| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ObjectOutputStream.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/io/ObjectOutput.html)   [**NEXT CLASS**](http://docs.google.com/java/io/ObjectOutputStream.PutField.html) | [**FRAMES**](http://docs.google.com/index.html?java/io/ObjectOutputStream.html)    [**NO FRAMES**](http://docs.google.com/ObjectOutputStream.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#2s8eyo1) | [METHOD](#26in1rg) |

## **java.io**

Class ObjectOutputStream

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.io.OutputStream](http://docs.google.com/java/io/OutputStream.html)  
 **java.io.ObjectOutputStream**

**All Implemented Interfaces:** [Closeable](http://docs.google.com/java/io/Closeable.html), [DataOutput](http://docs.google.com/java/io/DataOutput.html), [Flushable](http://docs.google.com/java/io/Flushable.html), [ObjectOutput](http://docs.google.com/java/io/ObjectOutput.html), [ObjectStreamConstants](http://docs.google.com/java/io/ObjectStreamConstants.html)

public class **ObjectOutputStream**extends [OutputStream](http://docs.google.com/java/io/OutputStream.html)implements [ObjectOutput](http://docs.google.com/java/io/ObjectOutput.html), [ObjectStreamConstants](http://docs.google.com/java/io/ObjectStreamConstants.html)

An ObjectOutputStream writes primitive data types and graphs of Java objects to an OutputStream. The objects can be read (reconstituted) using an ObjectInputStream. Persistent storage of objects can be accomplished by using a file for the stream. If the stream is a network socket stream, the objects can be reconstituted on another host or in another process.

Only objects that support the java.io.Serializable interface can be written to streams. The class of each serializable object is encoded including the class name and signature of the class, the values of the object's fields and arrays, and the closure of any other objects referenced from the initial objects.

The method writeObject is used to write an object to the stream. Any object, including Strings and arrays, is written with writeObject. Multiple objects or primitives can be written to the stream. The objects must be read back from the corresponding ObjectInputstream with the same types and in the same order as they were written.

Primitive data types can also be written to the stream using the appropriate methods from DataOutput. Strings can also be written using the writeUTF method.

The default serialization mechanism for an object writes the class of the object, the class signature, and the values of all non-transient and non-static fields. References to other objects (except in transient or static fields) cause those objects to be written also. Multiple references to a single object are encoded using a reference sharing mechanism so that graphs of objects can be restored to the same shape as when the original was written.

For example to write an object that can be read by the example in ObjectInputStream:

FileOutputStream fos = new FileOutputStream("t.tmp");  
 ObjectOutputStream oos = new ObjectOutputStream(fos);  
  
 oos.writeInt(12345);  
 oos.writeObject("Today");  
 oos.writeObject(new Date());  
  
 oos.close();

Classes that require special handling during the serialization and deserialization process must implement special methods with these exact signatures:

private void readObject(java.io.ObjectInputStream stream)  
 throws IOException, ClassNotFoundException;  
 private void writeObject(java.io.ObjectOutputStream stream)  
 throws IOException  
 private void readObjectNoData()   
 throws ObjectStreamException;

The writeObject method is responsible for writing the state of the object for its particular class so that the corresponding readObject method can restore it. The method does not need to concern itself with the state belonging to the object's superclasses or subclasses. State is saved by writing the individual fields to the ObjectOutputStream using the writeObject method or by using the methods for primitive data types supported by DataOutput.

Serialization does not write out the fields of any object that does not implement the java.io.Serializable interface. Subclasses of Objects that are not serializable can be serializable. In this case the non-serializable class must have a no-arg constructor to allow its fields to be initialized. In this case it is the responsibility of the subclass to save and restore the state of the non-serializable class. It is frequently the case that the fields of that class are accessible (public, package, or protected) or that there are get and set methods that can be used to restore the state.

Serialization of an object can be prevented by implementing writeObject and readObject methods that throw the NotSerializableException. The exception will be caught by the ObjectOutputStream and abort the serialization process.

Implementing the Externalizable interface allows the object to assume complete control over the contents and format of the object's serialized form. The methods of the Externalizable interface, writeExternal and readExternal, are called to save and restore the objects state. When implemented by a class they can write and read their own state using all of the methods of ObjectOutput and ObjectInput. It is the responsibility of the objects to handle any versioning that occurs.

