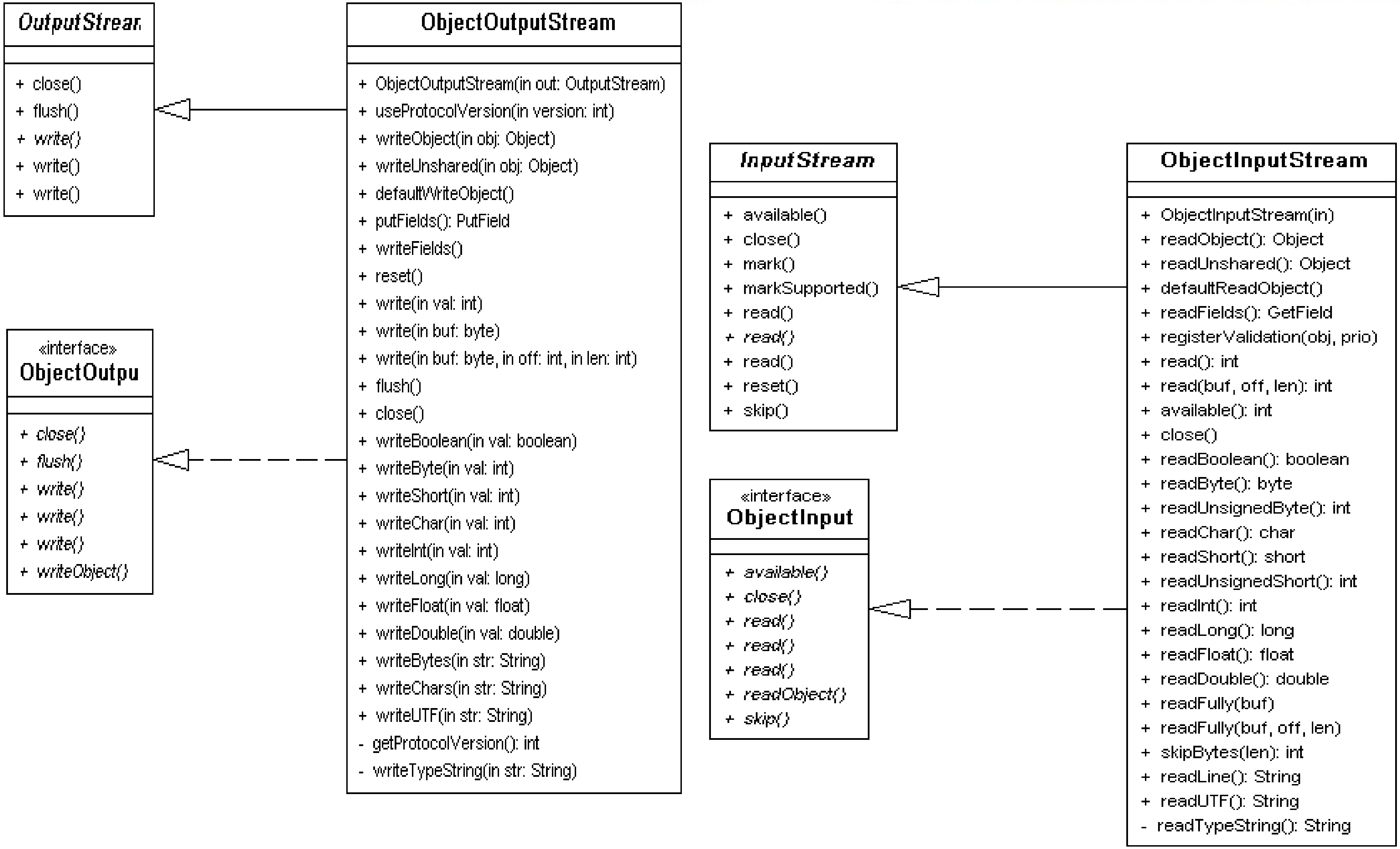
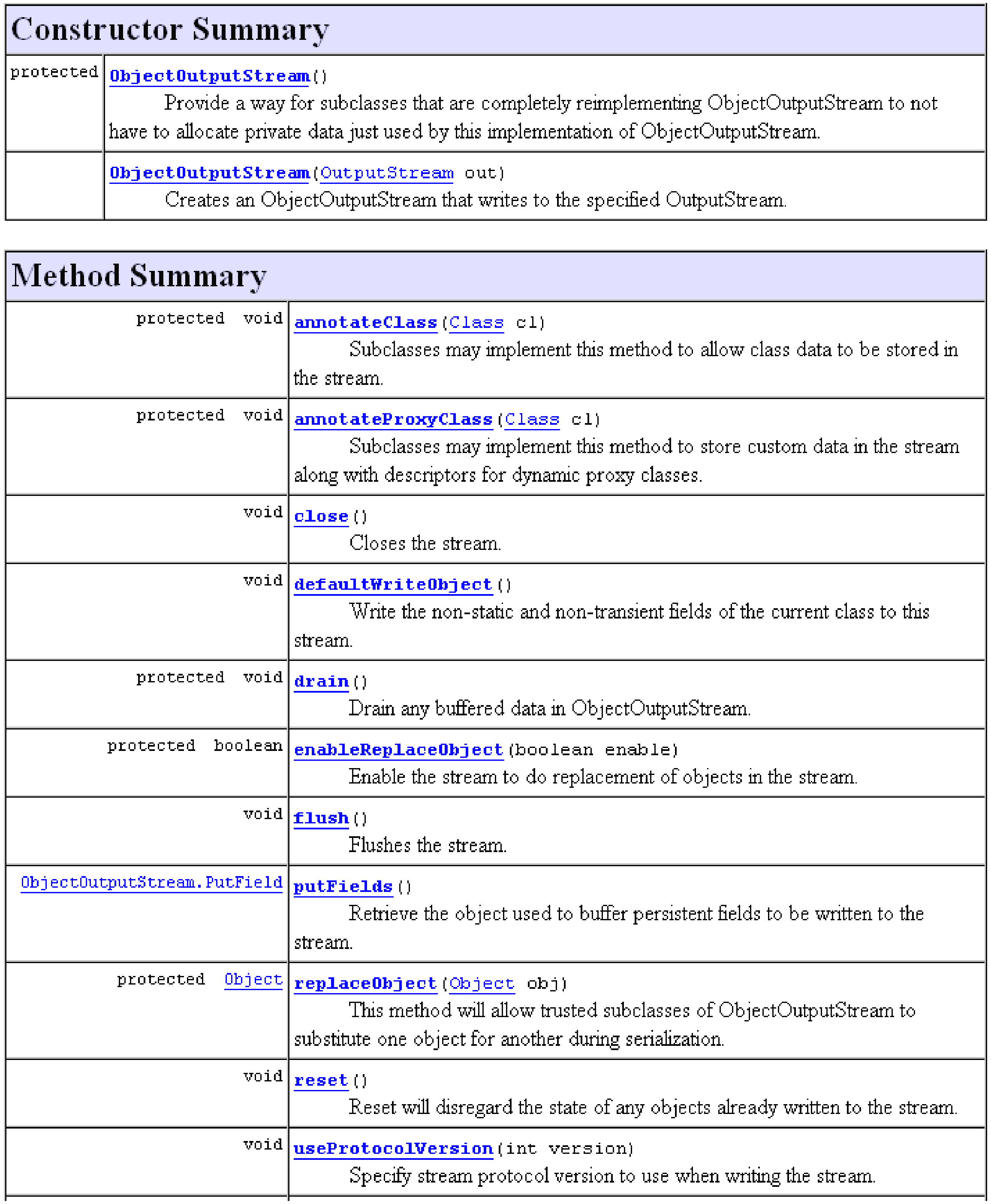


