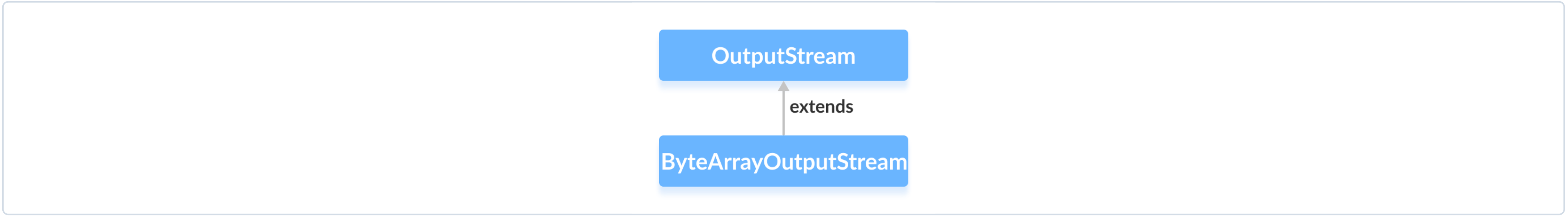
Java ByteArrayOutputStream Class

In this tutorial, we will learn about Java ByteArrayOutputStream and its methods with the help of examples.

The `ByteArrayOutputStream` class of the `java.io` package can be used to write an array of output data (in bytes).

It extends the `OutputStream` abstract class.



> **Note**: In `ByteArrayOutputStream` maintains an internal array of bytes to store the data.

Create a ByteArrayOutputStream

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

```
// Creates a ByteArrayOutputStream with default size
ByteArrayOutputStream out = new ByteArrayOutputStream();
```

Here, we have created an output stream that will write data to an array of bytes with default size 32 bytes. However, we can change the default size of the array.

```
// Creating a ByteArrayOutputStream with specified size
ByteArrayOutputStream out = new ByteArrayOutputStream(int size);
```

Here, the `size` specifies the length of the array.

Methods of ByteArrayOutputStream

The `ByteArrayOutputStream` class provides the implementation of the different methods present in the `OutputStream` class.

write() Method

- `write(int byte)` - writes the specified byte to the output stream
- `write(byte[] array)` - writes the bytes from the specified array to the output stream
- `write(byte[] arr, int start, int length)` - writes the number of bytes equal to `length` to the output stream from an array starting from the position `start`
- `writeTo(ByteArrayOutputStream out1)` - writes the entire data of the current output stream to the specified output stream

Example: ByteArrayOutputStream to write data

```java
import java.io.ByteArrayOutputStream;

class Main {
  public static void main(String[] args) {

    String data = "This is a line of text inside the string.";

    try {
      // Creates an output stream
      ByteArrayOutputStream out = new ByteArrayOutputStream();
      byte[] array = data.getBytes();

      // Writes data to the output stream
      out.write(array);

      // Retrieves data from the output stream in string format
      String streamData = out.toString();
      System.out.println("Output stream: " + streamData);

      out.close();
    }

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

Output

```
Output stream: This is a line of text inside the string.
```

In the above example, we have created a byte array output stream named `output`.

```
ByteArrayOutputStream output = new ByteArrayOutputStream();
```

To write the data to the output stream, we have used the `write()` method.

> **Note**: The `getBytes()` method used in the program converts a string into an array of bytes.

Access Data from ByteArrayOutputStream

- `toByteArray()` - returns the array present inside the output stream

- `toString()` - returns the entire data of the output stream in string form

For example,

```
import java.io.ByteArrayOutputStream;

class Main {
  public static void main(String[] args) {
    String data = "This is data.";

    try {
      // Creates an output stream
      ByteArrayOutputStream out = new ByteArrayOutputStream();

      // Writes data to the output stream
      out.write(data.getBytes());

      // Returns an array of bytes
      byte[] byteData = out.toByteArray();
      System.out.print("Data using toByteArray(): ");
      for(int i=0; i<byteData.length; i++) {
        System.out.print((char)byteData[i]);
      }

      // Returns a string
      String stringData = out.toString();
      System.out.println("\nData using toString(): " + stringData);

      out.close();
    }

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

Output

```
Data using toByteArray(): This is data.
Data using toString(): This is data.
```

In the above example, we have created an array of bytes to store the data returned by the `toByteArray()` method.

We then have used the for loop to access each byte from the array. Here, each byte is converted into the corresponding character using typecasting.

---

close() Method

To close the output stream, we can use the `close()` method.

However, the `close()` method has no effect in `ByteArrayOutputStream` class. We can use the methods of this class even after the `close()` method is called.

Other Methods of ByteArrayOutputStream

Methods	Descriptions
`size()`	returns the size of the array in the output stream
`flush()`	clears the output stream