Enum constants are serialized differently than ordinary serializable or externalizable objects. The serialized form of an enum constant consists solely of its name; field values of the constant are not transmitted. To serialize an enum constant, ObjectOutputStream writes the string returned by the constant's name method. Like other serializable or externalizable objects, enum constants can function as the targets of back references appearing subsequently in the serialization stream. The process by which enum constants are serialized cannot be customized; any class-specific writeObject and writeReplace methods defined by enum types are ignored during serialization. Similarly, any serialPersistentFields or serialVersionUID field declarations are also ignored--all enum types have a fixed serialVersionUID of 0L.

Primitive data, excluding serializable fields and externalizable data, is written to the ObjectOutputStream in block-data records. A block data record is composed of a header and data. The block data header consists of a marker and the number of bytes to follow the header. Consecutive primitive data writes are merged into one block-data record. The blocking factor used for a block-data record will be 1024 bytes. Each block-data record will be filled up to 1024 bytes, or be written whenever there is a termination of block-data mode. Calls to the ObjectOutputStream methods writeObject, defaultWriteObject and writeFields initially terminate any existing block-data record.

**Since:** JDK1.1 **See Also:**[DataOutput](http://docs.google.com/java/io/DataOutput.html), [ObjectInputStream](http://docs.google.com/java/io/ObjectInputStream.html), [Serializable](http://docs.google.com/java/io/Serializable.html), [Externalizable](http://docs.google.com/java/io/Externalizable.html), [Object Serialization Specification, Section 2, Object Output Classes](http://docs.google.com/platform/serialization/spec/output.html)

| **Nested Class Summary** | |
| --- | --- |
| static class | [**ObjectOutputStream.PutField**](http://docs.google.com/java/io/ObjectOutputStream.PutField.html)            Provide programmatic access to the persistent fields to be written to ObjectOutput. |

| **Field Summary** | |
| --- | --- |

| **Fields inherited from interface java.io.**[**ObjectStreamConstants**](http://docs.google.com/java/io/ObjectStreamConstants.html) |
| --- |
| [baseWireHandle](http://docs.google.com/java/io/ObjectStreamConstants.html#baseWireHandle), [PROTOCOL\_VERSION\_1](http://docs.google.com/java/io/ObjectStreamConstants.html#PROTOCOL_VERSION_1), [PROTOCOL\_VERSION\_2](http://docs.google.com/java/io/ObjectStreamConstants.html#PROTOCOL_VERSION_2), [SC\_BLOCK\_DATA](http://docs.google.com/java/io/ObjectStreamConstants.html#SC_BLOCK_DATA), [SC\_ENUM](http://docs.google.com/java/io/ObjectStreamConstants.html#SC_ENUM), [SC\_EXTERNALIZABLE](http://docs.google.com/java/io/ObjectStreamConstants.html#SC_EXTERNALIZABLE), [SC\_SERIALIZABLE](http://docs.google.com/java/io/ObjectStreamConstants.html#SC_SERIALIZABLE), [SC\_WRITE\_METHOD](http://docs.google.com/java/io/ObjectStreamConstants.html#SC_WRITE_METHOD), [STREAM\_MAGIC](http://docs.google.com/java/io/ObjectStreamConstants.html#STREAM_MAGIC), [STREAM\_VERSION](http://docs.google.com/java/io/ObjectStreamConstants.html#STREAM_VERSION), [SUBCLASS\_IMPLEMENTATION\_PERMISSION](http://docs.google.com/java/io/ObjectStreamConstants.html#SUBCLASS_IMPLEMENTATION_PERMISSION), [SUBSTITUTION\_PERMISSION](http://docs.google.com/java/io/ObjectStreamConstants.html#SUBSTITUTION_PERMISSION), [TC\_ARRAY](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_ARRAY), [TC\_BASE](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_BASE), [TC\_BLOCKDATA](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_BLOCKDATA), [TC\_BLOCKDATALONG](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_BLOCKDATALONG), [TC\_CLASS](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_CLASS), [TC\_CLASSDESC](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_CLASSDESC), [TC\_ENDBLOCKDATA](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_ENDBLOCKDATA), [TC\_ENUM](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_ENUM), [TC\_EXCEPTION](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_EXCEPTION), [TC\_LONGSTRING](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_LONGSTRING), [TC\_MAX](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_MAX), [TC\_NULL](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_NULL), [TC\_OBJECT](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_OBJECT), [TC\_PROXYCLASSDESC](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_PROXYCLASSDESC), [TC\_REFERENCE](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_REFERENCE), [TC\_RESET](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_RESET), [TC\_STRING](http://docs.google.com/java/io/ObjectStreamConstants.html#TC_STRING) |