OutputStrean
+ close()
+ flush()
+ *write()*
+ write()
+ write()
ObjectOutputStream
+ ObjectOutputStream(in out: OutputStream)
+ useProtocolVersion(in version: int)
+ writeObject(in obj: Object)
+ writeUnshared(in obj: Object)
+ defaultWriteObject()
+ putFields(): PutField
+ writeFields()
+ reset()
+ write(in val: int)
+ write(in buf: byte)
+ write(in buf: byte, in off: int, in len: int)
+ flush()
+ close()
+ writeBoolean(in val: boolean)
+ writeByte(in val: int)
+ writeShort(in val: int)
+ writeChar(in val: int)
+ writeInt(in val: int)
+ writeLong(in val: long)
+ writeFloat(in val: float)
+ writeDouble(in val: double)
+ writeBytes(in str: String)
+ writeChars(in str: String)
+ writeUTF(in str: String)
- getProtocolVersion(): int
- writeTypeString(in str: String)
«interface»
ObjectOutpu
+ *close()*
+ *flush()*
+ *write()*
+ *write()*
+ *write()*
+ *writeObject()*
InputStream
+ available()
+ close()
+ mark()
+ markSupported()
+ read()
+ *read()*
+ read()
+ reset()
+ skip()
«interface»
ObjectInput
+ *available()*
+ *close()*
+ *read()*
+ *read()*
+ *read()*
+ *readObject()*
+ *skip()*
ObjectInputStream
+ ObjectInputStream(in)
+ readObject(): Object
+ readUnshared(): Object
+ defaultReadObject()
+ readFields(): GetField
+ registerValidation(obj, prio)
+ read(): int
+ read(buf, off, len): int
+ available(): int
+ close()
+ readBoolean(): boolean
+ readByte(): byte
+ readUnsignedByte(): int
+ readChar(): char
+ readShort(): short
+ readUnsignedShort(): int
+ readInt(): int
+ readLong(): long
+ readFloat(): float
+ readDouble(): double
+ readFully(buf)
+ readFully(buf, off, len)
+ skipBytes(len): int
+ readLine(): String
+ readUTF(): String
- readTypeString(): String

