| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ObjectInputStream.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/ObjectInput.html)   [**NEXT CLASS**](http://docs.google.com/java/io/ObjectInputStream.GetField.html) | [**FRAMES**](http://docs.google.com/index.html?java/io/ObjectInputStream.html)    [**NO FRAMES**](http://docs.google.com/ObjectInputStream.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#35nkun2) |

## **java.io**

Class ObjectInputStream

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

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

public class **ObjectInputStream**extends [InputStream](http://docs.google.com/java/io/InputStream.html)implements [ObjectInput](http://docs.google.com/java/io/ObjectInput.html), [ObjectStreamConstants](http://docs.google.com/java/io/ObjectStreamConstants.html)

An ObjectInputStream deserializes primitive data and objects previously written using an ObjectOutputStream.

ObjectOutputStream and ObjectInputStream can provide an application with persistent storage for graphs of objects when used with a FileOutputStream and FileInputStream respectively. ObjectInputStream is used to recover those objects previously serialized. Other uses include passing objects between hosts using a socket stream or for marshaling and unmarshaling arguments and parameters in a remote communication system.

ObjectInputStream ensures that the types of all objects in the graph created from the stream match the classes present in the Java Virtual Machine. Classes are loaded as required using the standard mechanisms.

Only objects that support the java.io.Serializable or java.io.Externalizable interface can be read from streams.

The method readObject is used to read an object from the stream. Java's safe casting should be used to get the desired type. In Java, strings and arrays are objects and are treated as objects during serialization. When read they need to be cast to the expected type.

Primitive data types can be read from the stream using the appropriate method on DataInput.

The default deserialization mechanism for objects restores the contents of each field to the value and type it had when it was written. Fields declared as transient or static are ignored by the deserialization process. References to other objects cause those objects to be read from the stream as necessary. Graphs of objects are restored correctly using a reference sharing mechanism. New objects are always allocated when deserializing, which prevents existing objects from being overwritten.

Reading an object is analogous to running the constructors of a new object. Memory is allocated for the object and initialized to zero (NULL). No-arg constructors are invoked for the non-serializable classes and then the fields of the serializable classes are restored from the stream starting with the serializable class closest to java.lang.object and finishing with the object's most specific class.

For example to read from a stream as written by the example in ObjectOutputStream:

FileInputStream fis = new FileInputStream("t.tmp");  
 ObjectInputStream ois = new ObjectInputStream(fis);  
  
 int i = ois.readInt();  
 String today = (String) ois.readObject();  
 Date date = (Date) ois.readObject();  
  
 ois.close();

Classes control how they are serialized by implementing either the java.io.Serializable or java.io.Externalizable interfaces.

Implementing the Serializable interface allows object serialization to save and restore the entire state of the object and it allows classes to evolve between the time the stream is written and the time it is read. It automatically traverses references between objects, saving and restoring entire graphs.

Serializable classes that require special handling during the serialization and deserialization process should implement the following methods:

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

The readObject method is responsible for reading and restoring the state of the object for its particular class using data written to the stream by the corresponding writeObject method. The method does not need to concern itself with the state belonging to its superclasses or subclasses. State is restored by reading data from the ObjectInputStream for the individual fields and making assignments to the appropriate fields of the object. Reading primitive data types is supported by DataInput.

Any attempt to read object data which exceeds the boundaries of the custom data written by the corresponding writeObject method will cause an OptionalDataException to be thrown with an eof field value of true. Non-object reads which exceed the end of the allotted data will reflect the end of data in the same way that they would indicate the end of the stream: bytewise reads will return -1 as the byte read or number of bytes read, and primitive reads will throw EOFExceptions. If there is no corresponding writeObject method, then the end of default serialized data marks the end of the allotted data.

Primitive and object read calls issued from within a readExternal method behave in the same manner--if the stream is already positioned at the end of data written by the corresponding writeExternal method, object reads will throw OptionalDataExceptions with eof set to true, bytewise reads will return -1, and primitive reads will throw EOFExceptions. Note that this behavior does not hold for streams written with the old ObjectStreamConstants.PROTOCOL\_VERSION\_1 protocol, in which the end of data written by writeExternal methods is not demarcated, and hence cannot be detected.

