UNIVERSIDAD DE ALMERÍA

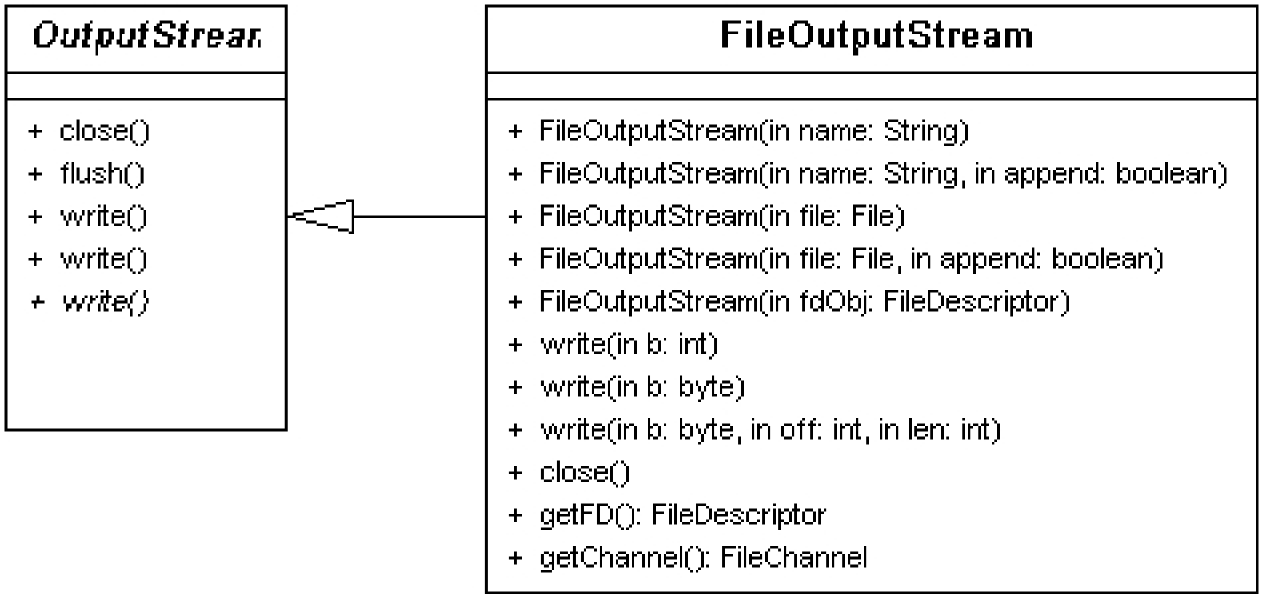
Grado en Ingeniería Informática
Metodología de la programación
2013