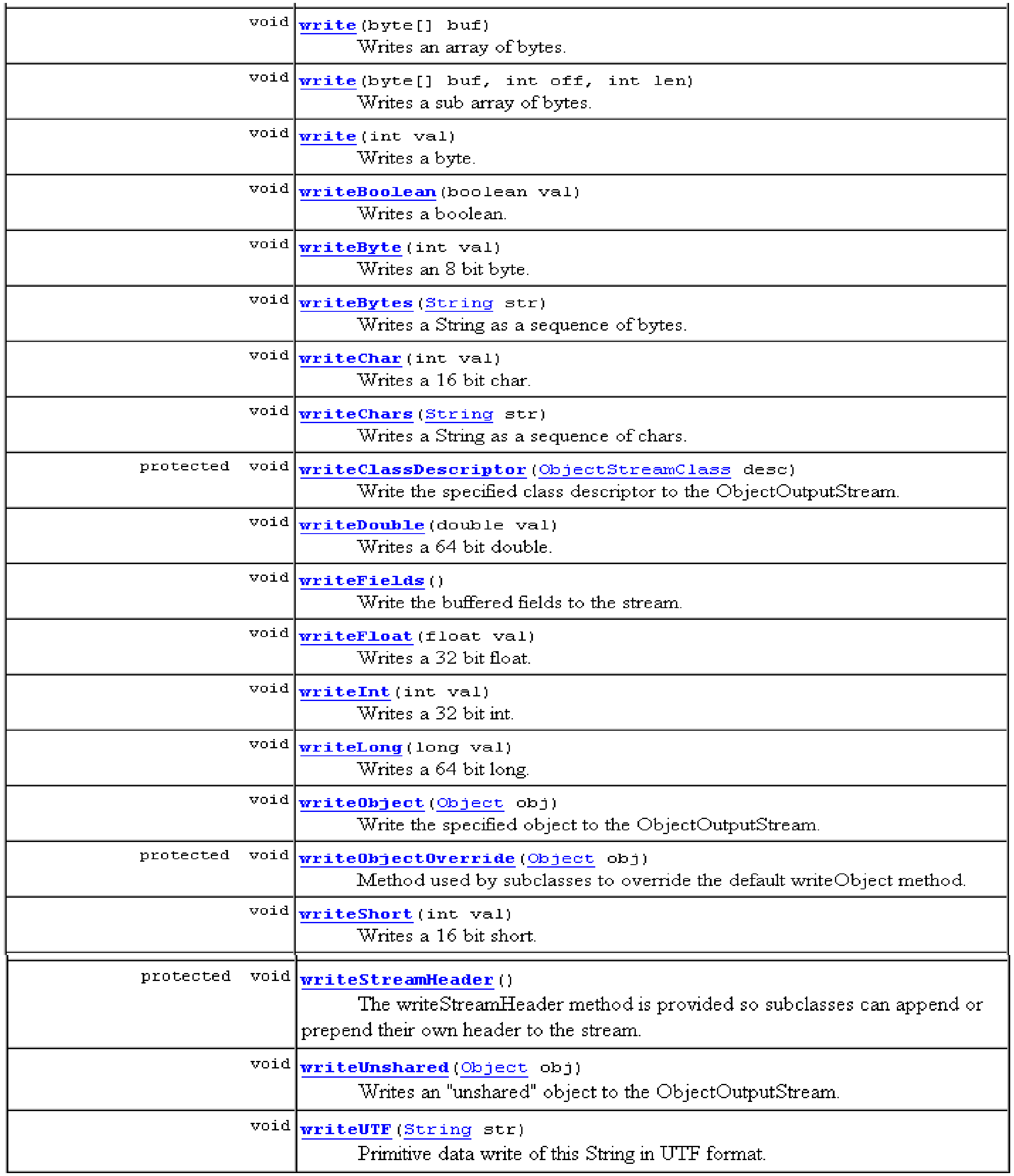


Constructor Summary

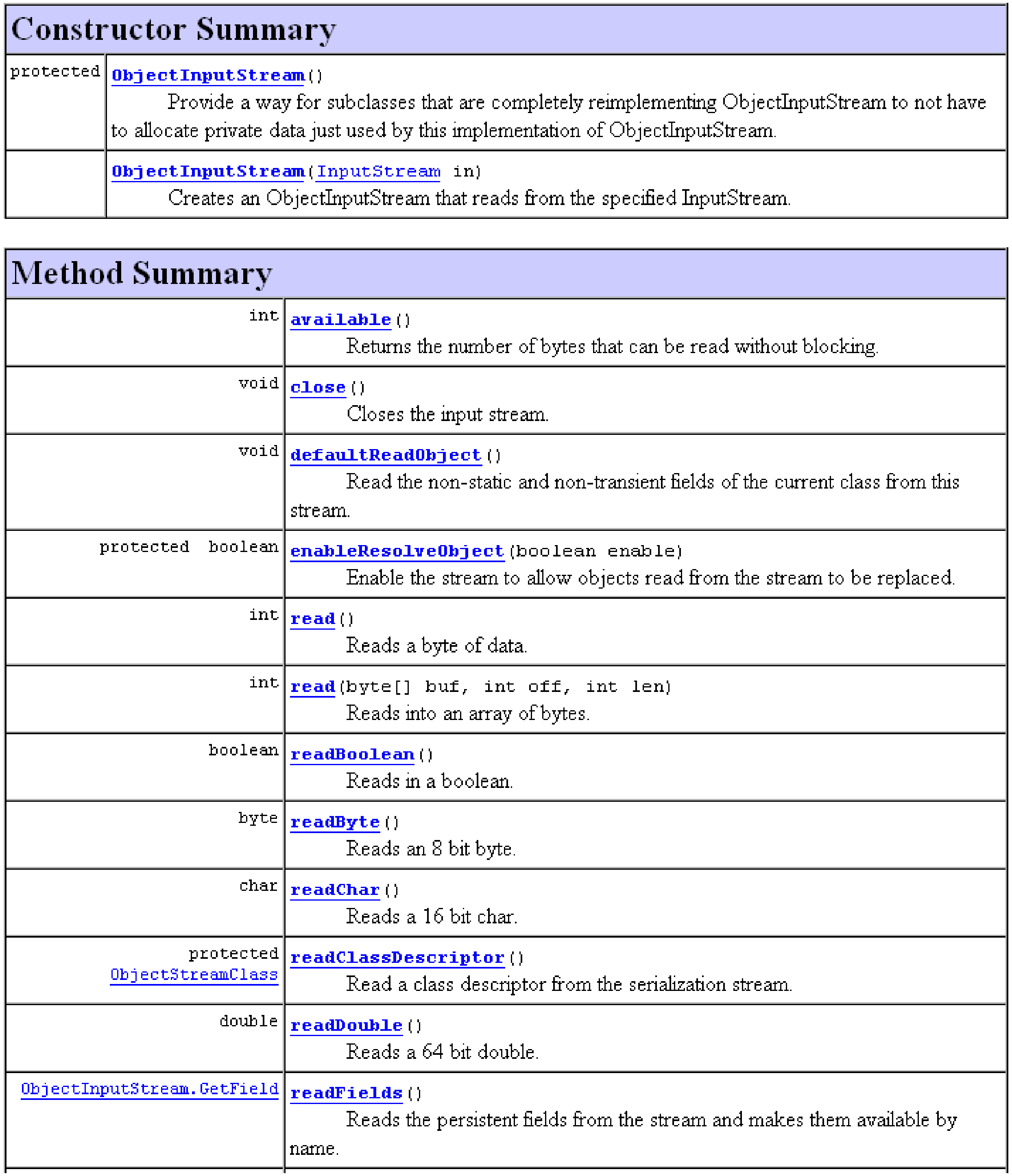
`protected`	`ObjectOutputStream()` Provide a way for subclasses that are completely reimplementing ObjectOutputStream to not have to allocate private data just used by this implementation of ObjectOutputStream.
	`ObjectOutputStream(OutputStream out)` Creates an ObjectOutputStream that writes to the specified OutputStream.

Method Summary

`protected void`	`annotateClass(Class cl)` Subclasses may implement this method to allow class data to be stored in the stream.
`protected void`	`annotateProxyClass(Class cl)` Subclasses may implement this method to store custom data in the stream along with descriptors for dynamic proxy classes.
`void`	`close()` Closes the stream.
`void`	`defaultWriteObject()` Write the non-static and non-transient fields of the current class to this stream.
`protected void`	`drain()` Drain any buffered data in ObjectOutputStream.
`protected boolean`	`enableReplaceObject(boolean enable)` Enable the stream to do replacement of objects in the stream.
`void`	`flush()` Flushes the stream.
`ObjectOutputStream.PutField`	`putFields()` Retrieve the object used to buffer persistent fields to be written to the stream.
`protected Object`	`replaceObject(Object obj)` This method will allow trusted subclasses of ObjectOutputStream to substitute one object for another during serialization.
`void`	`reset()` Reset will disregard the state of any objects already written to the stream.
`void`	`useProtocolVersion(int version)` Specify stream protocol version to use when writing the stream.

