Variable Types

1. Declaring variables

Let's take another look at how to create variables.

To create a variable, you need to write the following command: `type name;`.

Examples:

Command	Explanation
`String s;`	A `String` variable named `s` is created. This variable can store text.
`int x;`	An `int` variable named `x` is created. This variable can store integers.
`int a, b, c;` `int d;`	`Int` variables named `a`, `b`, `c`, and `d` are created. These variables can store integers.

> **Important!**
>
> You cannot create two variables with the same name in the same method. But in different methods, you can. That's like having boxes at different houses.

here are also restrictions on the *name of a variable*. On the one hand, it can be anything. But on the other hand, it cannot contain *spaces* or special characters such as `+`, `-`, etc. It is best to use only *Latin letters* and *numerals* in a variable's name.

Note that in **Java** it matters whether you write **uppercase** or **lowercase** letters. `int a` is not the same as `Int a`.

By the way, in Java you can create a *variable* and simultaneously assign a *value* to it. This saves time and space:

Compact code	Long code equivalent to the code on the left
`int a = 5;`	`int a;` `a = 5;`
`int b = 6;`	`int b;` `b = 6;`
`int c = 7;`	`int c;` `c = 7;`
`int d = c + 1;`	`int d;` `d = c + 1;`
`String s = "I'm Amigo";`	`String s;` `s = "I'm Amigo";`

That way is a lot more compact and clear.

Well, now that we've figured out how to create variables, let's get acquainted with the two most frequently used types in the Java language. They are `int` (integers) and `String` (text/strings).

---

2. The `int` type

An `int` variable can store integers. You can perform various operations (addition, subtraction, multiplication, division, and others) on `int` variables. Examples:

Code	Explanation

`int x = 1;`
`int y = x*2;`
`int z = 5*y*y + 2*y + 3;`
	`x` equals `1`
	`y` equals `2`
	`z` equals `20 + 4 + 3`, which equals `27`

`int a = 5;`
`int b = 1;`
`int c = (a-b) * (a+b);`
	`a` equals `5`
	`b` equals `1`
	`c` equals `4 * 6`, which equals `24`

`int a = 64;`
`int b = a/8;`
`int c = b/4;`
`int d = c*3;`
	`a` equals `64`
	`b` equals `8`
	`c` equals `2`
	`d` equals `6`

3. The `String` type

The `String` type lets you store lines of text, also known as strings.

To assign a string in Java, you need to write the *text of the string* inside *quotation marks*. Example:

Code	Explanation
`String s = "Amigo";`	`s` contains `"Amigo"`
`String s = "123";`	`s` contains `"123"`.
`String s = "Bond 007";`	`s` contains `Bond 007`

Looks easy, right? If so, then here's another interesting fact.

In Java, you can join strings together with a plus sign (+). Example:

Code	Explanation
`String s = "Amigo" + " is the best";`	`s` contains `Amigo is the best`
`String s = "";`	`s` contains an empty string — a string with no characters at all.
`int x = 333;` `String s = "Amigo" + x;`	`s` contains `Amigo333`

Notice that in the last example we concatenated a *string* and a *number*. Everything is simple here too: the number is converted to a string, and then the two strings are glued together.
When *concatenating strings* and *numbers*, you always end up with a *string*.

---

4. Displaying a variable on the screen

It seems that everything is so obvious and simple. Then maybe you can guess right away which *command* you can use to display a variable on the screen?

Indeed, everything is simple. To display something on the screen, we use the `System.out.println()` command. Whatever we want to display, we pass in as an argument.

Code	Screen output
`System.out.println("Amigo");`	Amigo
`System.out.println("Ami" + "go");`	Amigo
`String s = "Amigo";` `System.out.println(s);`	Amigo
`String s = "Am";` `System.out.println(s + "igo");`	Amigo

Hopefully this is a little clearer now. Now we're going to check whether you understood everything correctly. *Practice* is the litmus test: only practice can help you know whether you have understood everything well.