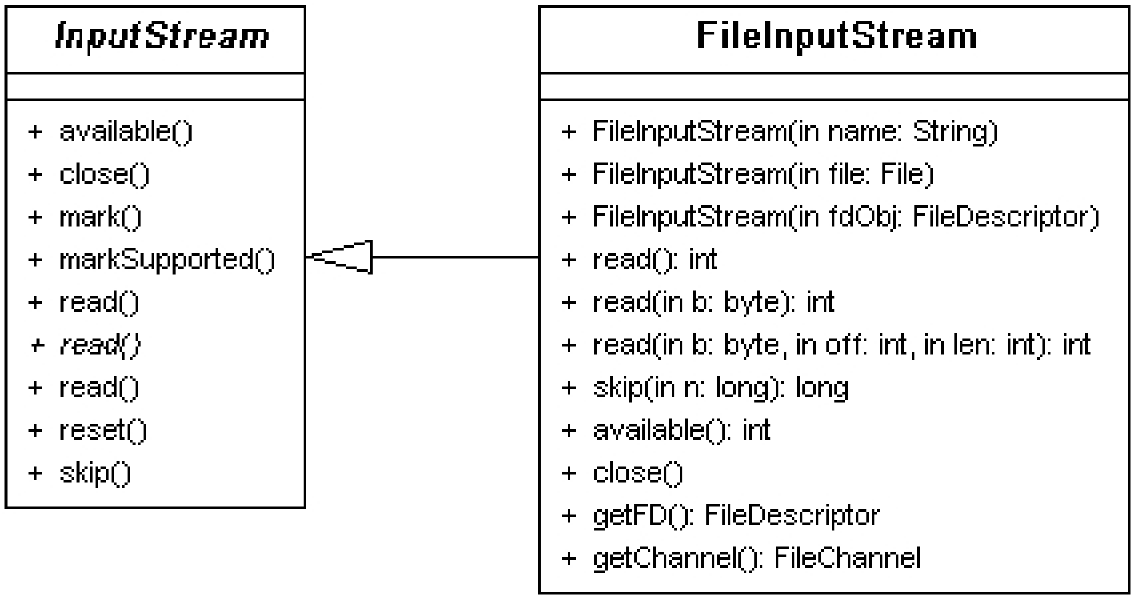
Persistencia. Archivos binarios

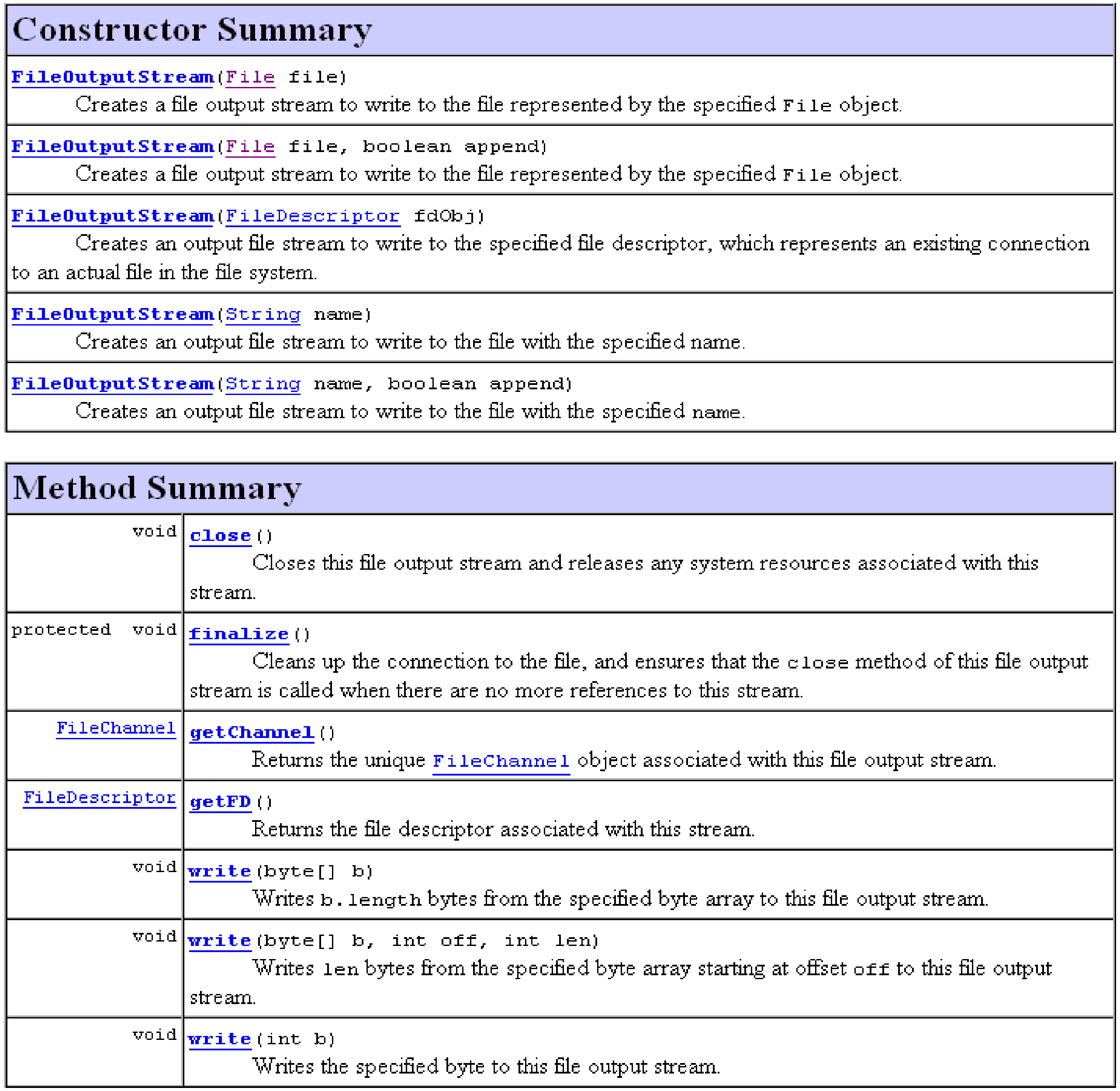
- Librería java.io del JDK

Las clases

- FileInputStream, FileOutputStream
- DataInputStream. DataOutputStream

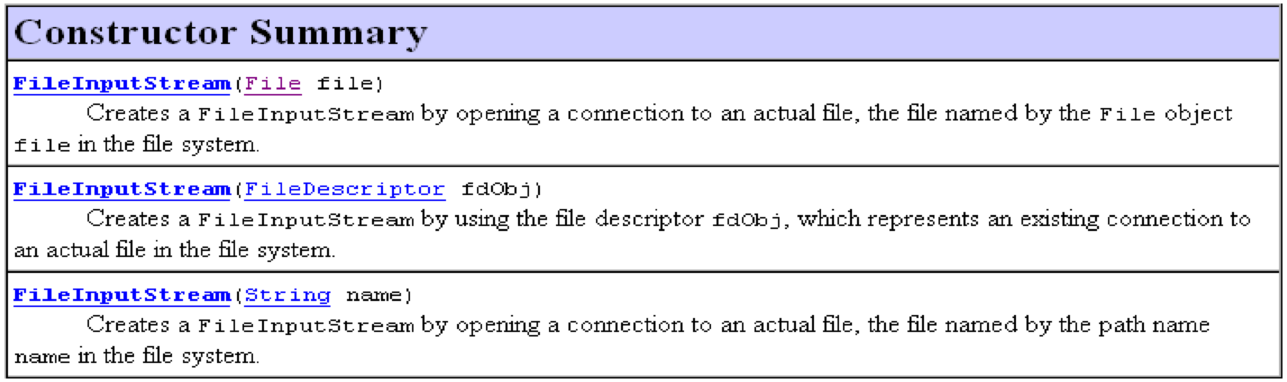

OutputStrear
+ close()
+ flush()
+ write()
+ write()
+ *write()*
FileOutputStream
+ FileOutputStream(in name: String)
+ FileOutputStream(in name: String, in append: boolean)
+ FileOutputStream(in file: File)
+ FileOutputStream(in file: File, in append: boolean)
+ FileOutputStream(in fdObj: FileDescriptor)
+ write(in b: int)
+ write(in b: byte)
+ write(in b: byte, in off: int, in len: int)
+ close()
+ getFD(): FileDescriptor
+ getChannel(): FileChannel


InputStream
+ available()
+ close()
+ mark()
+ markSupported()
+ read()
+ *read()*
+ read()
+ reset()
+ skip()
FileInputStream
+ FileInputStream(in name: String)
+ FileInputStream(in file: File)
+ FileInputStream(in fdObj: FileDescriptor)
+ read(): int
+ read(in b: byte): int
+ read(in b: byte, in off: int, in len: int): int
+ skip(in n: long): long
+ available(): int
+ close()
+ getFD(): FileDescriptor
+ getChannel(): FileChannel