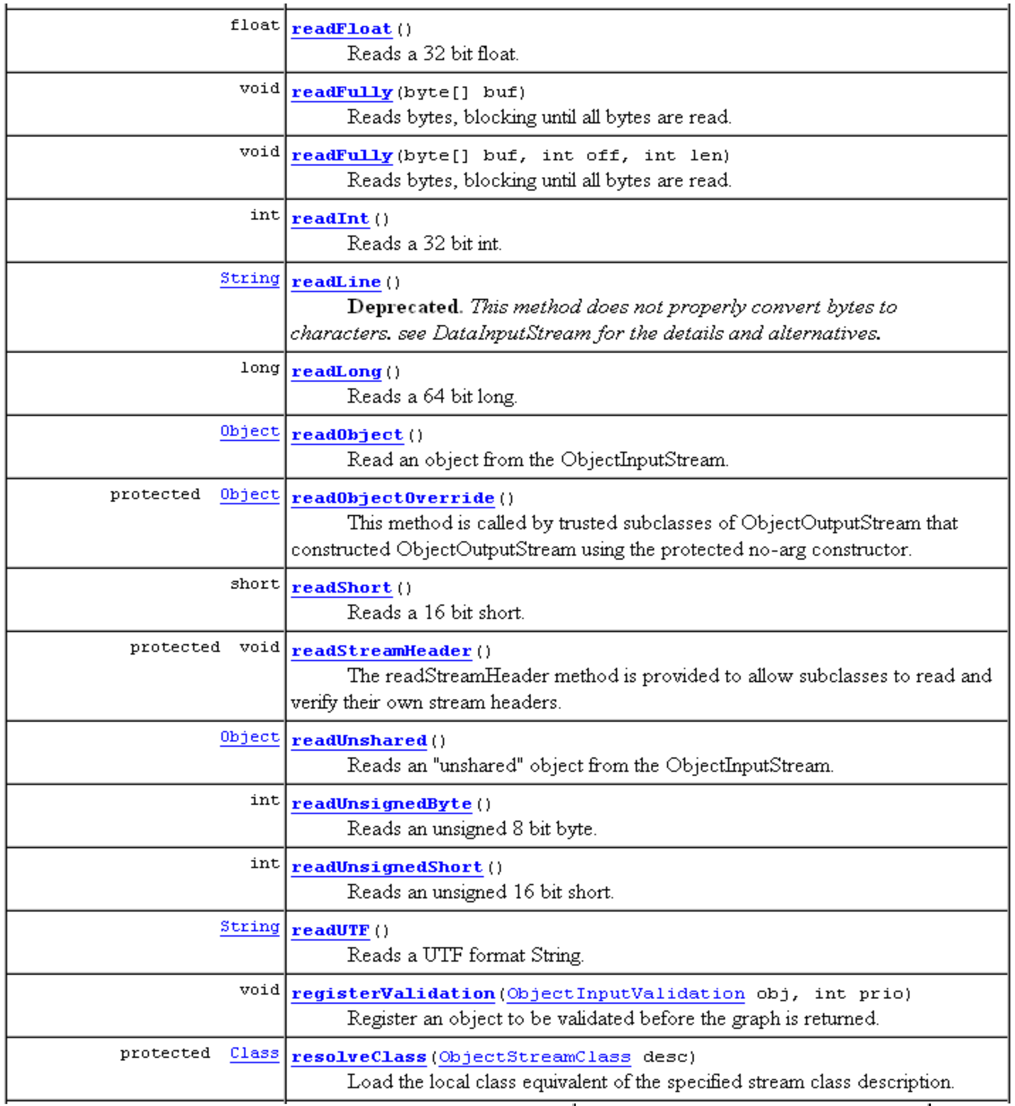


void	write(byte[] buf) Writes an array of bytes.
void	write(byte[] buf, int off, int len) Writes a sub array of bytes.
void	write(int val) Writes a byte.
void	writeBoolean(boolean val) Writes a boolean.
void	writeByte(int val) Writes an 8 bit byte.
void	writeBytes(String str) Writes a String as a sequence of bytes.
void	writeChar(int val) Writes a 16 bit char.
void	writeChars(String str) Writes a String as a sequence of chars.
protected void	writeClassDescriptor(ObjectStreamClass desc) Write the specified class descriptor to the ObjectOutputStream.
void	writeDouble(double val) Writes a 64 bit double.
void	writeFields() Write the buffered fields to the stream.
void	writeFloat(float val) Writes a 32 bit float.
void	writeInt(int val) Writes a 32 bit int.
void	writeLong(long val) Writes a 64 bit long.
void	writeObject(Object obj) Write the specified object to the ObjectOutputStream.
protected void	writeObjectOverride(Object obj) Method used by subclasses to override the default writeObject method.
void	writeShort(int val) Writes a 16 bit short.
protected void	writeStreamHeader() The writeStreamHeader method is provided so subclasses can append or prepend their own header to the stream.
void	writeUnshared(Object obj) Writes an "unshared" object to the ObjectOutputStream.
void	writeUTF(String str) Primitive data write of this String in UTF format.

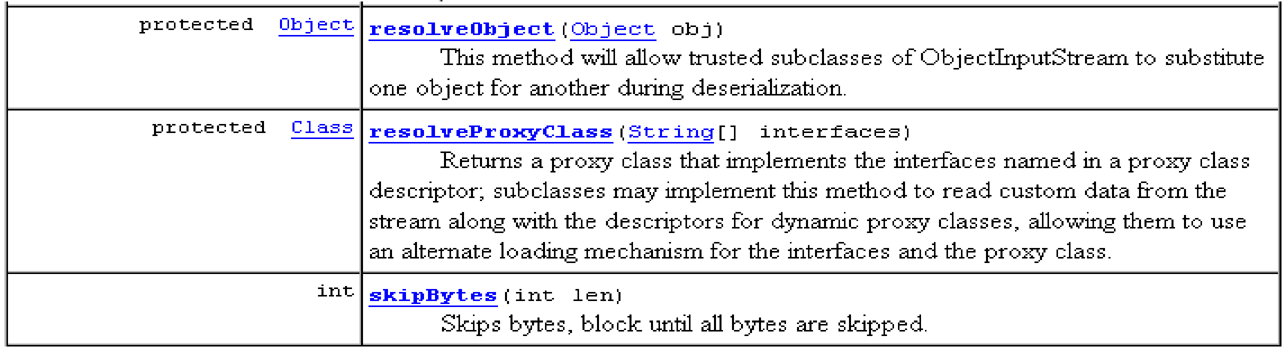


Constructor Summary	
protected	`ObjectInputStream()` Provide a way for subclasses that are completely reimplementing ObjectInputStream to not have to allocate private data just used by this implementation of ObjectInputStream.
	`ObjectInputStream(InputStream in)` Creates an ObjectInputStream that reads from the specified InputStream.

