CSCI 2120:

Software Design & Development II

UNIT3: I/O management

io api

ObjectOutputStream

Overview

Introduction

- **ObjectOutputStream in Java** is an output stream that writes serialized objects to a text file. It is responsible for writing objects, primitive type values, and strings to a byte stream.

- In other words, an `ObjectOutputStream` is an output stream that serializes primitive type values, strings, and objects to a stream.

- Since Java `ObjectOutputStream` contains all the methods of *`DataOutputStream`*, we can perform output operations for objects in addition to primitive type values and strings.

- `DataOutputStream` enables to perform output operations for primitive type values and strings. So, we can replace `DataOutputStream` completely with `ObjectOutputStream`.

ObjectOutputStream class declaration

ObjectOutputStream class extends OutputStream and implements ObjectOutput and ObjectStreamConstants interfaces. ObjectOutputStream class also implements Closeable, DataOutput, Flushable, and AutoCloseable interfaces.

The general syntax to declare ObjectOutputStream class in java is as follows:

```
public class ObjectOutputStream
       extends OutputStream
       implements ObjectOutput, ObjectStreamConstants
```

The inheritance diagram for ObjectOutputStream class is as below:

```
java.lang.Object
   java.io.OutputStream
       java.io.ObjectOutputStream
```

ObjectOutput Interface in Java

ObjectOutput in Java is an interface that extends the DataOutput and AutoCloseable interfaces. It supports object serialization in java and defines the writeObject() method for serializing an object.

The general syntax to declare ObjectOutput interface in java is as below:

```
public interface ObjectOutput
       extends DataOutput, AutoCloseable
```

ObjectOutput Interface Methods

In addition to methods inherited from `DataOutput` interface, `ObjectOutput` interface also defines some useful methods that are as follows:

ObjectOutput Interface Methods

Method	Description
void close()	This method closes the invoking stream. Further write attempts will cause an IOException.
void flush()	This method flushes the output stream.
void write(byte[ ] b)	This method writes an array of bytes to the invoking stream.
void write(byte[ ] b, int off, int len)	It writes a subarray of bytes.
void write(int b)	It writes a single byte to the invoking stream. The byte written to the stream is the low-order byte of b.
void writeObject(Object obj)	The writeObject() method writes an object obj to the underlying storage or stream.

Constructors of ObjectOutputStream class

`ObjectOutputStream` class provides two constructors with public and protected access modifiers in Java.

They are as follows:

Constructors of ObjectInputStream class

1. ObjectOutputStream(OutputStream outStream):
This constructor creates an `ObjectOutputStream` object that writes serialized objects from the specified `OutputStream outStream`. It throws an `IOException` if an I/O error occurs.

The general syntax to create an `ObjectOutputStream` object in java is as follows:

```
ObjectOutputStream oos = new ObjectOutputStream(OutputStream outStream);

//For example:
   File file = new File("./objfile.txt");
   FileOutputStream fos = new FileOutputStream(file);
   ObjectOutputStream oos = new ObjectOutputStream(fos);
```

Thus, we can wrap an `ObjectOutputStream` on any `OutputStream` using this constructor.

Constructors of ObjectInputStream class

2. protected ObjectOutputStream()

This constructor creates an `ObjectOutputStream` object.

ObjectOutputStream Methods

In addition to methods inherited from `OutputStream, ObjectOutputStream` class also defines some useful methods to perform output operations on objects. They are as follows:

ObjectOutputStream Methods

Method	Description
void close()	The close() method is used to closing the invoking stream. Further, write tries will cause an IOException.
void defaultWriteObject()	The defaultWriteObject() method is used to write the non-static and non-transient fields of the current class to this stream.
protected void drain()	The drain() method is used to drain any buffered data in ObjectOutputStream.
void flush()	The flush() method is used to flush the stream.
ObjectOutputStream.PutField putFields()	The putField() method is used to get the object used to buffer persistent fields to be written to the stream.
protected Object replaceObject(Object obj)	This method is used to replace one object with another during serialization.
void reset()	The reset() method is used to reset the state of any objects already written to the stream.

ObjectOutputStream Methods

Method	Description
void write(byte[ ] buf)	This method is used to write an array of bytes to the invoking stream.
void write(byte[ ] buf, int off, int len)	This method is used to write a subarray of bytes, starting at offset.
void write(int b)	This method is used to write a single byte to the invoking stream. The byte written is the low-order byte of b.
void writeBoolean(boolean b)	This method is used to write a boolean value to the invoking stream.
void writeByte(int b)	This method is used to write an 8-bit byte to the invoking stream.
void writeBytes(String str)	This method is used to write a string as a sequence of bytes representing str to the invoking stream.
void writeChar(int ch)	This method is used to write a 16-bit char to the invoking stream
void writeChars(String str)	This method is used to write a string as a sequence of chars to the invoking stream.