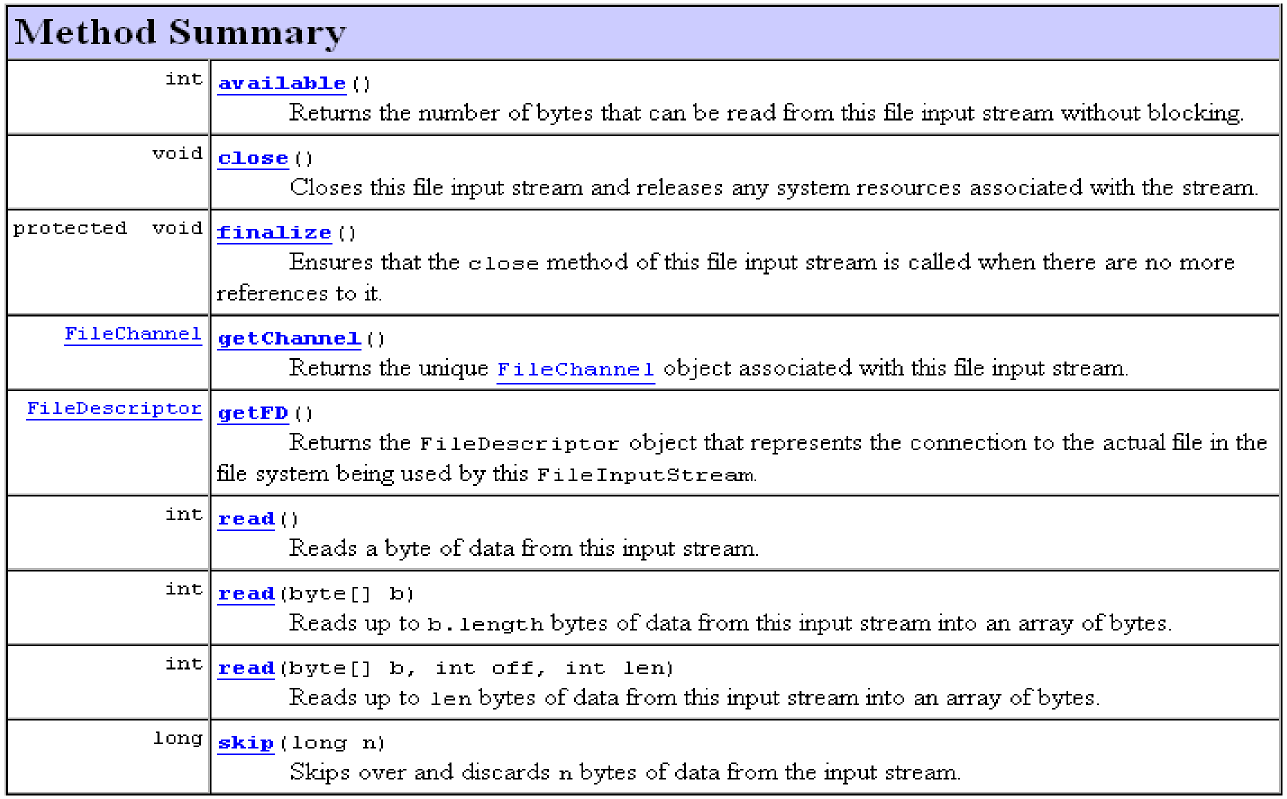


Constructor Summary
`FileOutputStream(File file)` Creates a file output stream to write to the file represented by the specified `File` object.
`FileOutputStream(File file, boolean append)` Creates a file output stream to write to the file represented by the specified `File` object.
`FileOutputStream(FileDescriptor fdObj)` Creates an output file stream to write to the specified file descriptor, which represents an existing connection to an actual file in the file system.
`FileOutputStream(String name)` Creates an output file stream to write to the file with the specified name.
`FileOutputStream(String name, boolean append)` Creates an output file stream to write to the file with the specified name.

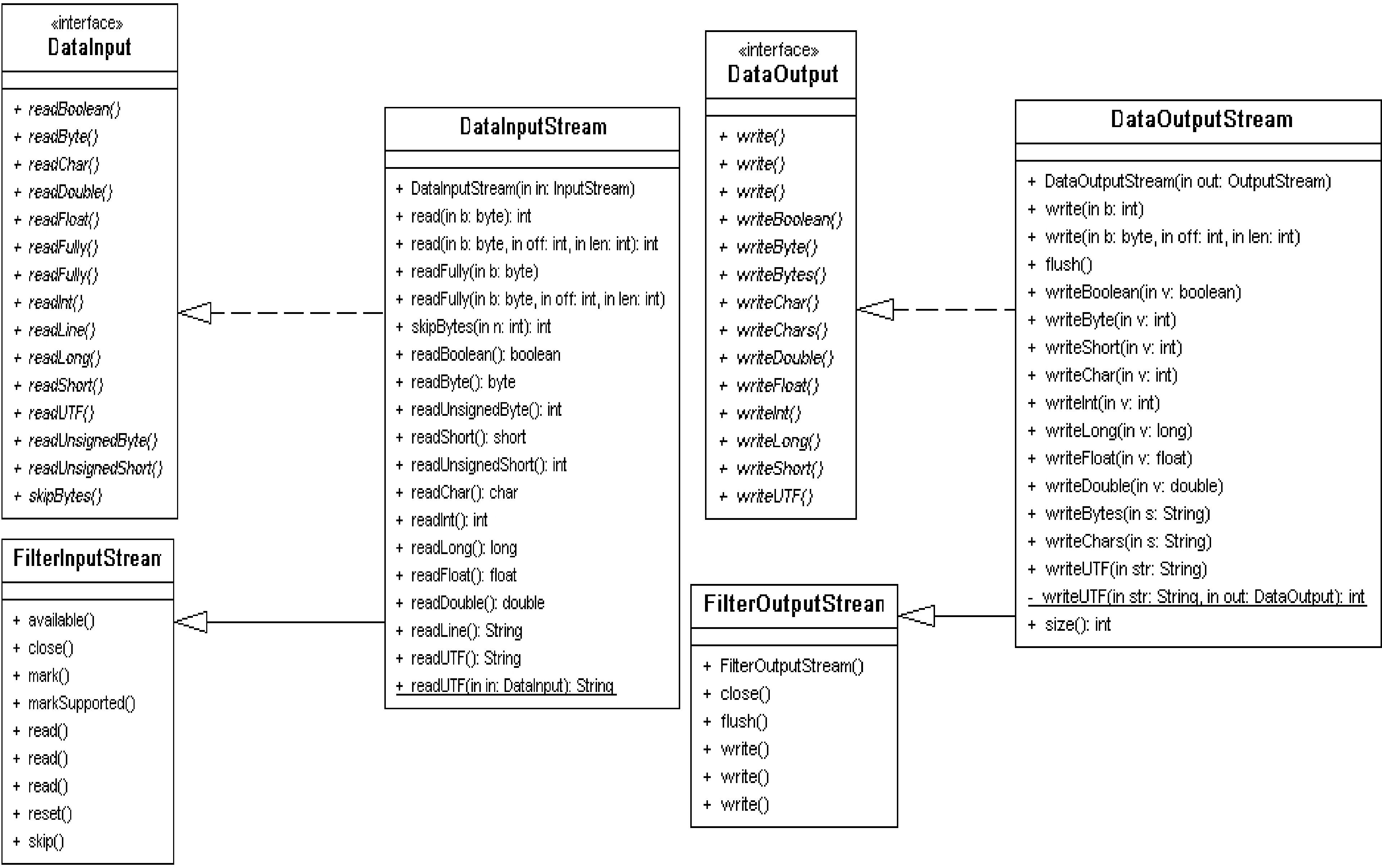
Method Summary	
`void`	`close()` Closes this file output stream and releases any system resources associated with this stream.
`protected void`	`finalize()` Cleans up the connection to the file, and ensures that the `close` method of this file output stream is called when there are no more references to this stream.
`FileChannel`	`getChannel()` Returns the unique `FileChannel` object associated with this file output stream.
`FileDescriptor`	`getFD()` Returns the file descriptor associated with this stream.
`void`	`write(byte[] b)` Writes `b.length` bytes from the specified byte array to this file output stream.
`void`	`write(byte[] b, int off, int len)` Writes `len` bytes from the specified byte array starting at offset `off` to this file output stream.
`void`	`write(int b)` Writes the specified byte to this file output stream.

