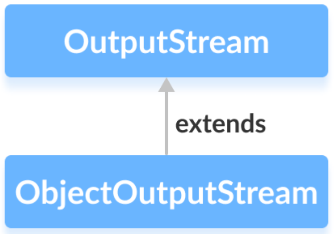
Java ObjectOutputStream Class

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

The `ObjectOutputStream` class of the `java.io` package can be used to write objects that can be read by `ObjectInputStream`.

It extends the `OutputStream` abstract class.



Working of ObjectOutputStream

Basically, the `ObjectOutputStream` encodes Java objects using the class name and object values. And, hence generates corresponding streams. This process is known as serialization.

Those converted streams can be stored in files and can be transferred among networks.

> **Note**: The `ObjectOutputStream` class only writes those objects that implement the `Serializable` interface. This is because objects need to be serialized while writing to the stream

Create an ObjectOutputStream

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

```
// Creates a FileOutputStream where objects from ObjectOutputStream are written
FileOutputStream fileStream = new FileOutputStream(String file);

// Creates the ObjectOutputStream
ObjectOutputStream objStream = new ObjectOutputStream(fileStream);
```

In the above example, we have created an object output stream named `objStream` that is linked with the file output stream named `fileStream`.

Methods of ObjectOutputStream

The `ObjectOutputStream` class provides implementations for different methods present in the `OutputStream` class.

write() Method

- `write()` - writes a byte of data to the output stream
- `writeBoolean()` - writes data in boolean form
- `writeChar()` - writes data in character form
- `writeInt()` - writes data in integer form
- `writeObject()` - writes object to the output stream

Example 1: Java ObjectOutputStream

Let's see how we can use `ObjectOutputStream` to store objects in a file and `ObjectInputStream` to read those objects from the files

```java
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

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

        int data1 = 5;
        String data2 = "This is programiz";

        try {

            FileOutputStream file = new FileOutputStream("file.txt");

            // Creates an ObjectOutputStream
            ObjectOutputStream output = new ObjectOutputStream(file);

            // writes objects to output stream
            output.writeInt(data1);
            output.writeObject(data2);

            // Reads data using the ObjectInputStream
            FileInputStream fileStream = new FileInputStream("file.txt");
            ObjectInputStream objStream = new ObjectInputStream(fileStream);

            System.out.println("Integer data :" + objStream.readInt());
            System.out.println("String data: " + objStream.readObject());
```

Output

```
Integer data: 5
String data: This is programiz
```

In the above example, we have used the `readInt()` method and `readObject()`method to read an integer data and object data from the files.

Here, we have used the `ObjectOutputStream` to write data to the file. We then read the data from the file using the `ObjectInputStream`.

Example 2: Java ObjectOutputStream

Let's take another example,

```java
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

class Dog implements Serializable {

    String name;
    String breed;

    public Dog(String name, String breed) {
        this.name = name;
        this.breed = breed;
    }
}

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

        // Creates an object of Dog class
        Dog dog1 = new Dog("Tyson", "Labrador");

        try {
            FileOutputStream fileOut = new FileOutputStream("file.txt");

            // Creates an ObjectOutputStream
            ObjectOutputStream objOut = new ObjectOutputStream(fileOut);
```

Output

```
Dog Name: Tyson
Dog Breed: Labrador
```

In the above example, we have created

- `ObjectOutputStream` named `objOut` using the `FileOutputStream` named `fileOut`
- `ObjectInputStream` named `objIn` using the `FileInputStream` named `fileIn`.
- An object `dog1` of the `Dog` class.

Here, we have then used the object output stream to write the object to the file. And, the object input stream to read the object from the file.

> **Note**: The `Dog` class implements the `Serializable` interface. It is because the `ObjectOutputStream` only writes objects that can be serialized to the output stream.

Other Methods Of ObjectOutputStream

Methods	Descriptions
`flush()`	clears all the data from the output stream
`drain()`	puts all the buffered data in the output stream

`close()`	closes the output stream