| **Constructor Summary** | |
| --- | --- |
| protected | [**ObjectOutputStream**](http://docs.google.com/java/io/ObjectOutputStream.html#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**](http://docs.google.com/java/io/ObjectOutputStream.html#ObjectOutputStream(java.io.OutputStream))([OutputStream](http://docs.google.com/java/io/OutputStream.html) out)            Creates an ObjectOutputStream that writes to the specified OutputStream. |

| **Method Summary** | |
| --- | --- |
| protected  void | [**annotateClass**](http://docs.google.com/java/io/ObjectOutputStream.html#annotateClass(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<?> cl)            Subclasses may implement this method to allow class data to be stored in the stream. |
| protected  void | [**annotateProxyClass**](http://docs.google.com/java/io/ObjectOutputStream.html#annotateProxyClass(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<?> cl)            Subclasses may implement this method to store custom data in the stream along with descriptors for dynamic proxy classes. |
| void | [**close**](http://docs.google.com/java/io/ObjectOutputStream.html#close())()            Closes the stream. |
| void | [**defaultWriteObject**](http://docs.google.com/java/io/ObjectOutputStream.html#defaultWriteObject())()            Write the non-static and non-transient fields of the current class to this stream. |
| protected  void | [**drain**](http://docs.google.com/java/io/ObjectOutputStream.html#drain())()            Drain any buffered data in ObjectOutputStream. |
| protected  boolean | [**enableReplaceObject**](http://docs.google.com/java/io/ObjectOutputStream.html#enableReplaceObject(boolean))(boolean enable)            Enable the stream to do replacement of objects in the stream. |
| void | [**flush**](http://docs.google.com/java/io/ObjectOutputStream.html#flush())()            Flushes the stream. |
| [ObjectOutputStream.PutField](http://docs.google.com/java/io/ObjectOutputStream.PutField.html) | [**putFields**](http://docs.google.com/java/io/ObjectOutputStream.html#putFields())()            Retrieve the object used to buffer persistent fields to be written to the stream. |
| protected  [Object](http://docs.google.com/java/lang/Object.html) | [**replaceObject**](http://docs.google.com/java/io/ObjectOutputStream.html#replaceObject(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            This method will allow trusted subclasses of ObjectOutputStream to substitute one object for another during serialization. |
| void | [**reset**](http://docs.google.com/java/io/ObjectOutputStream.html#reset())()            Reset will disregard the state of any objects already written to the stream. |
| void | [**useProtocolVersion**](http://docs.google.com/java/io/ObjectOutputStream.html#useProtocolVersion(int))(int version)            Specify stream protocol version to use when writing the stream. |
| void | [**write**](http://docs.google.com/java/io/ObjectOutputStream.html#write(byte%5B%5D))(byte[] buf)            Writes an array of bytes. |
| void | [**write**](http://docs.google.com/java/io/ObjectOutputStream.html#write(byte%5B%5D,%20int,%20int))(byte[] buf, int off, int len)            Writes a sub array of bytes. |
| void | [**write**](http://docs.google.com/java/io/ObjectOutputStream.html#write(int))(int val)            Writes a byte. |
| void | [**writeBoolean**](http://docs.google.com/java/io/ObjectOutputStream.html#writeBoolean(boolean))(boolean val)            Writes a boolean. |
| void | [**writeByte**](http://docs.google.com/java/io/ObjectOutputStream.html#writeByte(int))(int val)            Writes an 8 bit byte. |
| void | [**writeBytes**](http://docs.google.com/java/io/ObjectOutputStream.html#writeBytes(java.lang.String))([String](http://docs.google.com/java/lang/String.html) str)            Writes a String as a sequence of bytes. |
| void | [**writeChar**](http://docs.google.com/java/io/ObjectOutputStream.html#writeChar(int))(int val)            Writes a 16 bit char. |
| void | [**writeChars**](http://docs.google.com/java/io/ObjectOutputStream.html#writeChars(java.lang.String))([String](http://docs.google.com/java/lang/String.html) str)            Writes a String as a sequence of chars. |
| protected  void | [**writeClassDescriptor**](http://docs.google.com/java/io/ObjectOutputStream.html#writeClassDescriptor(java.io.ObjectStreamClass))([ObjectStreamClass](http://docs.google.com/java/io/ObjectStreamClass.html) desc)            Write the specified class descriptor to the ObjectOutputStream. |
| void | [**writeDouble**](http://docs.google.com/java/io/ObjectOutputStream.html#writeDouble(double))(double val)            Writes a 64 bit double. |
| void | [**writeFields**](http://docs.google.com/java/io/ObjectOutputStream.html#writeFields())()            Write the buffered fields to the stream. |
| void | [**writeFloat**](http://docs.google.com/java/io/ObjectOutputStream.html#writeFloat(float))(float val)            Writes a 32 bit float. |
| void | [**writeInt**](http://docs.google.com/java/io/ObjectOutputStream.html#writeInt(int))(int val)            Writes a 32 bit int. |
| void | [**writeLong**](http://docs.google.com/java/io/ObjectOutputStream.html#writeLong(long))(long val)            Writes a 64 bit long. |
| void | [**writeObject**](http://docs.google.com/java/io/ObjectOutputStream.html#writeObject(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Write the specified object to the ObjectOutputStream. |
| protected  void | [**writeObjectOverride**](http://docs.google.com/java/io/ObjectOutputStream.html#writeObjectOverride(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Method used by subclasses to override the default writeObject method. |
| void | [**writeShort**](http://docs.google.com/java/io/ObjectOutputStream.html#writeShort(int))(int val)            Writes a 16 bit short. |
| protected  void | [**writeStreamHeader**](http://docs.google.com/java/io/ObjectOutputStream.html#writeStreamHeader())()            The writeStreamHeader method is provided so subclasses can append or prepend their own header to the stream. |
| void | [**writeUnshared**](http://docs.google.com/java/io/ObjectOutputStream.html#writeUnshared(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Writes an "unshared" object to the ObjectOutputStream. |
| void | [**writeUTF**](http://docs.google.com/java/io/ObjectOutputStream.html#writeUTF(java.lang.String))([String](http://docs.google.com/java/lang/String.html) str)            Primitive data write of this String in [modified UTF-8](http://docs.google.com/DataInput.html#modified-utf-8) format. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Constructor Detail** |
| --- |