The readObjectNoData method is responsible for initializing the state of the object for its particular class in the event that the serialization stream does not list the given class as a superclass of the object being deserialized. This may occur in cases where the receiving party uses a different version of the deserialized instance's class than the sending party, and the receiver's version extends classes that are not extended by the sender's version. This may also occur if the serialization stream has been tampered; hence, readObjectNoData is useful for initializing deserialized objects properly despite a "hostile" or incomplete source stream.

Serialization does not read or assign values to 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.

Any exception that occurs while deserializing an object will be caught by the ObjectInputStream and abort the reading 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 deserialized 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 deserialize an enum constant, ObjectInputStream reads the constant name from the stream; the deserialized constant is then obtained by calling the static method Enum.valueOf(Class, String) with the enum constant's base type and the received constant name as arguments. 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 deserialized cannot be customized: any class-specific readObject, readObjectNoData, and readResolve methods defined by enum types are ignored during deserialization. Similarly, any serialPersistentFields or serialVersionUID field declarations are also ignored--all enum types have a fixed serialVersionUID of 0L.

**Since:** JDK1.1 **See Also:**[DataInput](http://docs.google.com/java/io/DataInput.html), [ObjectOutputStream](http://docs.google.com/java/io/ObjectOutputStream.html), [Serializable](http://docs.google.com/java/io/Serializable.html),  [Object Serialization Specification, Section 3, Object Input Classes](http://docs.google.com/platform/serialization/spec/input.html)

| **Nested Class Summary** | |
| --- | --- |
| static class | [**ObjectInputStream.GetField**](http://docs.google.com/java/io/ObjectInputStream.GetField.html)            Provide access to the persistent fields read from the input stream. |

| **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 | [**ObjectInputStream**](http://docs.google.com/java/io/ObjectInputStream.html#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**](http://docs.google.com/java/io/ObjectInputStream.html#ObjectInputStream(java.io.InputStream))([InputStream](http://docs.google.com/java/io/InputStream.html) in)            Creates an ObjectInputStream that reads from the specified InputStream. |