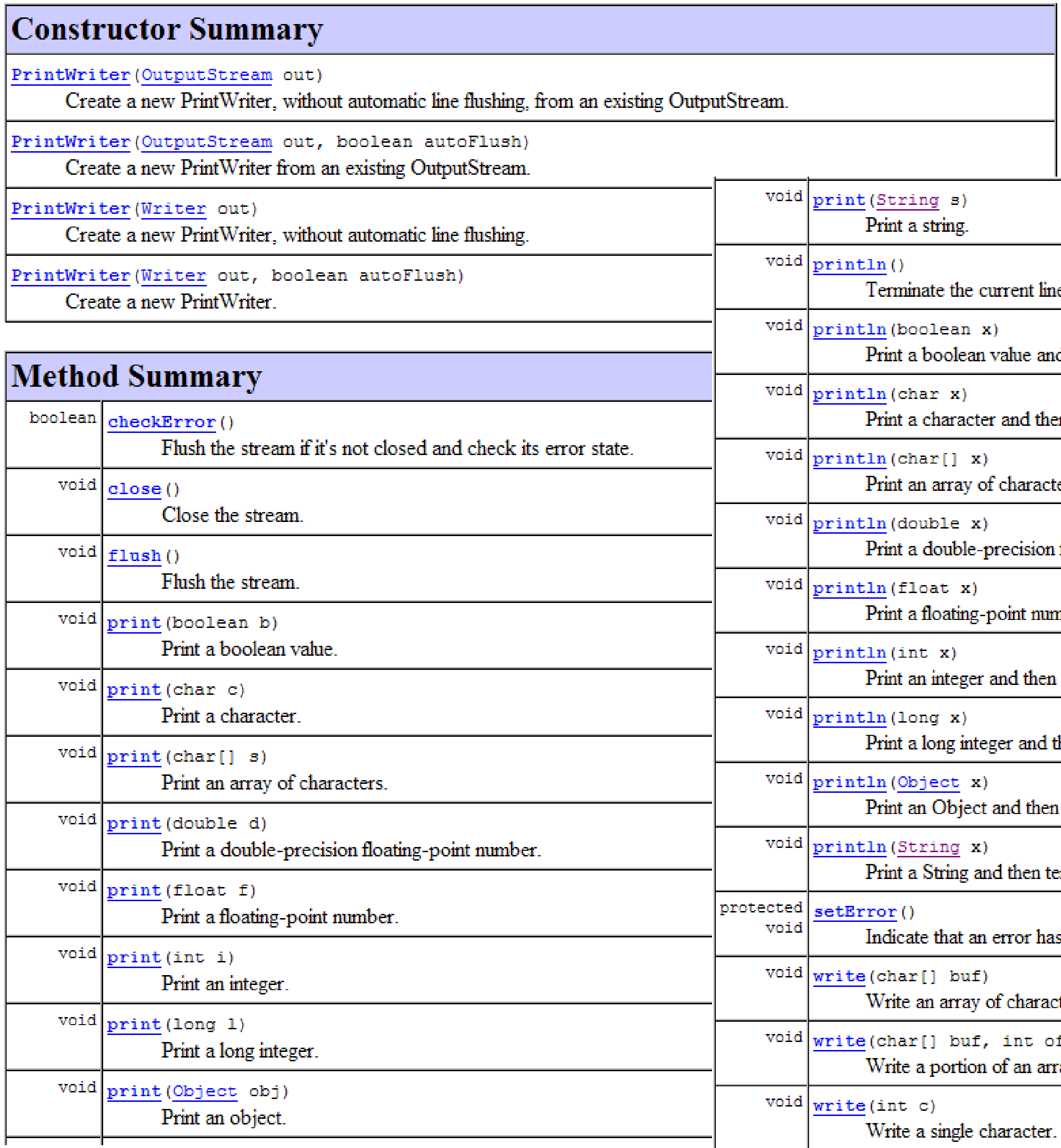
Method Summary	
`int`	`available()` Returns the number of bytes that can be read without blocking.
`void`	`close()` Closes the input stream.
`void`	`defaultReadObject()` Read the non-static and non-transient fields of the current class from this stream.
`protected boolean`	`enableResolveObject(boolean enable)` Enable the stream to allow objects read from the stream to be replaced.
`int`	`read()` Reads a byte of data.
`int`	`read(byte[] buf, int off, int len)` Reads into an array of bytes.
`boolean`	`readBoolean()` Reads in a boolean.
`byte`	`readByte()` Reads an 8 bit byte.
`char`	`readChar()` Reads a 16 bit char.
`protected ObjectStreamClass`	`readClassDescriptor()` Read a class descriptor from the serialization stream.
`double`	`readDouble()` Reads a 64 bit double.
`ObjectInputStream.GetField`	`readFields()` Reads the persistent fields from the stream and makes them available by name.



float	`readFloat()` Reads a 32 bit float.
void	`readFully(byte[] buf)` Reads bytes, blocking until all bytes are read.
void	`readFully(byte[] buf, int off, int len)` Reads bytes, blocking until all bytes are read.
int	`readInt()` Reads a 32 bit int.
String	`readLine()` **Deprecated.** *This method does not properly convert bytes to characters. see DataInputStream for the details and alternatives.*
long	`readLong()` Reads a 64 bit long.
Object	`readObject()` Read an object from the ObjectInputStream.
protected Object	`readObjectOverride()` This method is called by trusted subclasses of ObjectOutputStream that constructed ObjectOutputStream using the protected no-arg constructor.
short	`readShort()` Reads a 16 bit short.
protected void	`readStreamHeader()` The readStreamHeader method is provided to allow subclasses to read and verify their own stream headers.
Object	`readUnshared()` Reads an "unshared" object from the ObjectInputStream.
int	`readUnsignedByte()` Reads an unsigned 8 bit byte.
int	`readUnsignedShort()` Reads an unsigned 16 bit short.
String	`readUTF()` Reads a UTF format String.
void	`registerValidation(ObjectInputValidation obj, int prio)` Register an object to be validated before the graph is returned.
protected Class	`resolveClass(ObjectStreamClass desc)` Load the local class equivalent of the specified stream class description.



protected Object	`resolveObject(Object obj)` This method will allow trusted subclasses of ObjectInputStream to substitute one object for another during deserialization.
protected Class	`resolveProxyClass(String[] interfaces)` Returns a proxy class that implements the interfaces named in a proxy class descriptor; subclasses may implement this method to read custom data from the stream along with the descriptors for dynamic proxy classes, allowing them to use an alternate loading mechanism for the interfaces and the proxy class.
int	`skipBytes(int len)` Skips bytes, block until all bytes are skipped.