### ObjectOutputStream

public **ObjectOutputStream**([OutputStream](http://docs.google.com/java/io/OutputStream.html) out)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Creates an ObjectOutputStream that writes to the specified OutputStream. This constructor writes the serialization stream header to the underlying stream; callers may wish to flush the stream immediately to ensure that constructors for receiving ObjectInputStreams will not block when reading the header.

If a security manager is installed, this constructor will check for the "enableSubclassImplementation" SerializablePermission when invoked directly or indirectly by the constructor of a subclass which overrides the ObjectOutputStream.putFields or ObjectOutputStream.writeUnshared methods.

**Parameters:**out - output stream to write to **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs while writing stream header [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if untrusted subclass illegally overrides security-sensitive methods [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if out is null**Since:** 1.4 **See Also:**[ObjectOutputStream()](http://docs.google.com/java/io/ObjectOutputStream.html#ObjectOutputStream()), [putFields()](http://docs.google.com/java/io/ObjectOutputStream.html#putFields()), [ObjectInputStream.ObjectInputStream(InputStream)](http://docs.google.com/java/io/ObjectInputStream.html#ObjectInputStream(java.io.InputStream))

### ObjectOutputStream

protected **ObjectOutputStream**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Provide a way for subclasses that are completely reimplementing ObjectOutputStream to not have to allocate private data just used by this implementation of ObjectOutputStream.

If there is a security manager installed, this method first calls the security manager's checkPermission method with a SerializablePermission("enableSubclassImplementation") permission to ensure it's ok to enable subclassing.

**Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkPermission method denies enabling subclassing. [IOException](http://docs.google.com/java/io/IOException.html)**See Also:**[SecurityManager.checkPermission(java.security.Permission)](http://docs.google.com/java/lang/SecurityManager.html#checkPermission(java.security.Permission)), [SerializablePermission](http://docs.google.com/java/io/SerializablePermission.html)

| **Method Detail** |
| --- |

### useProtocolVersion

public void **useProtocolVersion**(int version)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Specify stream protocol version to use when writing the stream.

This routine provides a hook to enable the current version of Serialization to write in a format that is backwards compatible to a previous version of the stream format.

Every effort will be made to avoid introducing additional backwards incompatibilities; however, sometimes there is no other alternative.

