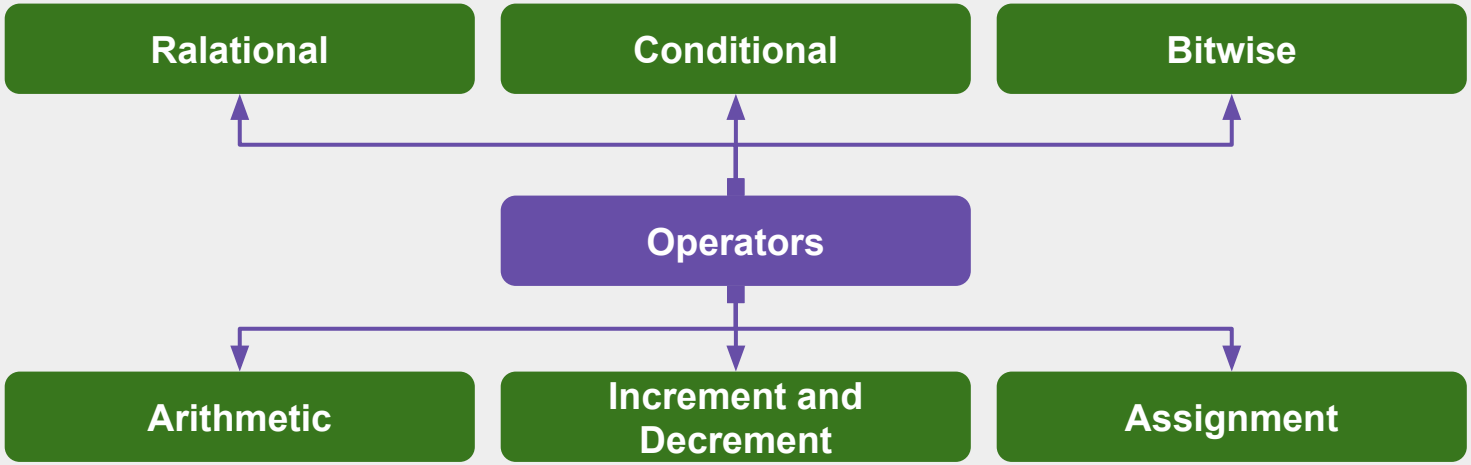
Operators

Outline

1. Operators
2. Arithmetic Operators
3. Conversions between Numeric Types
4. Casts
5. The Assignment Operators
6. Increment and Decrement Operators
7. Relational Operators
8. Conditional Operator
9. Bitwise operators
10. Operator Precedence
11. Interview Questions