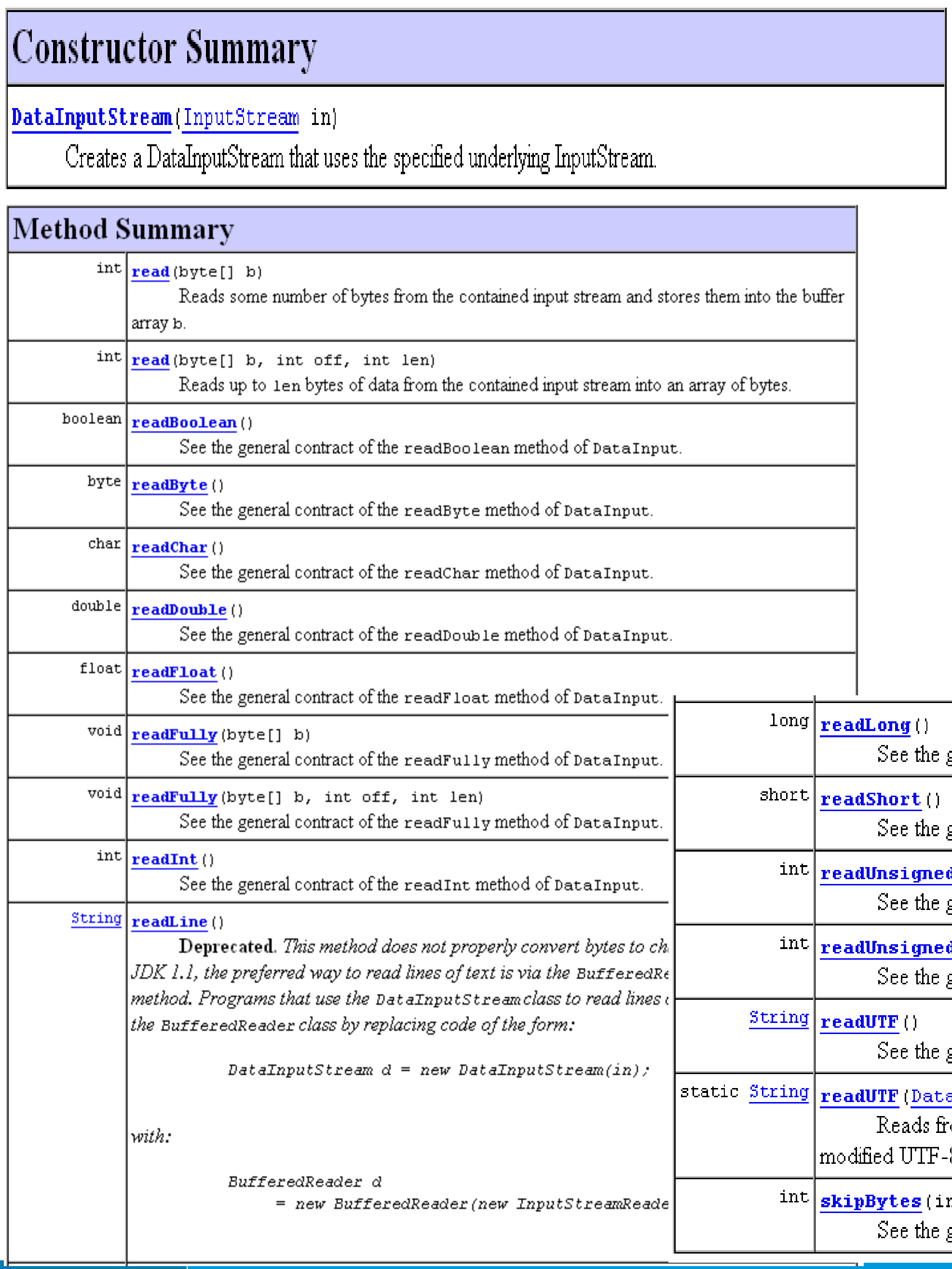


Constructor Summary
FileInputStream(File file) Creates a `FileInputStream` by opening a connection to an actual file, the file named by the `File` object `file` in the file system.
FileInputStream(FileDescriptor fdObj) Creates a `FileInputStream` by using the file descriptor `fdObj`, which represents an existing connection to an actual file in the file system.
FileInputStream(String name) Creates a `FileInputStream` by opening a connection to an actual file, the file named by the path name `name` in the file system.



Method Summary	
int	**available**() Returns the number of bytes that can be read from this file input stream without blocking.
void	**close**() Closes this file input stream and releases any system resources associated with the stream.
protected void	**finalize**() Ensures that the `close` method of this file input stream is called when there are no more references to it.
FileChannel	**getChannel**() Returns the unique `FileChannel` object associated with this file input stream.
FileDescriptor	**getFD**() Returns the `FileDescriptor` object that represents the connection to the actual file in the file system being used by this `FileInputStream`.
int	**read**() Reads a byte of data from this input stream.
int	**read**(byte[] b) Reads up to `b.length` bytes of data from this input stream into an array of bytes.
int	**read**(byte[] b, int off, int len) Reads up to `len` bytes of data from this input stream into an array of bytes.
long	**skip**(long n) Skips over and discards `n` bytes of data from the input stream.