**Parameters:**version - use ProtocolVersion from java.io.ObjectStreamConstants. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if called after any objects have been serialized. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if invalid version is passed in. [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur**Since:** 1.2 **See Also:**[ObjectStreamConstants.PROTOCOL\_VERSION\_1](http://docs.google.com/java/io/ObjectStreamConstants.html#PROTOCOL_VERSION_1), [ObjectStreamConstants.PROTOCOL\_VERSION\_2](http://docs.google.com/java/io/ObjectStreamConstants.html#PROTOCOL_VERSION_2)

### writeObject

public final void **writeObject**([Object](http://docs.google.com/java/lang/Object.html) obj)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Write the specified object to the ObjectOutputStream. The class of the object, the signature of the class, and the values of the non-transient and non-static fields of the class and all of its supertypes are written. Default serialization for a class can be overridden using the writeObject and the readObject methods. Objects referenced by this object are written transitively so that a complete equivalent graph of objects can be reconstructed by an ObjectInputStream.

Exceptions are thrown for problems with the OutputStream and for classes that should not be serialized. All exceptions are fatal to the OutputStream, which is left in an indeterminate state, and it is up to the caller to ignore or recover the stream state.

**Specified by:**[writeObject](http://docs.google.com/java/io/ObjectOutput.html#writeObject(java.lang.Object)) in interface [ObjectOutput](http://docs.google.com/java/io/ObjectOutput.html) **Parameters:**obj - the object to be written **Throws:** [InvalidClassException](http://docs.google.com/java/io/InvalidClassException.html) - Something is wrong with a class used by serialization. [NotSerializableException](http://docs.google.com/java/io/NotSerializableException.html) - Some object to be serialized does not implement the java.io.Serializable interface. [IOException](http://docs.google.com/java/io/IOException.html) - Any exception thrown by the underlying OutputStream.

### writeObjectOverride

protected void **writeObjectOverride**([Object](http://docs.google.com/java/lang/Object.html) obj)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Method used by subclasses to override the default writeObject method. This method is called by trusted subclasses of ObjectInputStream that constructed ObjectInputStream using the protected no-arg constructor. The subclass is expected to provide an override method with the modifier "final".

**Parameters:**obj - object to be written to the underlying stream **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if there are I/O errors while writing to the underlying stream**Since:** 1.2 **See Also:**[ObjectOutputStream()](http://docs.google.com/java/io/ObjectOutputStream.html#ObjectOutputStream()), [writeObject(Object)](http://docs.google.com/java/io/ObjectOutputStream.html#writeObject(java.lang.Object))

### writeUnshared

public void **writeUnshared**([Object](http://docs.google.com/java/lang/Object.html) obj)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes an "unshared" object to the ObjectOutputStream. This method is identical to writeObject, except that it always writes the given object as a new, unique object in the stream (as opposed to a back-reference pointing to a previously serialized instance). Specifically:

* An object written via writeUnshared is always serialized in the same manner as a newly appearing object (an object that has not been written to the stream yet), regardless of whether or not the object has been written previously.
* If writeObject is used to write an object that has been previously written with writeUnshared, the previous writeUnshared operation is treated as if it were a write of a separate object. In other words, ObjectOutputStream will never generate back-references to object data written by calls to writeUnshared.

While writing an object via writeUnshared does not in itself guarantee a unique reference to the object when it is deserialized, it allows a single object to be defined multiple times in a stream, so that multiple calls to readUnshared by the receiver will not conflict. Note that the rules described above only apply to the base-level object written with writeUnshared, and not to any transitively referenced sub-objects in the object graph to be serialized.

ObjectOutputStream subclasses which override this method can only be constructed in security contexts possessing the "enableSubclassImplementation" SerializablePermission; any attempt to instantiate such a subclass without this permission will cause a SecurityException to be thrown.

**Parameters:**obj - object to write to stream **Throws:** [NotSerializableException](http://docs.google.com/java/io/NotSerializableException.html) - if an object in the graph to be serialized does not implement the Serializable interface [InvalidClassException](http://docs.google.com/java/io/InvalidClassException.html) - if a problem exists with the class of an object to be serialized [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs during serialization**Since:** 1.4

### defaultWriteObject

public void **defaultWriteObject**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Write the non-static and non-transient fields of the current class to this stream. This may only be called from the writeObject method of the class being serialized. It will throw the NotActiveException if it is called otherwise.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying OutputStream

### putFields

public [ObjectOutputStream.PutField](http://docs.google.com/java/io/ObjectOutputStream.PutField.html) **putFields**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Retrieve the object used to buffer persistent fields to be written to the stream. The fields will be written to the stream when writeFields method is called.

