CSCI 2120:
Software Design & Development II

UNIT3: I/O management

io api

ObjectInputStream

Overview

1. Introduction
2. ObjectInputStream class declaration
3. ObjectInput interface
4. ObjectInput interface methods
5. Constructors of ObjectInputStream class
6. ObjectInputStream methods
7. ObjectInputStream examples

Introduction

- **ObjectInputStream in Java** is an input stream that reads serialized objects from a file. It is responsible for reading objects, primitive type values, and strings from a byte stream.

- In other words, an `ObjectInputStream` is an input stream that deserializes primitive type values, strings, and objects from a stream that was previously written in a file using an `ObjectOutputStream`.

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

- `DataInputStream` enables to perform input operations for primitive type values and strings. So, we can replace `DataInputStream` completely with `ObjectInputStream.`

ObjectInputStream class declaration

`ObjectInputStream` class extends `InputStream` and implements `ObjectInput` and `ObjectStreamConstants` interfaces. `ObjectInputStream` class also implements `Closeable, DataInput`, and `AutoCloseable` interfaces.

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

```
public class ObjectInputStream
       extends InputStream
       implements ObjectInput, ObjectStreamConstants
```

The inheritance diagram for `ObjectInputStream` class is as below:

```
java.lang.Object
   java.io.InputStream
       java.io.ObjectInputStream
```

ObjectInput Interface in Java

`ObjectInput` in Java is an interface that extends the `DataInput` and `AutoCloseable` interfaces. It supports object serialization in java and defines the `readObject()` method for deserializing an object.

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

```
public interface ObjectInput
       extends DataInput, AutoCloseable
```

ObjectInput Interface Methods

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

Note:

All the methods defined in `ObjectInput` interface can throw `IOException` when an I/O error occurs. The `readObject( )` method can also throw an exception named `ClassNotFoundException.`

ObjectInput Interface Methods

Method	Description
int available()	This method returns the number of bytes that are now available to read the input buffer.
void close()	This method is used to close the invoking input stream.
int read()	This method reads a byte of data. It returns an integer form of the next available byte of input. –1 is returned when the end of the file is met.
int read(byte[ ] b)	It reads into an array of bytes.
int read(byte[ ] b, int off, int len)	It reads into an array of bytes.
Object readObject()	This method is used to read and return an object from invoking stream.
long skip(long n)	This method skips n bytes of input.

Constructors of ObjectInputStream class

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

They are as follows:

Constructors of ObjectInputStream class

1. ObjectInputStream(InputStream inStream):
This constructor creates an `ObjectInputStream` object that reads serialized objects from the specified `InputStream inStream`. It throws an `IOException` if an I/O error occurs such as the file is not opening.

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

```
ObjectInputStream ois = new ObjectInputStream(InputStream inStream);

//For example:
File file = new File("./objfile.txt");
FileInputStream fis = new FileInputStream(file);
ObjectInputStream ois = new ObjectInputStream(fis);
```

Thus, we can wrap an `ObjectInputStream` on any InputStream using this constructor.

Constructors of ObjectInputStream class

2. protected ObjectInputStream():

This constructor creates an `ObjectInputStream` object.

ObjectInputStream Methods

In addition to methods inherited from `InputStream`, `ObjectInputStream` class also defines some useful methods to perform input operations on objects. They are as follows:

Note:

All these methods of `ObjectInputStream` class can throw `IOException` if I/O errors occur while reading from the underlying `InputStream`.

If the end of file is reached then `EOFException` can be thrown.

ObjectInputStream Methods

Method	Description
int available()	This method is used to retrieve the number of bytes available to read in the input buffer.
void close()	This method is used to close the invoking input stream.
void defaultReadObject()	This method reads the non-static and non-transient data fields of the current class from the stream.
protected boolean enableResolveObject(boolean enable)	It is used to enable the stream to allow objects read from the stream to be replaced.
int read()	This method is used to read a byte of data.
double readDouble()	This method reads and returns a 64 bit double from the invoking stream.
int read(byte[ ] buf, int off, int len)	This method reads into an array of bytes.

ObjectInputStream Methods

Method	Description
boolean readBoolean()	This method reads and returns a boolean value from the invoking stream.
byte readByte()	This method reads and returns an 8-bit byte from the invoking stream.
char readChar()	This method reads and returns a 16-bit char from the invoking stream.
protected ObjectStreamClass readClassDescriptor()	This method is used to read a class descriptor from the serialization stream.
ObjectInputStream.GetField readFields()	This method reads the persistent data fields from the stream and makes them available by name.
float readFloat()	This method reads and returns a 32-bit float from the invoking stream.

ObjectInputStream Methods

Method	Description
int readInt( )	It reads and returns a 32-bit int from the invoking stream.
long readLong( )	It reads and returns a 64 bit long from the invoking stream.
final Object readObject( )	The readObject() method reads and returns an object from the invoking stream.
short readShort( )	The readShort() method reads and returns a short from the invoking stream.
void readFully(byte[ ] buf)	The readFully() method reads bytes, blocking until all bytes are read.
void readFully(byte[ ] buf, int off, int len)	This overloaded readFully() method reads bytes, blocking until all bytes are read.
int readUnsignedByte( )	The readUnsignedByte() method reads and returns a 8 bit unsigned byte from the invoking stream.

ObjectInputStream Methods

Method	Description
int readUnsignedShort( )	The readUnsignedShort reads and returns a 16 bit unsigned short from the invoking stream.
String readUTF()	The readUTF() method reads a String in modified UTF-8 format.
int skipBytes(int lenBytes)	It skips lenBytes bytes in the invoking stream, returning the number of bytes actually skipped.
protected void readStreamHeader()	This method allows subclasses to read and verify their own stream headers.
Object readUnshared()	The readUnshared() method reads an “unshared” object from the ObjectInputStream.

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.

Example 1: Read/Write a Student object

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

public class ObjectInputStreamTester1 {
   public static void main(String[] args) throws IOException, ClassNotFoundException {

      FileOutputStream fos = new FileOutputStream("./src/objfilein1.dat"); // Create an file output stream for file object.txt.
      ObjectOutputStream oos = new ObjectOutputStream(fos);                // Create ObjectOutputStream passing fos to constructor.
      oos.writeUTF("Ted");                                                 // Write a string
      oos.writeInt(10);                                                    // int values
      oos.writeInt(484);                                                   // int values
      oos.writeDouble(96.55);                                              // double value and object to the file.
      oos.writeObject(new java.util.Date());                               // object to the file.
      FileInputStream fis = new FileInputStream("./src/objfilein1.dat");   // Create an FileInputStream object for the file.
      ObjectInputStream ois = new ObjectInputStream(fis);                  // Create InputStreamObject passing fis to constructor.
      String name = ois.readUTF();                                         // Read a string
      int rollNo = ois.readInt();                                          // Read int value,
      int marksObt = ois.readInt();                                        // Read int value
      double per = ois.readDouble();                                       // Read double value
      Date date = (Date) ois.readObject();                                 // Read object to the file & Casting into date.
      // Display data read on the console.
      System.out.printf("Name: %s, Roll#: %d, Total Marks: %d, Percentage: %.01f, Date: %s\n", name, rollNo, marksObt, per, date);
      oos.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 may add a buffer in the stream using the following statements:

```
FileOutputStream fos = new FileOutputStream("D:\\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("D:\\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