

The int type: whole numbers

1. The int type

If you want to store *whole number* in variables, then you need to use the **int** type.

The word **int** is short for **Integer**, which of course is a good hint that this type lets you store *integer numbers*.

Variables whose type is **int** are capable of storing *integer numbers* ranging from **-2 billion** to **+2 billion**. To be more precise, from **-2,147,483,648** to **+2,147,483,647**.

Interesting fact

These decidedly non-round numbers are related to how the computer's memory is organized.

In Java, **4 bytes** of memory are allocated for the `int` type. Each byte of memory consists of **8 bits**. Each bit can only represent 2 values: 0 or 1. An `int` variable contains **32 bits** and can represent **4,294,967,296** values.

Half of this range was set aside for negative numbers, and the other half for positive numbers. And that's how we get the range from **-2,147,483,648** to **+2,147,483,647**.

2. Creating an int variable

The **int** type is for storing integers. To create a variable in code that can store *integer numbers*, you need to use a statement like this:

```
int name;
```

Declaring an `int` variable

Where name is the name of the variable. Examples:

| Statement | Description |
|-------------------------------|--|
| <code>int x;</code> | An <code>x</code> integer variable is created |
| <code>int count;</code> | A <code>count</code> integer variable is created |
| <code>int currentYear;</code> | A <code>currentYear</code> integer variable is created |

The case of the letters matters. That means the commands `int color` and `int Color` will declare two **different** variables.

And the commands `Int Color` and `INT COLOR` won't make any sense to the **compiler**, causing it to report an error. `int` is a special keyword for the integer type and it must be written in **lowercase**.

3. Shorthand for creating variables

If you need to create many variables of the same type in the same place in a program, you can use this shorthand notation:

```
int name1, name2, name3;
```

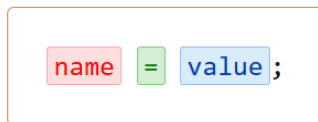
Shorthand for creating multiple variables of the same type

Examples:

| Statements | Shorthand |
|--|-----------------------------------|
| <pre>int x; int y; int z;</pre> | <pre>int x, y, z;</pre> |
| <pre>int count; int totalCount;</pre> | <pre>int count, totalCount;</pre> |
| <pre>int day; int month; int year;</pre> | <pre>int day, month, year;</pre> |

4. Assigning values

To **put** a **value** into an `int variable`, you need to this statement:



Assigning a value to a variable

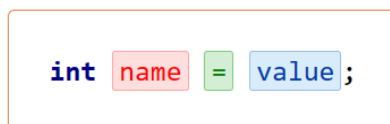
Where the value can be any integer expression. Examples:

| Statement | Note |
|---|--|
| <code>int a;</code> <code>a = 5;</code> | |
| <code>int b;</code> <code>b = 2*1000*1000*1000;</code> | |
| <code>int c;</code> <code>c = -10000000;</code> | |
| <code>int d;</code> <code>d = 3000000000;</code> | This code won't compile, because 3,000,000,000 is greater than the maximum possible value for an <code>int</code> , which is 2,147,483,647 |

5. Shorthand for creating and initializing a variable

You can use a single command to create (declare) a *variable* and *assign* a *value* to it. This is what is done most often, since we usually declare a variable when we need to store a value.

Here's what the command looks like:



Shorthand for creating and initializing a variable

| Statement | Note |
|--|---|
| <code>int a = 5;</code> | |
| <code>int b = 2*1000*1000*1000;</code> | The value of the variable will be 2 billion |
| <code>int c = -10000000;</code> | The value of the variable will be negative 10 million |
| <code>int d = 3000000000;</code> | This code won't compile, because 3,000,000,000 is greater than the maximum possible value for an <code>int</code> : 2,147,483,647 |

You can also declare several variables in a single line. In this case, the command will look like:

```
int name1 = value1, name2 = value2, name3 = value3;
```

Shorthand for creating and initializing multiple variables

Examples:

| Statement | Note |
|--|--|
| <code>int a = 5, b = 10, c = a + b;</code> | <code>a</code> equals <code>5</code> , <code>b</code> equals <code>10</code> , <code>c</code> equals <code>15</code> |