«interface»
DataInput
+ readBoolean()
+ readByte()
+ readChar()
+ readDouble()
+ readFloat()
+ readFully()
+ readFully()
+ readInt()
+ readLine()
+ readLong()
+ readShort()
+ readUTF()
+ readUnsignedByte()
+ readUnsignedShort()
+ skipBytes()
FilterInputStrean
+ available()
+ close()
+ mark()
+ markSupported()
+ read()
+ read()
+ read()
+ reset()
+ skip()
DataInputStream
+ DataInputStream(in in: InputStream)
+ read(in b: byte): int
+ read(in b: byte, in off: int, in len: int): int
+ readFully(in b: byte)
+ readFully(in b: byte, in off: int, in len: int)
+ skipBytes(in n: int): int
+ readBoolean(): boolean
+ readByte(): byte
+ readUnsignedByte(): int
+ readShort(): short
+ readUnsignedShort(): int
+ readChar(): char
+ readInt(): int
+ readLong(): long
+ readFloat(): float
+ readDouble(): double
+ readLine(): String
+ readUTF(): String
+ readUTF(in in: DataInput): String
«interface»
DataOutput
+ write()
+ write()
+ write()
+ writeBoolean()
+ writeByte()
+ writeBytes()
+ writeChar()
+ writeChars()
+ writeDouble()
+ writeFloat()
+ writeInt()
+ writeLong()
+ writeShort()
+ writeUTF()
FilterOutputStrean
+ FilterOutputStream()
+ close()
+ flush()
+ write()
+ write()
+ write()
DataOutputStream
+ DataOutputStream(in out: OutputStream)
+ write(in b: int)
+ write(in b: byte, in off: int, in len: int)
+ flush()
+ writeBoolean(in v: boolean)
+ writeByte(in v: int)
+ writeShort(in v: int)
+ writeChar(in v: int)
+ writeInt(in v: int)
+ writeLong(in v: long)
+ writeFloat(in v: float)
+ writeDouble(in v: double)
+ writeBytes(in s: String)
+ writeChars(in s: String)
+ writeUTF(in str: String)
- writeUTF(in str: String, in out: DataOutput): int
+ size(): int



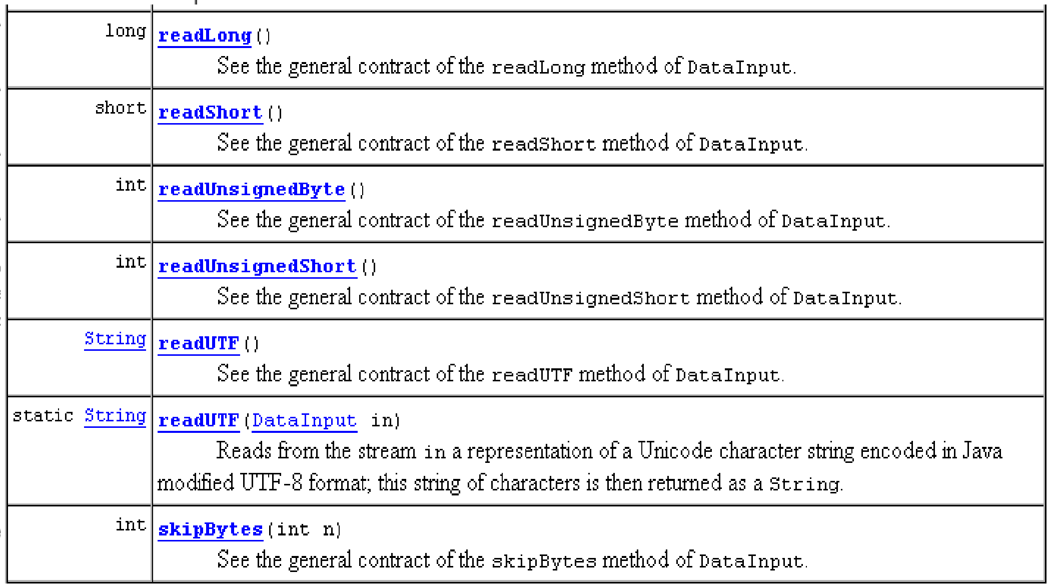
Constructor Summary

DataInputStream(InputStream in)
Creates a DataInputStream that uses the specified underlying InputStream.

