Screen Output

1. Parameters of the `println()` method

A *method body* consists of *commands*. You could even say that a *method* is a group of *commands* that has been given a name, i.e. the method name. Either perspective is accurate.

There are different kinds of commands. The Java language has a command for every occasion. Each command defines some specific action. A *semicolon* goes at the end of each command.

Examples of commands:

Command	**Description (what it does)**
`System.out.println(1);`	Displays a number on the screen: `1`
`System.out.println("Amigo");`	Displays text on the screen: `Amigo`
`System.out.println("Risha & Amigo");`	Displays text on the screen: `Risha & Amigo`

Actually, this is just one command — `System.out.println`. The *arguments* passed to it are contained in the *parentheses*. Depending on the value of the parameters, a command can perform various actions. This is super convenient.

Important:

In Java, size matters in terms whether the letters in a method are uppercase or lowercase. The `System.out.println()` command will work, but `system.out.println()` will not.

If you want to display text, you need to mark it on both sides with *double quotes*.

A single quote looks like this `'`, and a double quote looks like this `"`. A double quote is not two single quotes: please don't get confused by that.

The *double quotes* symbol is the one next to the *Enter* key.

2. Differences between `println()` and `print()`

There are two variations of the command for screen output: `System.out.println()` and `System.out.print()`

If you write the `System.out.println()` command several times, each time the passed text will be displayed *on a new line*. If you use `System.out.print()`, then the text will be displayed *on the same line*. Example:

Commands	What will be displayed
`System.out.println("Amigo");`	`Amigo`
`System.out.println("IsThe");`	`IsThe`
`System.out.println("Best");`	`Best`
`System.out.print("Amigo");`	`AmigoIsThe`
`System.out.println("IsThe");`	`Best`
`System.out.print("Best");`	
`System.out.print("Amigo");`	`AmigoIsTheBest`
`System.out.print("IsThe");`	
`System.out.print("Best");`	

A small note. The `println()` command does not display the text on a new line. Instead, it displays text on the current line — the next text that is displayed will appear on a new line.

The `println()` command displays text and then adds a special invisible *newline* character. As a result, the next text will be displayed *at the beginning of a new line*.

This is what the fully written program will look like, along with a declaration of an `Amigo` class and a `main` method. Keep your eyes on the screen:

```java
public class Amigo
{
   public static void main (String[] args)
   {
      System.out.print("Amigo ");
      System.out.print("The ");
      System.out.print("Best");
   }
}
```

Program with a declaration of the `Amigo` class and `main` method