**Returns:**an instance of the class Putfield that holds the serializable fields **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur**Since:** 1.2

### writeFields

public void **writeFields**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Write the buffered fields to the stream.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream [NotActiveException](http://docs.google.com/java/io/NotActiveException.html) - Called when a classes writeObject method was not called to write the state of the object.**Since:** 1.2

### reset

public void **reset**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Reset will disregard the state of any objects already written to the stream. The state is reset to be the same as a new ObjectOutputStream. The current point in the stream is marked as reset so the corresponding ObjectInputStream will be reset at the same point. Objects previously written to the stream will not be refered to as already being in the stream. They will be written to the stream again.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if reset() is invoked while serializing an object.

### annotateClass

protected void **annotateClass**([Class](http://docs.google.com/java/lang/Class.html)<?> cl)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Subclasses may implement this method to allow class data to be stored in the stream. By default this method does nothing. The corresponding method in ObjectInputStream is resolveClass. This method is called exactly once for each unique class in the stream. The class name and signature will have already been written to the stream. This method may make free use of the ObjectOutputStream to save any representation of the class it deems suitable (for example, the bytes of the class file). The resolveClass method in the corresponding subclass of ObjectInputStream must read and use any data or objects written by annotateClass.

**Parameters:**cl - the class to annotate custom data for **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - Any exception thrown by the underlying OutputStream.

### annotateProxyClass

protected void **annotateProxyClass**([Class](http://docs.google.com/java/lang/Class.html)<?> cl)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Subclasses may implement this method to store custom data in the stream along with descriptors for dynamic proxy classes.

This method is called exactly once for each unique proxy class descriptor in the stream. The default implementation of this method in ObjectOutputStream does nothing.

The corresponding method in ObjectInputStream is resolveProxyClass. For a given subclass of ObjectOutputStream that overrides this method, the resolveProxyClass method in the corresponding subclass of ObjectInputStream must read any data or objects written by annotateProxyClass.

**Parameters:**cl - the proxy class to annotate custom data for **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - any exception thrown by the underlying OutputStream**Since:** 1.3 **See Also:**[ObjectInputStream.resolveProxyClass(String[])](http://docs.google.com/java/io/ObjectInputStream.html#resolveProxyClass(java.lang.String%5B%5D))

### replaceObject

protected [Object](http://docs.google.com/java/lang/Object.html) **replaceObject**([Object](http://docs.google.com/java/lang/Object.html) obj)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

This method will allow trusted subclasses of ObjectOutputStream to substitute one object for another during serialization. Replacing objects is disabled until enableReplaceObject is called. The enableReplaceObject method checks that the stream requesting to do replacement can be trusted. The first occurrence of each object written into the serialization stream is passed to replaceObject. Subsequent references to the object are replaced by the object returned by the original call to replaceObject. To ensure that the private state of objects is not unintentionally exposed, only trusted streams may use replaceObject.

The ObjectOutputStream.writeObject method takes a parameter of type Object (as opposed to type Serializable) to allow for cases where non-serializable objects are replaced by serializable ones.

When a subclass is replacing objects it must insure that either a complementary substitution must be made during deserialization or that the substituted object is compatible with every field where the reference will be stored. Objects whose type is not a subclass of the type of the field or array element abort the serialization by raising an exception and the object is not be stored.

This method is called only once when each object is first encountered. All subsequent references to the object will be redirected to the new object. This method should return the object to be substituted or the original object.

Null can be returned as the object to be substituted, but may cause NullReferenceException in classes that contain references to the original object since they may be expecting an object instead of null.

**Parameters:**obj - the object to be replaced **Returns:**the alternate object that replaced the specified one **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - Any exception thrown by the underlying OutputStream.

### enableReplaceObject

protected boolean **enableReplaceObject**(boolean enable)  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Enable the stream to do replacement of objects in the stream. When enabled, the replaceObject method is called for every object being serialized.

If enable is true, and there is a security manager installed, this method first calls the security manager's checkPermission method with a SerializablePermission("enableSubstitution") permission to ensure it's ok to enable the stream to do replacement of objects in the stream.