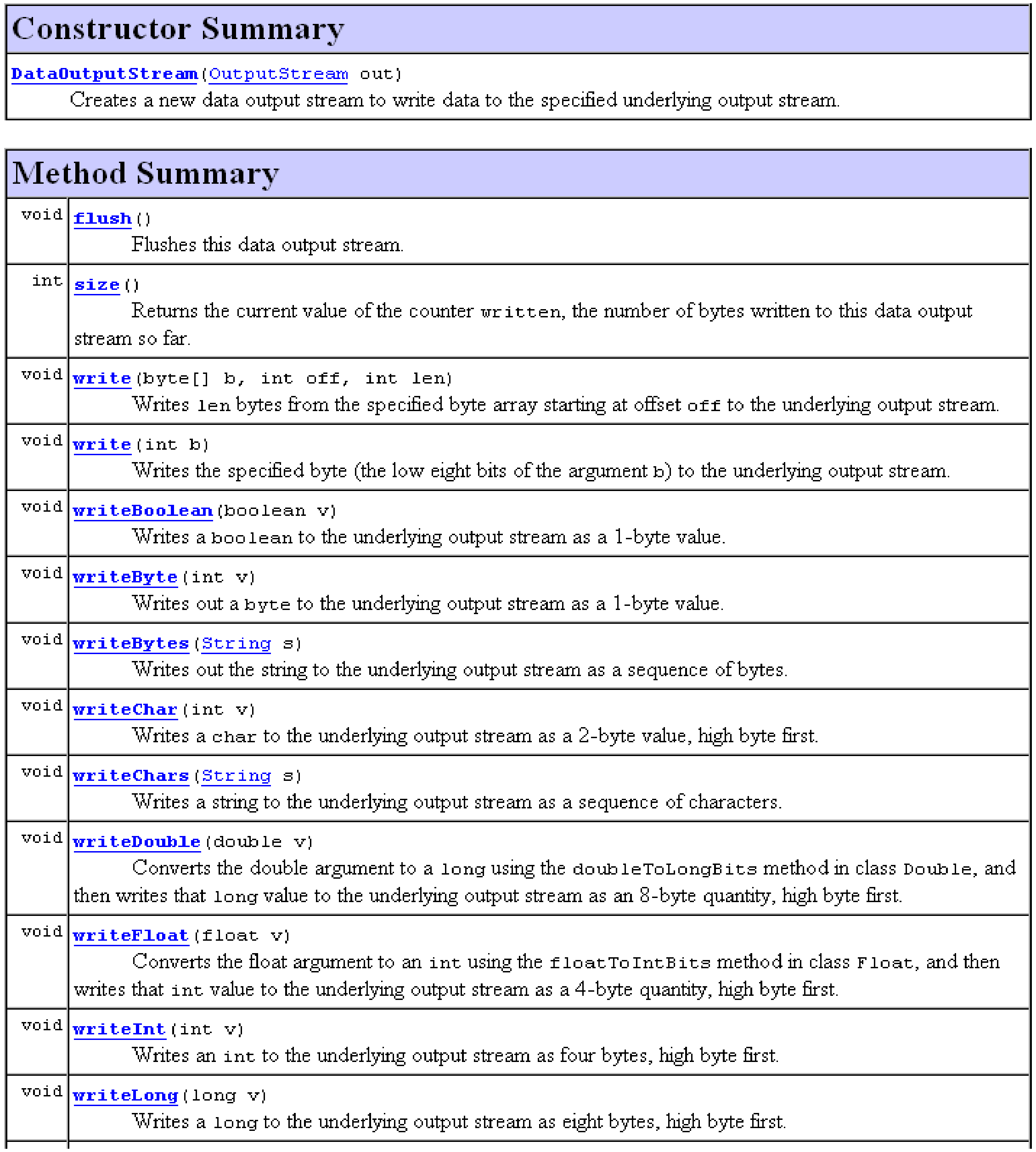
Method Summary

int	**read**(byte[] b) Reads some number of bytes from the contained input stream and stores them into the buffer array b.
int	**read**(byte[] b, int off, int len) Reads up to len bytes of data from the contained input stream into an array of bytes.
boolean	**readBoolean**() See the general contract of the readBoolean method of DataInput.
byte	**readByte**() See the general contract of the readByte method of DataInput.
char	**readChar**() See the general contract of the readChar method of DataInput.
double	**readDouble**() See the general contract of the readDouble method of DataInput.
float	**readFloat**() See the general contract of the readFloat method of DataInput.
void	**readFully**(byte[] b) See the general contract of the readFully method of DataInput.
void	**readFully**(byte[] b, int off, int len) See the general contract of the readFully method of DataInput.
int	**readInt**() See the general contract of the readInt method of DataInput.

String	**readLine**()
	Deprecated. *This method does not properly convert bytes to ch JDK 1.1, the preferred way to read lines of text is via the* BufferedRe *method. Programs that use the* DataInputStream *class to read lines the* BufferedReader *class by replacing code of the form:*
	`DataInputStream d = new DataInputStream(in);`
	with:
	`BufferedReader d`
	`= new BufferedReader(new InputStreamReade`



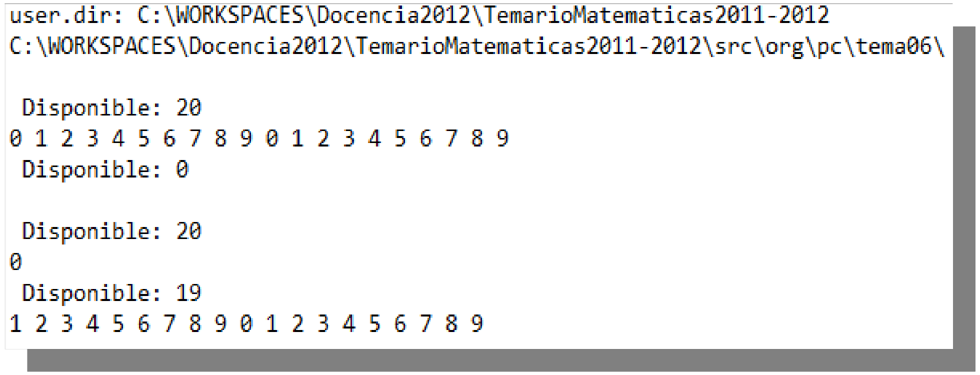
long	**readLong**() See the general contract of the readLong method of DataInput.
short	**readShort**() See the general contract of the readShort method of DataInput.
int	**readUnsignedByte**() See the general contract of the readUnsignedByte method of DataInput.
int	**readUnsignedShort**() See the general contract of the readUnsignedShort method of DataInput.
String	**readUTF**() See the general contract of the readUTF method of DataInput.
static String	**readUTF**(DataInput in) Reads from the stream in a representation of a Unicode character string encoded in Java modified UTF-8 format; this string of characters is then returned as a String.
int	**skipBytes**(int n) See the general contract of the skipBytes method of DataInput.



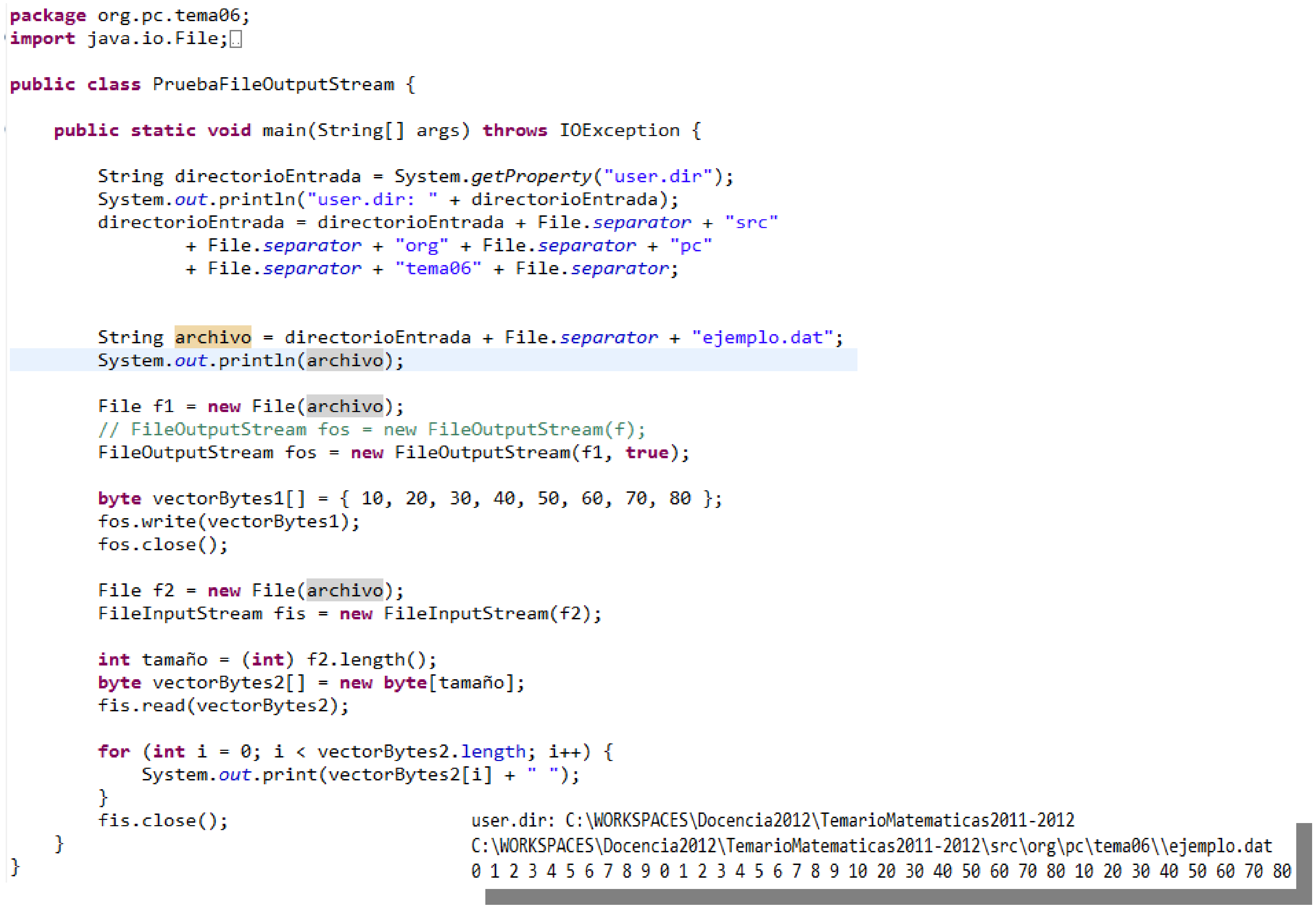
Constructor Summary
`DataOutputStream(OutputStream out)` Creates a new data output stream to write data to the specified underlying output stream.