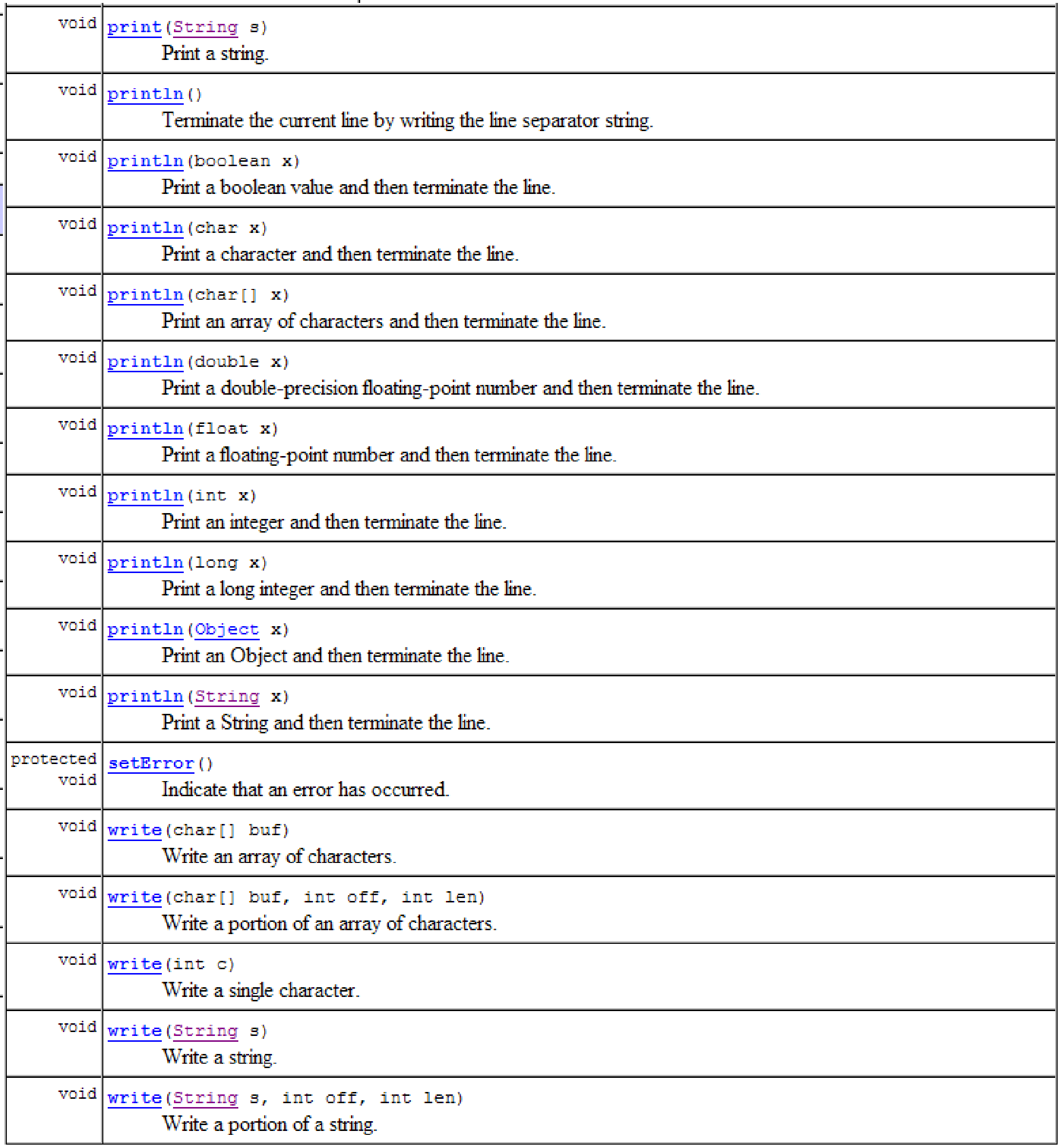


Constructor Summary

`PrintWriter(OutputStream out)` Create a new PrintWriter, without automatic line flushing, from an existing OutputStream.
`PrintWriter(OutputStream out, boolean autoFlush)` Create a new PrintWriter from an existing OutputStream.
`PrintWriter(Writer out)` Create a new PrintWriter, without automatic line flushing.
`PrintWriter(Writer out, boolean autoFlush)` Create a new PrintWriter.

Method Summary

`boolean`	`checkError()` Flush the stream if it's not closed and check its error state.
`void`	`close()` Close the stream.
`void`	`flush()` Flush the stream.
`void`	`print(boolean b)` Print a boolean value.
`void`	`print(char c)` Print a character.
`void`	`print(char[] s)` Print an array of characters.
`void`	`print(double d)` Print a double-precision floating-point number.
`void`	`print(float f)` Print a floating-point number.
`void`	`print(int i)` Print an integer.
`void`	`print(long l)` Print a long integer.
`void`	`print(Object obj)` Print an object.