**Parameters:**enable - boolean parameter to enable replacement of objects **Returns:**the previous setting before this method was invoked **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its checkPermission method denies enabling the stream to do replacement of objects in the stream.**See Also:**[SecurityManager.checkPermission(java.security.Permission)](http://docs.google.com/java/lang/SecurityManager.html#checkPermission(java.security.Permission)), [SerializablePermission](http://docs.google.com/java/io/SerializablePermission.html)

### writeStreamHeader

protected void **writeStreamHeader**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

The writeStreamHeader method is provided so subclasses can append or prepend their own header to the stream. It writes the magic number and version to the stream.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

### writeClassDescriptor

protected void **writeClassDescriptor**([ObjectStreamClass](http://docs.google.com/java/io/ObjectStreamClass.html) desc)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Write the specified class descriptor to the ObjectOutputStream. Class descriptors are used to identify the classes of objects written to the stream. Subclasses of ObjectOutputStream may override this method to customize the way in which class descriptors are written to the serialization stream. The corresponding method in ObjectInputStream, readClassDescriptor, should then be overridden to reconstitute the class descriptor from its custom stream representation. By default, this method writes class descriptors according to the format defined in the Object Serialization specification.

Note that this method will only be called if the ObjectOutputStream is not using the old serialization stream format (set by calling ObjectOutputStream's useProtocolVersion method). If this serialization stream is using the old format (PROTOCOL\_VERSION\_1), the class descriptor will be written internally in a manner that cannot be overridden or customized.

**Parameters:**desc - class descriptor to write to the stream **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error has occurred.**Since:** 1.3 **See Also:**[ObjectInputStream.readClassDescriptor()](http://docs.google.com/java/io/ObjectInputStream.html#readClassDescriptor()), [useProtocolVersion(int)](http://docs.google.com/java/io/ObjectOutputStream.html#useProtocolVersion(int)), [ObjectStreamConstants.PROTOCOL\_VERSION\_1](http://docs.google.com/java/io/ObjectStreamConstants.html#PROTOCOL_VERSION_1)

### write

public void **write**(int val)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a byte. This method will block until the byte is actually written.

**Specified by:**[write](http://docs.google.com/java/io/DataOutput.html#write(int)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html)**Specified by:**[write](http://docs.google.com/java/io/ObjectOutput.html#write(int)) in interface [ObjectOutput](http://docs.google.com/java/io/ObjectOutput.html)**Specified by:**[write](http://docs.google.com/java/io/OutputStream.html#write(int)) in class [OutputStream](http://docs.google.com/java/io/OutputStream.html) **Parameters:**val - the byte to be written to the stream **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error has occurred.

### write

public void **write**(byte[] buf)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes an array of bytes. This method will block until the bytes are actually written.

**Specified by:**[write](http://docs.google.com/java/io/DataOutput.html#write(byte%5B%5D)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html)**Specified by:**[write](http://docs.google.com/java/io/ObjectOutput.html#write(byte%5B%5D)) in interface [ObjectOutput](http://docs.google.com/java/io/ObjectOutput.html)**Overrides:**[write](http://docs.google.com/java/io/OutputStream.html#write(byte%5B%5D)) in class [OutputStream](http://docs.google.com/java/io/OutputStream.html) **Parameters:**buf - the data to be written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error has occurred.**See Also:**[OutputStream.write(byte[], int, int)](http://docs.google.com/java/io/OutputStream.html#write(byte%5B%5D,%20int,%20int))

### write

public void **write**(byte[] buf,  
 int off,  
 int len)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a sub array of bytes.

**Specified by:**[write](http://docs.google.com/java/io/DataOutput.html#write(byte%5B%5D,%20int,%20int)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html)**Specified by:**[write](http://docs.google.com/java/io/ObjectOutput.html#write(byte%5B%5D,%20int,%20int)) in interface [ObjectOutput](http://docs.google.com/java/io/ObjectOutput.html)**Overrides:**[write](http://docs.google.com/java/io/OutputStream.html#write(byte%5B%5D,%20int,%20int)) in class [OutputStream](http://docs.google.com/java/io/OutputStream.html) **Parameters:**buf - the data to be writtenoff - the start offset in the datalen - the number of bytes that are written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error has occurred.

### flush

public void **flush**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Flushes the stream. This will write any buffered output bytes and flush through to the underlying stream.