Operators

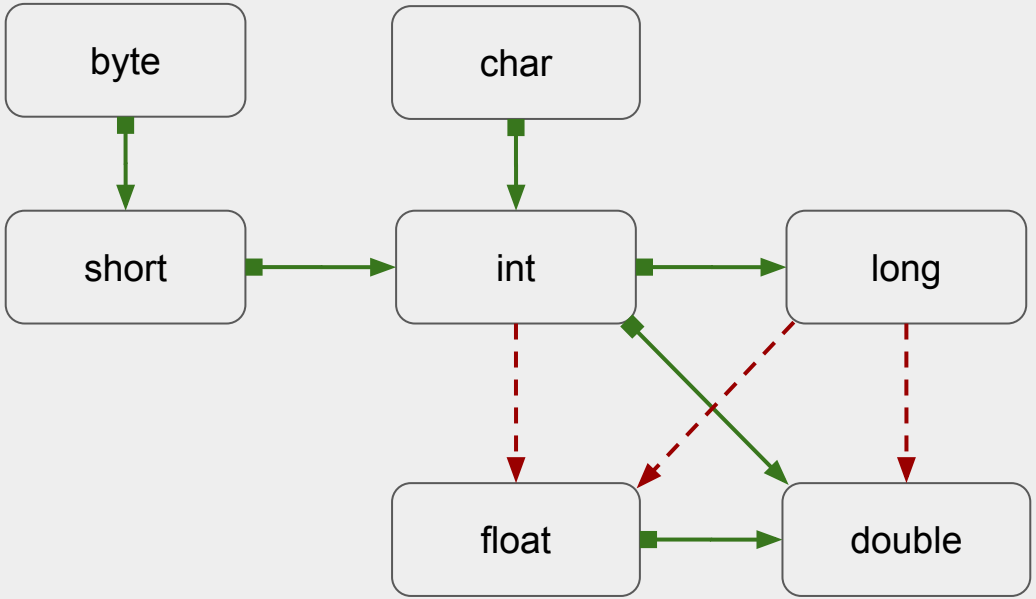
Operators are used to combine values



Arithmetic Operators

addition	**+**
subtraction	**-**
multiplication	*****
division	**/**
Integer remainder(modulus)	**%**

Conversions between Numeric Types



- The six solid arrows denote conversions without information loss.
- The three dashed arrows denote conversions that may lose precision.

```
int n = 123456789;
float f = n; // f is 1.23456792E8
```

Conversions between Numeric Types

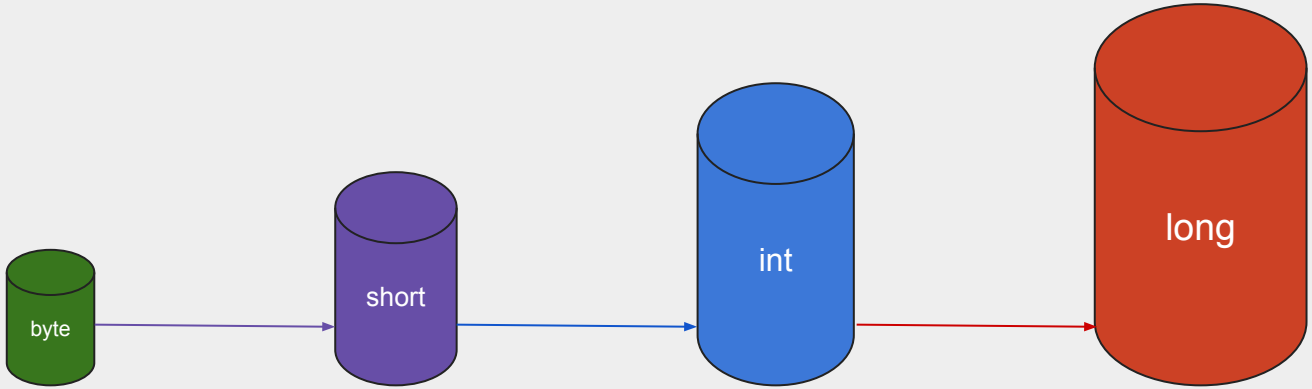
When two values are combined with a binary operator (such as n + f where n is an integer and f is a floating-point value), both operands are converted to a common type before the operation is carried out.

- If either of the operands is of type double, the other one will be converted to a double.
- Otherwise, if either of the operands is of type float, the other one will be converted to a float.
- Otherwise, if either of the operands is of type long, the other one will be converted to a long.
- Otherwise, both operands will be converted to an int.

Casts

- Widening Casting (automatically) - converting a smaller type to a larger type size

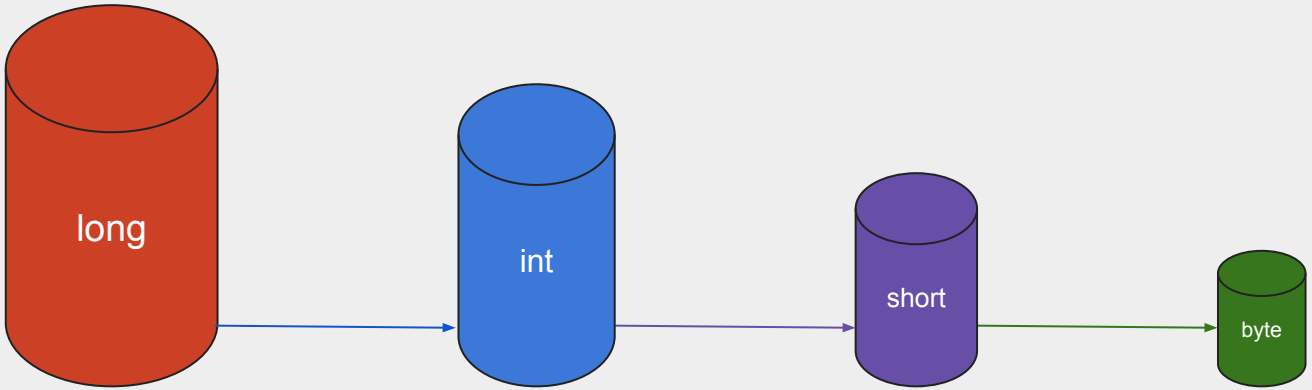
```
byte -> short -> char -> int -> long -> float -> double
```



