The String type: strings and text

1. The `String` type

The `String` type is one of the most used types in Java. It just might be the most used type. There's a reason why it is so popular: such variables let you store text — and who doesn't want to do that? Additionally, unlike the `int` and `double` types, you can call methods on objects of the `String` type, and these methods do some useful and interesting things.

What's more, all Java objects (all of them!) can be transformed into a `String`. Well, to be more precise, all Java objects can return a text (string) representation of themselves. The name of the `String` type starts with a capital letter, because it is a full-fledged class.

We'll return to this type more than once (it is super useful and interesting), but today we will make a brief introduction.

---

2. Creating `String` variables

The `String` type is designed for storing strings (text). To create a *variable* in code that can store *text*, you need to use a statement like this:

```
String name;
```

Creating a `String` variable

Where `name` is the name of the variable.

Examples:

Statement	Description
`String name;`	A string variable named `name` is created
`String message;`	A string variable named `message` is created
`String text;`	A string variable named `text` is created

Just as with the `int` and `double` types, you can use the shorthand notation to create multiple `String` variables:

```
String name1, name2, name3;
```

Shorthand for creating multiple `String` variables

---

3. Assigning values to `String` variables

To put a value into a `String` variable, you need to this statement:

```
name = "value";
```

Assigning a value to a `String` variable

And now we have come upon the first difference between this type and those we have already studied. All values of the `String` type are *strings of text* and must be enclosed in *double quotes*.

Examples:

Statement	Note
`String name = "Steve";`	The `name` variable contains the text `Steve`
`String city = "New York";`	The `city` variable contains the text `New York`
`String message = "Hello!";`	The `message` variable contains the text `Hello!`

4. Initializing `String` variables

As with the `int` and `double` types, variables of the `String` type can be initialized immediately when they are created. In fact, this is something you can do with all types in Java. So we won't mention it anymore.

```
String name1 = "value1", name2 = "value2", name3 = "value3";
```

Shorthand for creating and initializing variables

```
String name = "Steve", city = "New York", message = "Hello!";
```

Example of a statement that creates and initializes variables

Please note:

The *Java compiler* will complain if you declare a variable without assigning any value to it and then try to use it.

This code won't work:

Statement	Note
`String name;` `System.out.println(name);`	The `name` variable is not initialized. The program won't compile.
`int a;` `a++;`	The `a` variable is not initialized. The program won't compile.
`double x;` `double y = x;`	The `x` variable is not initialized. The program won't compile.