

Update in Place

Compound assignment operators



Update the balance

PAUSE AND TRY

Write each update in Java, the 1.4 way. What is the final balance?

YOUR BANK ACCOUNT

1. It starts with \$50.
2. You deposit \$20.
3. You spend \$15.

YOUR CODE

```
int balance = 50;  
_____  
_____  
System.out.println(balance);
```

By the end of class you can...

- 1 Use += and -= to update a variable from its own value.
- 2 Use *=, /=, and %=, and predict the result for int and double variables.
- 3 Explain why the whole right side is worked out first.
- 4 Use ++ and -- to add or subtract 1.
- 5 Trace a series of updates line by line.

A white rounded square containing a large teal number '1', indicating the first step in a process.

1

PART 1

Add to it in one step

total += 8 means total = total + 8



A **compound assignment operator** does the arithmetic between the variable's value and the value on the right, then assigns the result to the variable on the left.

ADD TO

```
int total = 25;  
total += 8;  
System.out.println(total);
```

OUTPUT

33

Long form (1.4)	Short form
<code>total = total + 8;</code>	<code>total += 8;</code>
<code>total = total - 8;</code>	<code>total -= 8;</code>

Same result. The variable name is written once.

WORKED EXAMPLE

A running total

SHOPPING CART

```
int cart = 0;  
int shirt = 15;  
int socks = 4;  
cart += shirt;  
cart += socks;  
System.out.println(cart);
```

OUTPUT

19

Line	Works out	cart
cart = 0	-	0
cart += shirt	0 + 15	15
cart += socks	15 + 4	19

Each update starts from the value the variable holds **right now**.

WATCH OUT

=+ is not +=



The arithmetic symbol comes **first**. Written the other way, += just stores a positive number, and the compiler does not complain.

TYPO: +=

```
int gems = 10;  
gems += 4;  
System.out.println(gems);
```

OUTPUT

4

NOT 14

Java reads line 2 as

`gems = +4;`

The 10 is replaced, not added to. The fix:

`gems += 4;`

A large, white, rounded square containing the number '2' in a bold, teal, sans-serif font.

2

PART 2

The other four operators

One for each arithmetic operator

Statement	Means	n was 20, now
<code>n -= 6;</code>	<code>n = n - 6;</code>	14
<code>n *= 3;</code>	<code>n = n * 3;</code>	60
<code>n /= 6;</code>	<code>n = n / 6;</code>	3
<code>n %= 6;</code>	<code>n = n % 6;</code>	2

WITH INT VALUES

`/=` keeps the whole-number part.

`%=` keeps the remainder.



The five compound assignment operators: `+=` `-=` `*=` `/=` `%=`. Each can be used in place of `=` in numeric expressions.

Several updates in a row

LINE BY LINE

```
int temp = 50;  
temp -= 8;  
temp *= 2;  
temp /= 5;  
temp %= 7;
```



Line 4: 84 / 5 is int division, so temp holds **16**, not 16.8.

Line	Works out	temp
1	-	50
2	50 - 8	42
3	42 * 2	84
4	84 / 5	16
5	16 % 7	2

The same operators work on double

A DOUBLE VARIABLE

```
double price = 18.0;  
price /= 4;  
System.out.println(price);  
price *= 1.5;  
System.out.println(price);
```

OUTPUT

```
4.5  
6.75
```

Variable	x /= 4, x was 18
int x	4
double x	4.5

The variable's type decides whether the decimal part survives.

Trace k

PAUSE AND TRY

Write the value of k after each line.

```
int k = 9;  
k += 3;  
k /= 5;  
k *= 4;  
k %= 3;
```

After line	k
2	
3	
4	
5	

A white rounded square with a teal number '3' inside, tilted slightly to the right.

3

PART 3

The right side comes first

$x -= a + b$ means $x = x - (a + b)$



Java evaluates the **entire right side** first, then combines that one value with the variable. Picture parentheses around the right side.

```
int stock = 30;  
int sold = 4;  
int lost = 2;  
stock -= sold + lost;  
System.out.println(stock);  
int pay = 7;  
pay *= 3 - 1;  
System.out.println(pay);
```

OUTPUT

```
24  
14
```

With the invisible parentheses

```
stock = stock - (sold + lost);
```

```
pay = pay * (3 - 1);
```

Right side first

PAUSE AND TRY

Each line starts with `y = 8`. What does `y` hold afterward?

1`y /= 2 + 2;``y = ____`**2**`y %= 2 + 3;``y = ____`**3**`y += 4 / 2;``y = ____`

4

PART 4

Add or subtract one

++ adds 1, -- subtracts 1



The **post-increment** operator ++ and **post-decrement** operator -- add 1 to or subtract 1 from a numeric variable, and store the new value in it.

```
int visitors = 0;
visitors++;
visitors++;
visitors++;
System.out.println(visitors);
int lives = 3;
lives--;
System.out.println(lives);
```

OUTPUT

3
2

These all add 1 to visitors

```
visitors = visitors + 1;
```

```
visitors += 1;
```

```
visitors++;
```

count + 1 does not change count



An expression like `count + 1` only works out a value. Only an assignment, a compound operator, or `++` changes the variable.

```
int count = 5;  
System.out.println(count + 1);  
System.out.println(count);  
count++;  
System.out.println(count);
```

OUTPUT

```
6  
5  
6
```

LINE BY LINE

Line 2 prints 6, but stores nothing.

Line 3 shows count is still 5.

Line 4 stores 6 in count.

Words to know

Compound assignment

An operator like `+=` that does arithmetic, then stores the result in the variable.

`+=` and `-=`

Add to, or subtract from, the variable's current value.

`*=` and `/=`

Multiply or divide. With two ints, `/=` keeps the whole part.

`%=`

Keep the remainder after dividing.

Post-increment

`x++` adds 1 to the value stored in `x`.

Post-decrement

`x--` subtracts 1 from the value stored in `x`.

Say it shorter



With a partner, rewrite each line in a shorter form.

```
miles = miles + 12;
```

```
cash = cash - price;
```

```
eggs = eggs % 12;
```

```
count = count + 1;
```

Next: the Update in Place puzzle worksheet.