Code Explanation

In this lesson, we will explore how the "Hello World!" program works in Java.

We'll cover the following

- The given code
- Detailed explanation
  - class HelloWorld
  - void main()
  - { }
  - Print statement
  - Semicolons

The given code

```
class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello World!");
    }
}
```

Detailed explanation

Now let's look at the **Hello World!** sequentially.

`class HelloWorld`

It is written as:

```
class HelloWorld
```

Java is an object-oriented programming language. Therefore, every line that needs to be executed should be inside a class. This line declares a class `HelloWorld`.

- **Declarations and Definitions:** A class declaration provides the name and visibility of a class. In our example, `class HelloWorld` is the class declaration.

  It consists (in this case) of one keyword, `class` followed by the identifier `HelloWorld`.

- The `class` **declaration** is then followed by a block (surrounded by curly braces `{}`), which provides the class's definition.

This means that we are defining a class named `HelloWorld`. Other classes, or in our case, the command line, can refer to the class by this name.

`void main()`

The `main` is the point from where all the Java programs start its execution. In order to execute your program, you must follow the valid signature of `main()` method which is given below:

```
public static void main( String args[] )
```

Don't worry about the details of classes, objects, and the signature of the `main()` method. We will cover it in detail in the upcoming chapters. For now, just remember that we will always write our code inside the class, and this class must contain the `main` method.

> **Note:** In object-oriented programming languages, the program consists of objects. Objects are comprised of methods and the associated data. All objects of the same kind are instances of the same class, which provides a template for all of its objects. For a Java program to run, it must have at least one object, which must define a method called `main()`.

{ }

A block of code is defined with the `{ }` tokens. `{` signifies the start of a block of code, and `}` signifies the end.

> **NOTE:** The `{ }` tokens have other uses as well.

Print statement

```
System.out.println( "Hello World!" );
```

- `System.out.println( "Hello World!" );`

  Gives processor the command to `output` the literal specified in the double quotes ("") and adds a new line to the console. In this case, **Hello World!** is printed.

> **Did you know?** `System.out.print()` prints the content and does not add the new line.

Semicolons

Statements in Java must be terminated with a **semicolon**,

- Just as sentences in English must be terminated with a period.
- Just as sentences in English can span several lines, so can the statements in Java.

In fact, you can use as many spaces and new lines between the words of a Java program as you wish to beautify your code just as spaces are used to justify the text printed on the pages of a book.

Now that we have learned the basics of our code. Let's learn how to compile our code in Java in the next lesson!