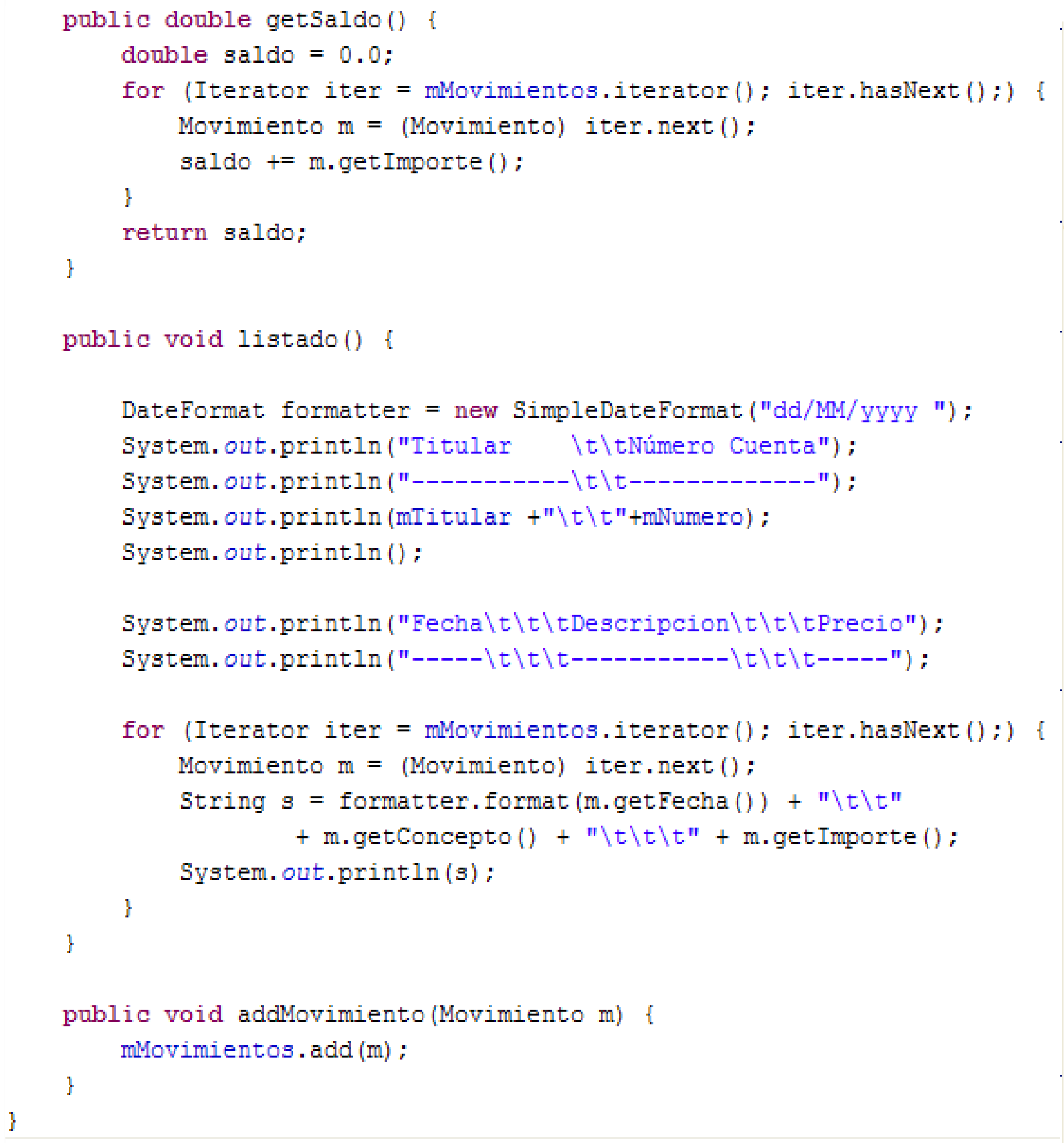


`void`	`print(String s)` Print a string.
`void`	`println()` Terminate the current line by writing the line separator string.
`void`	`println(boolean x)` Print a boolean value and then terminate the line.
`void`	`println(char x)` Print a character and then terminate the line.
`void`	`println(char[] x)` Print an array of characters and then terminate the line.
`void`	`println(double x)` Print a double-precision floating-point number and then terminate the line.
`void`	`println(float x)` Print a floating-point number and then terminate the line.
`void`	`println(int x)` Print an integer and then terminate the line.
`void`	`println(long x)` Print a long integer and then terminate the line.
`void`	`println(Object x)` Print an Object and then terminate the line.
`void`	`println(String x)` Print a String and then terminate the line.
`protected void`	`setError()` Indicate that an error has occurred.
`void`	`write(char[] buf)` Write an array of characters.
`void`	`write(char[] buf, int off, int len)` Write a portion of an array of characters.
`void`	`write(int c)` Write a single character.
`void`	`write(String s)` Write a string.
`void`	`write(String s, int off, int len)` Write a portion of a string.