Method Summary	
`void`	`flush()` Flushes this data output stream.
`int`	`size()` Returns the current value of the counter `written`, the number of bytes written to this data output stream so far.
`void`	`write(byte[] b, int off, int len)` Writes `len` bytes from the specified byte array starting at offset `off` to the underlying output stream.
`void`	`write(int b)` Writes the specified byte (the low eight bits of the argument `b`) to the underlying output stream.
`void`	`writeBoolean(boolean v)` Writes a `boolean` to the underlying output stream as a 1-byte value.
`void`	`writeByte(int v)` Writes out a `byte` to the underlying output stream as a 1-byte value.
`void`	`writeBytes(String s)` Writes out the string to the underlying output stream as a sequence of bytes.
`void`	`writeChar(int v)` Writes a `char` to the underlying output stream as a 2-byte value, high byte first.
`void`	`writeChars(String s)` Writes a string to the underlying output stream as a sequence of characters.
`void`	`writeDouble(double v)` Converts the double argument to a `long` using the `doubleToLongBits` method in class `Double`, and then writes that `long` value to the underlying output stream as an 8-byte quantity, high byte first.
`void`	`writeFloat(float v)` Converts the float argument to an `int` using the `floatToIntBits` method in class `Float`, and then writes that `int` value to the underlying output stream as a 4-byte quantity, high byte first.
`void`	`writeInt(int v)` Writes an `int` to the underlying output stream as four bytes, high byte first.
`void`	`writeLong(long v)` Writes a `long` to the underlying output stream as eight bytes, high byte first.

```java
package org.pc.tema06;
import java.io.File;

public class PruebaFileOutputStreamBasico {
    public static void main(String[] args) throws IOException {

        String directorioEntrada = System.getProperty("user.dir");
        System.out.println("user.dir: " + directorioEntrada);
        directorioEntrada = directorioEntrada + File.separator + "src"
                + File.separator + "org" + File.separator + "pc"
                + File.separator + "tema06" + File.separator;

        System.out.println(directorioEntrada);
        String archivo = directorioEntrada + File.separator + "ejemplo.dat";

        File f0 = new File(archivo);
        FileOutputStream fos = new FileOutputStream(f0);
        // FileOutputStream fos = new FileOutputStream(f,true); para añadir
        for (int i = 256; i < 266; i++) {
            fos.write(i);
        }
        for (byte i = 0; i < 10; i++) {
            fos.write(i);
        }
        fos.close();

        File f1 = new File(archivo);
        FileInputStream fis1 = new FileInputStream(f1);
        int tamaño = (int) f1.length();
        System.out.println("\n Disponible: " + fis1.available());
        for (int i = 0; i < tamaño; i++) {
            System.out.print(fis1.read() + " ");
        }
        System.out.println("\n Disponible: " + fis1.available());
        fis1.close();

        File f2 = new File(archivo);
        FileInputStream fis2 = new FileInputStream(f2);
        System.out.println("\n Disponible: " + fis2.available());
        System.out.print(fis2.read() + " ");
        System.out.println("\n Disponible: " + fis2.available());
        while (fis2.available() > 0) {
            System.out.print(fis2.read() + " ");
        }
        fis2.close();
    }
}
```



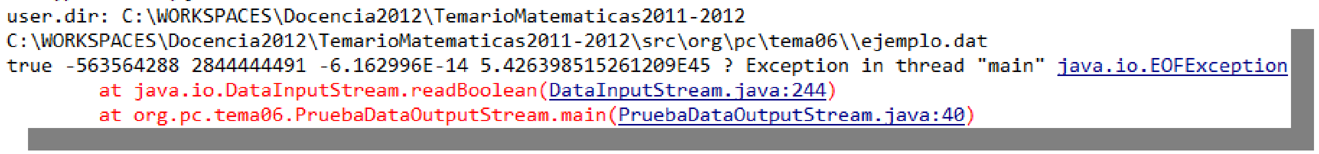
```
user.dir: C:\WORKSPACES\Docencia2012\TemarioMatematicas2011-2012
C:\WORKSPACES\Docencia2012\TemarioMatematicas2011-2012\src\org\pc\tema06\

 Disponible: 20
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
 Disponible: 0

 Disponible: 20
0
 Disponible: 19
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
```



```java
package org.pc.tema06;
import java.io.File;

public class PruebaFileOutputStream {

    public static void main(String[] args) throws IOException {

        String directorioEntrada = System.getProperty("user.dir");
        System.out.println("user.dir: " + directorioEntrada);
        directorioEntrada = directorioEntrada + File.separator + "src"
                + File.separator + "org" + File.separator + "pc"
                + File.separator + "tema06" + File.separator;


        String archivo = directorioEntrada + File.separator + "ejemplo.dat";
        System.out.println(archivo);

        File f1 = new File(archivo);
        // FileOutputStream fos = new FileOutputStream(f);
        FileOutputStream fos = new FileOutputStream(f1, true);

        byte vectorBytes1[] = { 10, 20, 30, 40, 50, 60, 70, 80 };
        fos.write(vectorBytes1);
        fos.close();

        File f2 = new File(archivo);
        FileInputStream fis = new FileInputStream(f2);

        int tamaño = (int) f2.length();
        byte vectorBytes2[] = new byte[tamaño];
        fis.read(vectorBytes2);

        for (int i = 0; i < vectorBytes2.length; i++) {
            System.out.print(vectorBytes2[i] + " ");
        }
        fis.close();
    }
}
```

```
user.dir: C:\WORKSPACES\Docencia2012\TemarioMatematicas2011-2012
C:\WORKSPACES\Docencia2012\TemarioMatematicas2011-2012\src\org\pc\tema06\\ejemplo.dat
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 10 20 30 40 50 60 70 80 10 20 30 40 50 60 70 80
```