ObjectOutputStream Methods

Method	Description
protected void writeClassDescriptor(ObjectStreamClass desc)	This method is used to write the specified class descriptor to the ObjectOutputStream.
void writeDouble(double d)	This method is used to write a 64 bit double value to the invoking stream.
void writeFields()	The writeFields() method is used to write the buffered fields to the stream.
void writeFloat(float f)	This method is used to write a 32-bit float value to the invoking stream.
void writeInt(int i)	It is used to write a 32-bit int value to the invoking stream.
void writeLong(long l)	The writeLong() method writes a 64-bit long value to the invoking stream.
void writeObject(Object obj)	The writeObject() method is used to write the specified object to the ObjectOutputStream.

ObjectOutputStream Methods

Method	Description
protected void writeObjectOverride(Object obj)	This method is used by subclasses to override the default writeObject() method.
void writeShort(int s)	The writeShort() method is used to write a 16-bit short value to the invoking stream
protected void writeStreamHeader()	The writeStreamHeader method is used by subclasses to append or prepend their own header to the stream.
void writeUnshared(Object obj)	This method is used to write an “unshared” object to the ObjectOutputStream.
void writeUTF(String str)	The writeUTF() method is used to write primitive values representing string str in modified UTF-8 format to the invoking stream.

Example 1: Read/Write a Student object

Let’s take an example program where we will write student name, roll no, marks, percentage, and the current data to a file named objfile.txt and then will read these data using `ObjectOutputStream` and `ObjectInputStream` classes respectively.

To improve performance, we may add a buffer in the stream. Look at the following source code below.

Example 1: Read/Write a Student object

```java
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Date;
```

Example 1: Read/Write a Student object

```java
public class ObjectOutputStreamTester1 {
   public static void main(String[ ] args) throws IOException, ClassNotFoundException {
       FileOutputStream fos = new FileOutputStream("./src/objfileout1.dat"); // Create fileoutputstream for file objfile.txt.
       BufferedOutputStream bos = new BufferedOutputStream(fos);            // Create bufferedoutputstream pass fos to constructor.
       ObjectOutputStream oos = new ObjectOutputStream(bos);                // Create ObjectOutputStream pass bos to constructor.
       oos.writeUTF("Ted");                                                 // Write a string
       oos.writeInt(10);                                                    // Write int values
       oos.writeInt(484);                                                   // Write int values
       oos.writeDouble(96.55);                                              // Write double value
       oos.writeObject(new java.util.Date());                               // Write object to the file.
       bos.flush();
       bos.close();
       oos.flush();
       oos.close();
       FileInputStream fis = new FileInputStream("./src/objfileout1.dat");  // Create an FileInputStream object for the objfile
       BufferedInputStream bis = new BufferedInputStream(fis);              // Create a bis and pass fis to its constructor.
       ObjectInputStream ois = new ObjectInputStream(bis);                  // Create InputStreamObject, pass fis to constructor.
       String name = ois.readUTF();                                         // Read a string,
       int rollNo = ois.readInt();                                          // Read int value
       int marksObt = ois.readInt();                                        // Read int values
       double per = ois.readDouble();                                       // Read double value
       Date date = (Date) ois.readObject();                                 // Read object from the file.
       System.out.printf("Name: %s, Roll#: %d, Total Marks: %d, Percentage: %.01f, Date: %s\n",
               name, rollNo, marksObt, per, date);                          // Display data read on the console.
       bis.close();
       ois.close();
   }
}
```

Example 1: Read/Write a Student object

Output:

```
Name: Ted, Roll#: 10, Total Marks: 484, Percentage: 96.6, Date: Mon Jul 18 01:23:03 CDT 2022
```

Example 1: Read/Write a Student object

Explanation:

1. In this example program, an `ObjectOutputStream` instance is created to write data into the `object.txt` file. A `string`, `int` values, a `double` value, and an `object` are written to the file.

2. To improve performance, we have used a buffer to wrap the `FileOutputStream`.

```
FileOutputStream fos = new FileOutputStream("./objfile.txt");
BufferedOutputStream bos = new BufferedOutputStream(fos);
ObjectOutputStream oos = new ObjectOutputStream(bos);

//Or, in a single line:
ObjectOutputStream oos = new ObjectOutputStream(new BufferedOutputStream(new
FileOutputStream("./objfile.txt")));
```

Example 1: Read/Write a Student object

Explanation:

3. Multiple objects or primitive type values can be written to the output stream. The objects must be read back from the analogous `ObjectInputStream` with the same types and in the same order as they were written in the `objfile.txt` file.

4. The `readObject()` method of `ObjectInputStream` throws an exception named `ClassNotFoundException` because when *JVM* restores an object, it first loads the class for the object if the class has not been loaded. Since `ClassNotFoundException` is a *checked exception*, the main method declares to throw it.

5. An `ObjectInputStream` instance is created to read input from the `objfile.txt` file. We have to read the data from the file in the same order and format as they were written to the file.

6. A `string`, `int` values, a `double` value, and an `object` are read from the file. Since `readObject()` method returns an `Object`, therefore, it is cast into `Date` and assigned to a variable date of type `Date`.

END