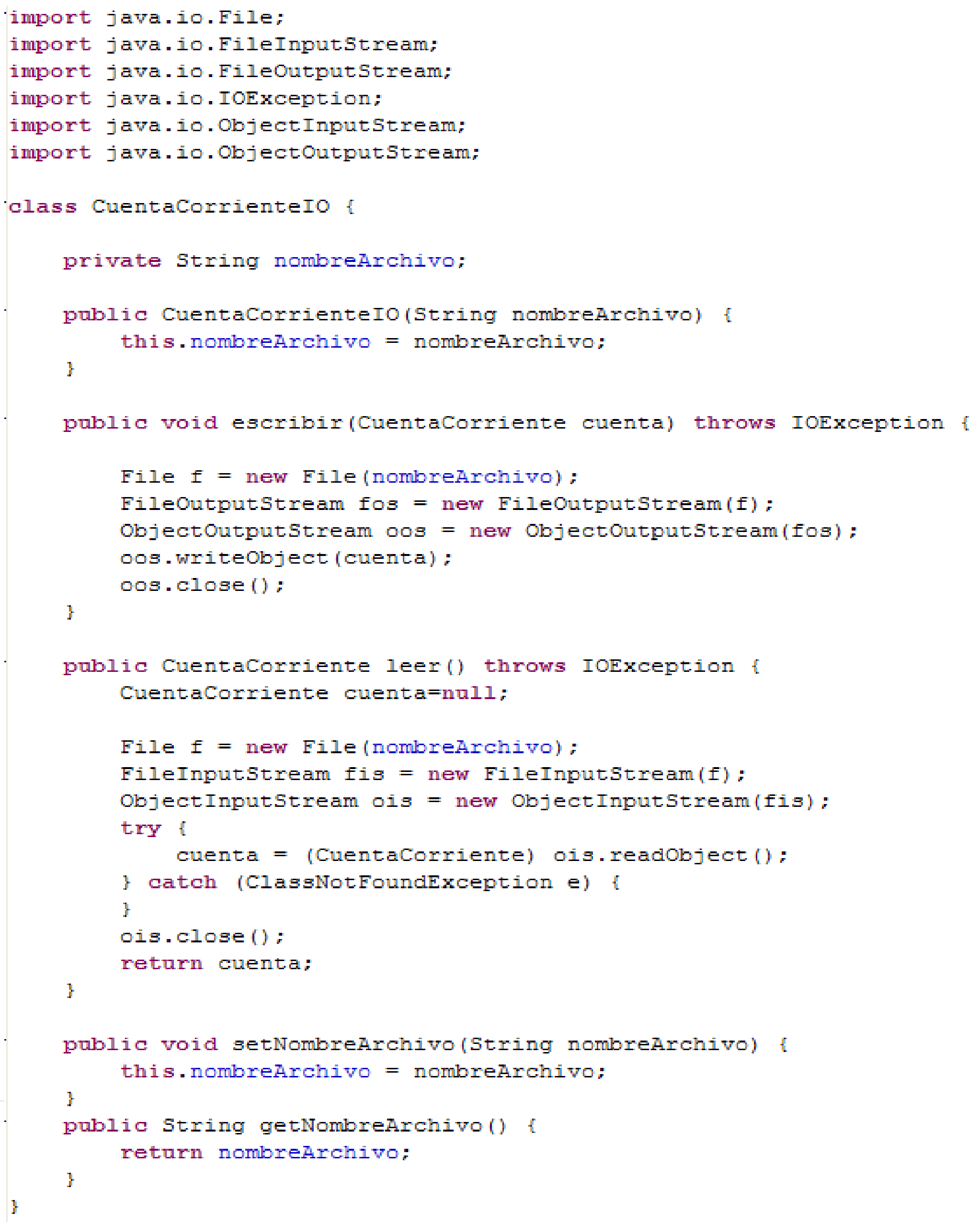
`int`	`readUnsignedShort()` Reads an unsigned 16-bit number from this file.
`String`	`readUTF()` Reads in a string from this file.
`void`	`seek(long pos)` Sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.
`void`	`setLength(long newLength)` Sets the length of this file.
`int`	`skipBytes(int n)` Attempts to skip over `n` bytes of input discarding the skipped bytes.
`void`	`write(byte[] b)` Writes `b.length` bytes from the specified byte array to this file, starting at the current file pointer.
`void`	`write(byte[] b, int off, int len)` Writes `len` bytes from the specified byte array starting at offset `off` to this file.
`void`	`write(int b)` Writes the specified byte to this file.
`void`	`writeBoolean(boolean v)` Writes a `boolean` to the file as a one-byte value.
`void`	`writeByte(int v)` Writes a `byte` to the file as a one-byte value.
`void`	`writeBytes(String s)` Writes the string to the file as a sequence of bytes.
`void`	`writeChar(int v)` Writes a `char` to the file as a two-byte value, high byte first.
`void`	`writeChars(String s)` Writes a string to the file 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 file as an eight-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 file as a four-byte quantity, high byte first.
`void`	`writeInt(int v)` Writes an `int` to the file as four bytes, high byte first.
`void`	`writeLong(long v)` Writes a `long` to the file as eight bytes, high byte first.
`void`	`writeShort(int v)` Writes a `short` to the file as two bytes, high byte first.
`void`	`writeUTF(String str)` Writes a string to the file using UTF-8 encoding in a machine-independent manner.

```
import java.io.Serializable;
import java.util.Date;

public class Movimiento implements Serializable

    private String mConcepto;
    private Date mFecha;
    private double mImporte;

    public Movimiento() {
        mFecha = new Date();
    }

    public double getImporte() {
        return mImporte;
    }

    public String getConcepto() {
        return mConcepto;
    }

    public void setConcepto(String concepto) {
        mConcepto = concepto;
    }

    public Date getFecha() {
        return mFecha;
    }

    public void setFecha(Date fecha) {
        mFecha = fecha;
    }

    public void setImporte(double importe) {
        mImporte = importe;
    }
}
```

```
import java.io.Serializable;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Iterator;

public class CuentaCorriente implements Serializable {

    private String mNumero;
    private String mTitular;
    private ArrayList mMovimientos;

    public CuentaCorriente(String numero, String titular) {
        this.mNumero = numero;
        this.mTitular = titular;
        this.mMovimientos = new ArrayList();
    }

    public void ingresar(String concepto, double x) throws Exception {
        if (x <= 0)
            throw new Exception("No se puede ingresar una cantidad negativa");
        Movimiento m = new Movimiento();
        m.setConcepto(concepto);
        m.setImporte(x);
        this.mMovimientos.add(m);
    }

    public void retirar(String concepto, double x) throws Exception {
        if (x <= 0)
            throw new Exception("No se puede retirar una cantidad negativa");
        if (getSaldo() < x)
            throw new Exception("Saldo insuficiente");
        Movimiento m = new Movimiento();
        m.setConcepto(concepto);
        m.setImporte(-x);
        this.mMovimientos.add(m);
```



```java
    public double getSaldo() {
        double saldo = 0.0;
        for (Iterator iter = mMovimientos.iterator(); iter.hasNext();) {
            Movimiento m = (Movimiento) iter.next();
            saldo += m.getImporte();
        }
        return saldo;
    }

    public void listado() {

        DateFormat formatter = new SimpleDateFormat("dd/MM/yyyy ");
        System.out.println("Titular    \t\tNúmero Cuenta");
        System.out.println("-----------\t\t-------------");
        System.out.println(mTitular +"\t\t"+mNumero);
        System.out.println();

        System.out.println("Fecha\t\t\tDescripcion\t\t\tPrecio");
        System.out.println("-----\t\t\t-----------\t\t\t-----");

        for (Iterator iter = mMovimientos.iterator(); iter.hasNext();) {
            Movimiento m = (Movimiento) iter.next();
            String s = formatter.format(m.getFecha()) + "\t\t"
                    + m.getConcepto() + "\t\t\t" + m.getImporte();
            System.out.println(s);
        }
    }

    public void addMovimiento(Movimiento m) {
        mMovimientos.add(m);
    }
}
```



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

class CuentaCorrienteIO {

    private String nombreArchivo;

    public CuentaCorrienteIO(String nombreArchivo) {
        this.nombreArchivo = nombreArchivo;
    }

    public void escribir(CuentaCorriente cuenta) throws IOException {

        File f = new File(nombreArchivo);
        FileOutputStream fos = new FileOutputStream(f);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(cuenta);
        oos.close();
    }

    public CuentaCorriente leer() throws IOException {
        CuentaCorriente cuenta=null;

        File f = new File(nombreArchivo);
        FileInputStream fis = new FileInputStream(f);
        ObjectInputStream ois = new ObjectInputStream(fis);
        try {
            cuenta = (CuentaCorriente) ois.readObject();
        } catch (ClassNotFoundException e) {
        }
        ois.close();
        return cuenta;
    }

    public void setNombreArchivo(String nombreArchivo) {
        this.nombreArchivo = nombreArchivo;
    }
    public String getNombreArchivo() {
        return nombreArchivo;
    }
}
```

```java
public class PruebaCuentaCorriente {

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

        CuentaCorriente cuenta = new CuentaCorriente ("111-111", "Jose Pérez");
        CuentaCorrienteIO cuentaIO = new CuentaCorrienteIO("c:\\prueba\\cuenta.data");

        Movimiento m1 = new Movimiento();
        m1.setConcepto("concepto 1");
        m1.setImporte(-12.12);
        cuenta.addMovimiento(m1);

        Movimiento m2 = new Movimiento();
        m2.setConcepto("concepto 2");
        m2.setImporte(12.12);
        m2.setFecha(new Date());
        cuenta.addMovimiento(m2);

        System.out.println("Saldo: "+cuenta.getSaldo());
        cuenta.ingresar("ahorrillos", 8.0);
        System.out.println("Saldo: "+cuenta.getSaldo());

        cuentaIO.escribir(cuenta);
        System.out.println("Listado===========");
        cuenta=cuentaIO.leer();
        cuenta.listado();
    }
}
```

```
Saldo: 0.0
Saldo: 8.0
Listado===========
Titular                 Número Cuenta
-----------             -------------
Jose Pérez              111-111

Fecha                   Descripcion                    Precio
-----                   -----------                    -----
16/05/2005              concepto 1                     -12.12
16/05/2005              concepto 2                     12.12
16/05/2005              ahorrillos                     8.0
```