Casts

- Narrowing Casting (manually) - converting a larger type to a smaller size type

```
double -> float -> long -> int -> char -> short -> byte
```



The Assignment Operators

There is a convenient shortcut for using binary operators in an assignment.

```
x += 4; //is equivalent to x = x + 4;
x += 3.5; //is valid
int y = x += 4; //is valid
```

The Assignment Operators

=	int c = 12;
+=	c+=2 is roughly the same as c = c + 2
-=	c-=2 is roughly the same as c = c - 2
*=	c*=2 is roughly the same as c = c * 2
/=	c/=2 is roughly the same as c = c / 2
%=	c%=2 is roughly the same as c = c % 2
<<=	c<<=2 is roughly the same as c = c << 2
>>=	c>>=2 is roughly the same as c = c >> 2
&=	c&=2 is roughly the same as c = c & 2
|=	c|=2 is roughly the same as c = c | 2
^=	c^=2 is roughly the same as c = c ^ 2

Increment and Decrement Operators

One of the most common operations with a numeric variable is to add or subtract 1. Java, following in the footsteps of C and C++, has both increment and decrement operators

```
int n = 12;
n++;
```

Since these operators change the value of a variable, they cannot be applied to numbers themselves.

```
4++; // is not a legal statement.
```

There are two forms of these operators ***postfix form(n++)*** and ***prefix form(++n)***. The difference between the two appears only when they are used inside expressions.

```
int m = 7;
int n = 7;
int a = 2 * ++m; // now a is 16, m is 8
int b = 2 * n++; // now b is 14, n is 8
```

Relational Operators

==	equality sign
>	greater than sign
<	less than sign
>=	greater than or equal sign
<=	less than or equal sign

Relational Operators

`&&`	`expression1 && expression2`
`||`	`expression1 || expression2`
`!`	`!expression1`

a	b	a || b	a && b	!a
true	false	true	false	false
true	true	true	true	false
false	true	true	false	true
false	false	false	false	true

Conditional Operator

Java provides the ***conditional ?: operator*** that selects a value, depending on a Boolean expression. The expression *condition ? expression1 : expression2* evaluates to the first expression if the condition is true, to the second expression otherwise. Sometimes a **conditional operator** is called a **ternary operator**.

```
int min = x < y ? x : y;
```

The Bitwise operators

- Bitwise OR (|)
- Bitwise AND (&)
- Bitwise XOR (^)
- Bitwise Complement (~)

x	y	|
1	1	1
1	0	1
0	1	1
0	0	0

x	y	&
1	1	1
1	0	1
0	1	1
0	0	0

x	y	^
1	1	0
1	0	1
0	1	1
0	0	0

The Shifting Operators

```
int x = 11;
n = 1;
```

x_{binary}

Left shift << n

`x << n;`

Right shift >> n

`x >> n;`

Operator Precedence

If no parentheses are used, Operators on the same level are processed from left to right, except for those that are right-associative, as indicated in the table. For example, && has a higher precedence than ||, so the expression

```
a && b || c means (a && b) || c
```

Since += associates right to left, the expression

```
a += b += c means a += (b += c)
```

Java Operator Precedence Table

Interview Questions