**Specified by:**[flush](http://docs.google.com/java/io/Flushable.html#flush()) in interface [Flushable](http://docs.google.com/java/io/Flushable.html)**Specified by:**[flush](http://docs.google.com/java/io/ObjectOutput.html#flush()) in interface [ObjectOutput](http://docs.google.com/java/io/ObjectOutput.html)**Overrides:**[flush](http://docs.google.com/java/io/OutputStream.html#flush()) in class [OutputStream](http://docs.google.com/java/io/OutputStream.html) **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error has occurred.

### drain

protected void **drain**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Drain any buffered data in ObjectOutputStream. Similar to flush but does not propagate the flush to the underlying stream.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

### close

public void **close**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Closes the stream. This method must be called to release any resources associated with the stream.

**Specified by:**[close](http://docs.google.com/java/io/Closeable.html#close()) in interface [Closeable](http://docs.google.com/java/io/Closeable.html)**Specified by:**[close](http://docs.google.com/java/io/ObjectOutput.html#close()) in interface [ObjectOutput](http://docs.google.com/java/io/ObjectOutput.html)**Overrides:**[close](http://docs.google.com/java/io/OutputStream.html#close()) in class [OutputStream](http://docs.google.com/java/io/OutputStream.html) **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error has occurred.

### writeBoolean

public void **writeBoolean**(boolean val)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a boolean.

**Specified by:**[writeBoolean](http://docs.google.com/java/io/DataOutput.html#writeBoolean(boolean)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**val - the boolean to be written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

### writeByte

public void **writeByte**(int val)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes an 8 bit byte.

**Specified by:**[writeByte](http://docs.google.com/java/io/DataOutput.html#writeByte(int)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**val - the byte value to be written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

### writeShort

public void **writeShort**(int val)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a 16 bit short.

**Specified by:**[writeShort](http://docs.google.com/java/io/DataOutput.html#writeShort(int)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**val - the short value to be written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

### writeChar

public void **writeChar**(int val)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a 16 bit char.

**Specified by:**[writeChar](http://docs.google.com/java/io/DataOutput.html#writeChar(int)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**val - the char value to be written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

### writeInt

public void **writeInt**(int val)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a 32 bit int.

**Specified by:**[writeInt](http://docs.google.com/java/io/DataOutput.html#writeInt(int)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**val - the integer value to be written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

### writeLong

public void **writeLong**(long val)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a 64 bit long.

**Specified by:**[writeLong](http://docs.google.com/java/io/DataOutput.html#writeLong(long)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**val - the long value to be written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

### writeFloat

public void **writeFloat**(float val)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a 32 bit float.

**Specified by:**[writeFloat](http://docs.google.com/java/io/DataOutput.html#writeFloat(float)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**val - the float value to be written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

### writeDouble

public void **writeDouble**(double val)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a 64 bit double.

**Specified by:**[writeDouble](http://docs.google.com/java/io/DataOutput.html#writeDouble(double)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**val - the double value to be written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

### writeBytes

public void **writeBytes**([String](http://docs.google.com/java/lang/String.html) str)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a String as a sequence of bytes.

**Specified by:**[writeBytes](http://docs.google.com/java/io/DataOutput.html#writeBytes(java.lang.String)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**str - the String of bytes to be written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

### writeChars

public void **writeChars**([String](http://docs.google.com/java/lang/String.html) str)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes a String as a sequence of chars.

**Specified by:**[writeChars](http://docs.google.com/java/io/DataOutput.html#writeChars(java.lang.String)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**str - the String of chars to be written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

### writeUTF

public void **writeUTF**([String](http://docs.google.com/java/lang/String.html) str)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Primitive data write of this String in [modified UTF-8](http://docs.google.com/DataInput.html#modified-utf-8) format. Note that there is a significant difference between writing a String into the stream as primitive data or as an Object. A String instance written by writeObject is written into the stream as a String initially. Future writeObject() calls write references to the string into the stream.

**Specified by:**[writeUTF](http://docs.google.com/java/io/DataOutput.html#writeUTF(java.lang.String)) in interface [DataOutput](http://docs.google.com/java/io/DataOutput.html) **Parameters:**str - the String to be written **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occur while writing to the underlying stream

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ObjectOutputStream.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/io/ObjectOutput.html)   [**NEXT CLASS**](http://docs.google.com/java/io/ObjectOutputStream.PutField.html) | [**FRAMES**](http://docs.google.com/index.html?java/io/ObjectOutputStream.html)    [**NO FRAMES**](http://docs.google.com/ObjectOutputStream.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#2s8eyo1) | [METHOD](#26in1rg) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).