Java OutputStream Class

In this tutorial, we will learn about the Java OutputStream and its methods with the help of an example.

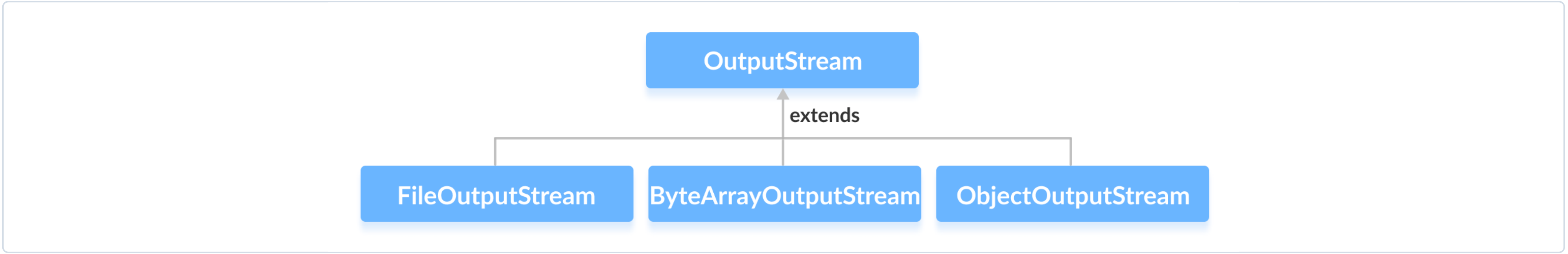
The `OutputStream` class of the `java.io` package is an abstract superclass that represents an output stream of bytes.

Since `OutputStream` is an abstract class, it is not useful by itself. However, its subclasses can be used to write data.

Subclasses of OutputStream

In order to use the functionality of `OutputStream`, we can use its subclasses. Some of them are:

- FileOutputStream
- ByteArrayOutputStream
- ObjectOutputStream



We will learn about all these subclasses in the next tutorial.

Create an OutputStream

In order to create an `OutputStream`, we must import the `java.io.OutputStream` package first. Once we import the package, here is how we can create the output stream.

```
// Creates an OutputStream
OutputStream object = new FileOutputStream();
```

Here, we have created an object of output stream using `FileOutputStream`. It is because `OutputStream` is an abstract class, so we cannot create an object of `OutputStream`.

> **Note**: We can also create the output stream from other subclasses of the `OutputStream` class.

Methods of OutputStream

The `OutputStream` class provides different methods that are implemented by its subclasses. Here are some of the methods:

- `write()` - writes the specified byte to the output stream
- `write(byte[] array)` - writes the bytes from the specified array to the output stream

- `flush()` - forces to write all data present in output stream to the destination
- `close()` - closes the output stream

Example: OutputStream Using FileOutputStream

Here is how we can implement `OutputStream` using the `FileOutputStream` class.

```
import java.io.FileOutputStream;
import java.io.OutputStream;

public class Main {

    public static void main(String args[]) {
        String data = "This is a line of text inside the file.";

        try {
            OutputStream out = new FileOutputStream("output.txt");

            // Converts the string into bytes
            byte[] dataBytes = data.getBytes();

            // Writes data to the output stream
            out.write(dataBytes);
            System.out.println("Data is written to the file.");

            // Closes the output stream
            out.close();
        }

        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

In the above example, we have created an output stream using the `FileOutputStream` class. The output stream is now linked with the file **output.txt**.

```
OutputStream out = new FileOutputStream("output.txt");
```

To write data to the **output.txt** file, we have implemented these methods.

```
output.write();      // To write data to the file
output.close();      // To close the output stream
```

When we run the program, the **output.txt** file is filled with the following content.

```
This is a line of text inside the file.
```