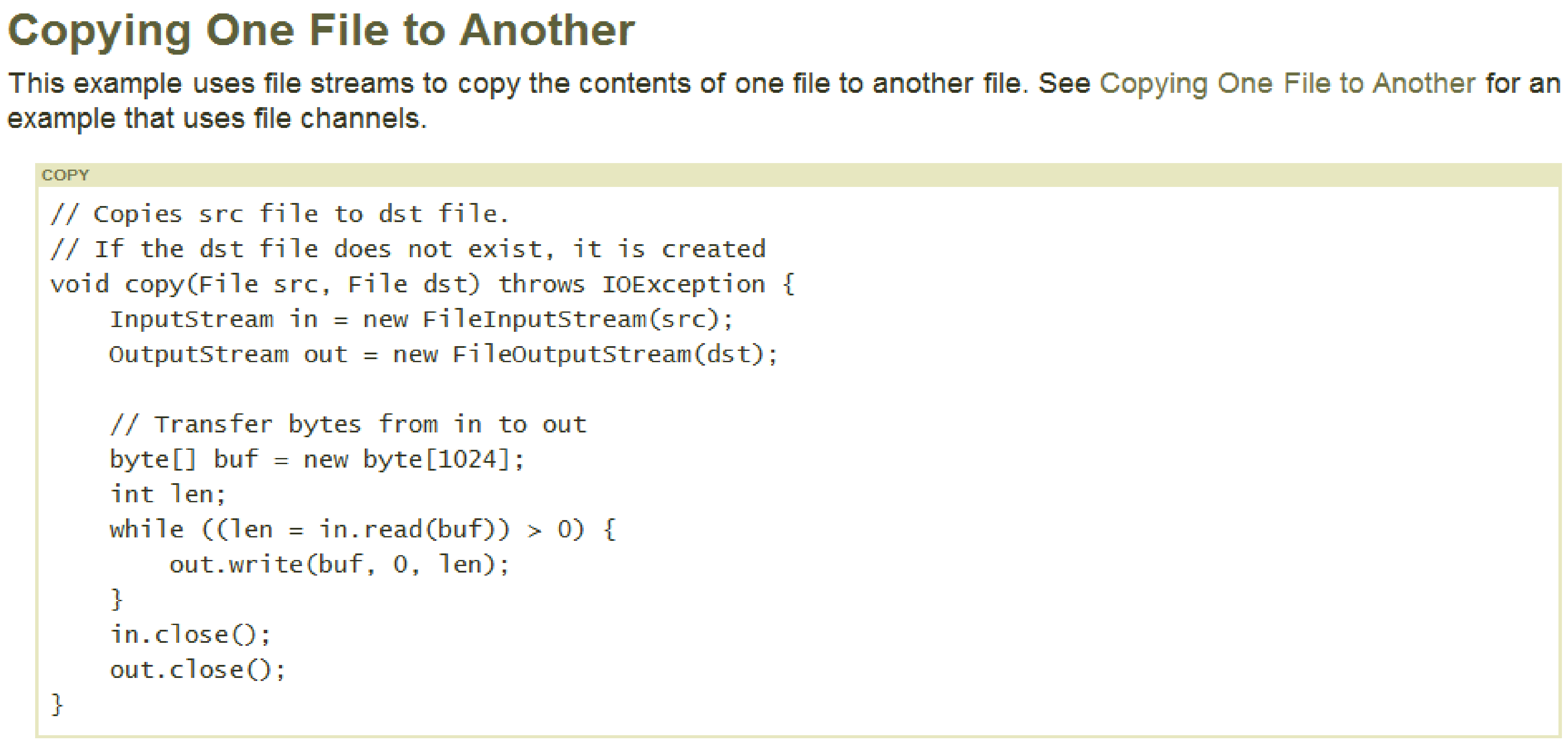
```java
package org.pc.tema06;
import java.io.*;
public class PruebaDataOutputStream {

    public static void main(String[] args) throws IOException {

        String directorioEntrada = System.getProperty("user.dir");
        System.out.println("user.dir: " + directorioEntrada);

        directorioEntrada = directorioEntrada + File.separator + "src"
                + File.separator + "org" + File.separator + "pc"
                + File.separator + "tema06" + File.separator;

        String archivo = directorioEntrada + File.separator + "ejemplo.dat";
        System.out.println(archivo);

        File f1 = new File(archivo);
        FileOutputStream fos = new FileOutputStream(f1);
        DataOutputStream dos = new DataOutputStream(fos);

        dos.writeInt(987654321);
        dos.writeLong(11111111L);
        dos.writeFloat(22222222F);
        dos.writeDouble(3333333D);
        dos.writeChar('A');
        dos.writeBoolean(true);
        dos.close();

        File f2 = new File(archivo);
        FileInputStream fis = new FileInputStream(f2);
        DataInputStream dis = new DataInputStream(fis);

        System.out.print(dis.readBoolean() + " ");
        System.out.print(dis.readInt() + " ");
        System.out.print(dis.readLong() + " ");
        System.out.print(dis.readFloat() + " ");
        System.out.print(dis.readDouble() + " ");
        System.out.print(dis.readChar() + " ");
        System.out.print(dis.readBoolean() + " ");
        dis.close();
    }
}
```



```
user.dir: C:\WORKSPACES\Docencia2012\TemarioMatematicas2011-2012
C:\WORKSPACES\Docencia2012\TemarioMatematicas2011-2012\src\org\pc\tema06\\ejemplo.dat
true -563564288 2844444491 -6.162996E-14 5.426398515261209E45 ? Exception in thread "main" java.io.EOFException
        at java.io.DataInputStream.readBoolean(DataInputStream.java:244)
        at org.pc.tema06.PruebaDataOutputStream.main(PruebaDataOutputStream.java:40)
```

Ejemplos de archivos de texto (javaalmanac)

www.exampledepot.com



Copying One File to Another

This example uses file streams to copy the contents of one file to another file. See Copying One File to Another for an example that uses file channels.

COPY

```
// Copies src file to dst file.
// If the dst file does not exist, it is created
void copy(File src, File dst) throws IOException {
    InputStream in = new FileInputStream(src);
    OutputStream out = new FileOutputStream(dst);

    // Transfer bytes from in to out
    byte[] buf = new byte[1024];
    int len;
    while ((len = in.read(buf)) > 0) {
        out.write(buf, 0, len);
    }
    in.close();
    out.close();
}
```

¡Muchas Gracias