| **Method Summary** | |
| --- | --- |
| int | [**available**](http://docs.google.com/java/io/ObjectInputStream.html#available())()            Returns the number of bytes that can be read without blocking. |
| void | [**close**](http://docs.google.com/java/io/ObjectInputStream.html#close())()            Closes the input stream. |
| void | [**defaultReadObject**](http://docs.google.com/java/io/ObjectInputStream.html#defaultReadObject())()            Read the non-static and non-transient fields of the current class from this stream. |
| protected  boolean | [**enableResolveObject**](http://docs.google.com/java/io/ObjectInputStream.html#enableResolveObject(boolean))(boolean enable)            Enable the stream to allow objects read from the stream to be replaced. |
| int | [**read**](http://docs.google.com/java/io/ObjectInputStream.html#read())()            Reads a byte of data. |
| int | [**read**](http://docs.google.com/java/io/ObjectInputStream.html#read(byte%5B%5D,%20int,%20int))(byte[] buf, int off, int len)            Reads into an array of bytes. |
| boolean | [**readBoolean**](http://docs.google.com/java/io/ObjectInputStream.html#readBoolean())()            Reads in a boolean. |
| byte | [**readByte**](http://docs.google.com/java/io/ObjectInputStream.html#readByte())()            Reads an 8 bit byte. |
| char | [**readChar**](http://docs.google.com/java/io/ObjectInputStream.html#readChar())()            Reads a 16 bit char. |
| protected  [ObjectStreamClass](http://docs.google.com/java/io/ObjectStreamClass.html) | [**readClassDescriptor**](http://docs.google.com/java/io/ObjectInputStream.html#readClassDescriptor())()            Read a class descriptor from the serialization stream. |
| double | [**readDouble**](http://docs.google.com/java/io/ObjectInputStream.html#readDouble())()            Reads a 64 bit double. |
| [ObjectInputStream.GetField](http://docs.google.com/java/io/ObjectInputStream.GetField.html) | [**readFields**](http://docs.google.com/java/io/ObjectInputStream.html#readFields())()            Reads the persistent fields from the stream and makes them available by name. |
| float | [**readFloat**](http://docs.google.com/java/io/ObjectInputStream.html#readFloat())()            Reads a 32 bit float. |
| void | [**readFully**](http://docs.google.com/java/io/ObjectInputStream.html#readFully(byte%5B%5D))(byte[] buf)            Reads bytes, blocking until all bytes are read. |
| void | [**readFully**](http://docs.google.com/java/io/ObjectInputStream.html#readFully(byte%5B%5D,%20int,%20int))(byte[] buf, int off, int len)            Reads bytes, blocking until all bytes are read. |
| int | [**readInt**](http://docs.google.com/java/io/ObjectInputStream.html#readInt())()            Reads a 32 bit int. |
| [String](http://docs.google.com/java/lang/String.html) | [**readLine**](http://docs.google.com/java/io/ObjectInputStream.html#readLine())()  **Deprecated.** *This method does not properly convert bytes to characters. see DataInputStream for the details and alternatives.* |
| long | [**readLong**](http://docs.google.com/java/io/ObjectInputStream.html#readLong())()            Reads a 64 bit long. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**readObject**](http://docs.google.com/java/io/ObjectInputStream.html#readObject())()            Read an object from the ObjectInputStream. |
| protected  [Object](http://docs.google.com/java/lang/Object.html) | [**readObjectOverride**](http://docs.google.com/java/io/ObjectInputStream.html#readObjectOverride())()            This method is called by trusted subclasses of ObjectOutputStream that constructed ObjectOutputStream using the protected no-arg constructor. |
| short | [**readShort**](http://docs.google.com/java/io/ObjectInputStream.html#readShort())()            Reads a 16 bit short. |
| protected  void | [**readStreamHeader**](http://docs.google.com/java/io/ObjectInputStream.html#readStreamHeader())()            The readStreamHeader method is provided to allow subclasses to read and verify their own stream headers. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**readUnshared**](http://docs.google.com/java/io/ObjectInputStream.html#readUnshared())()            Reads an "unshared" object from the ObjectInputStream. |
| int | [**readUnsignedByte**](http://docs.google.com/java/io/ObjectInputStream.html#readUnsignedByte())()            Reads an unsigned 8 bit byte. |
| int | [**readUnsignedShort**](http://docs.google.com/java/io/ObjectInputStream.html#readUnsignedShort())()            Reads an unsigned 16 bit short. |
| [String](http://docs.google.com/java/lang/String.html) | [**readUTF**](http://docs.google.com/java/io/ObjectInputStream.html#readUTF())()            Reads a String in [modified UTF-8](http://docs.google.com/DataInput.html#modified-utf-8) format. |
| void | [**registerValidation**](http://docs.google.com/java/io/ObjectInputStream.html#registerValidation(java.io.ObjectInputValidation,%20int))([ObjectInputValidation](http://docs.google.com/java/io/ObjectInputValidation.html) obj, int prio)            Register an object to be validated before the graph is returned. |
| protected  [Class](http://docs.google.com/java/lang/Class.html)<?> | [**resolveClass**](http://docs.google.com/java/io/ObjectInputStream.html#resolveClass(java.io.ObjectStreamClass))([ObjectStreamClass](http://docs.google.com/java/io/ObjectStreamClass.html) desc)            Load the local class equivalent of the specified stream class description. |
| protected  [Object](http://docs.google.com/java/lang/Object.html) | [**resolveObject**](http://docs.google.com/java/io/ObjectInputStream.html#resolveObject(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            This method will allow trusted subclasses of ObjectInputStream to substitute one object for another during deserialization. |
| protected  [Class](http://docs.google.com/java/lang/Class.html)<?> | [**resolveProxyClass**](http://docs.google.com/java/io/ObjectInputStream.html#resolveProxyClass(java.lang.String%5B%5D))([String](http://docs.google.com/java/lang/String.html)[] 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**](http://docs.google.com/java/io/ObjectInputStream.html#skipBytes(int))(int len)            Skips bytes. |

| **Methods inherited from class java.io.**[**InputStream**](http://docs.google.com/java/io/InputStream.html) |
| --- |
| [mark](http://docs.google.com/java/io/InputStream.html#mark(int)), [markSupported](http://docs.google.com/java/io/InputStream.html#markSupported()), [read](http://docs.google.com/java/io/InputStream.html#read(byte%5B%5D)), [reset](http://docs.google.com/java/io/InputStream.html#reset()), [skip](http://docs.google.com/java/io/InputStream.html#skip(long)) |

| **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)) |

| **Methods inherited from interface java.io.**[**ObjectInput**](http://docs.google.com/java/io/ObjectInput.html) |
| --- |
| [read](http://docs.google.com/java/io/ObjectInput.html#read(byte%5B%5D)), [skip](http://docs.google.com/java/io/ObjectInput.html#skip(long)) |

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

### ObjectInputStream

public **ObjectInputStream**([InputStream](http://docs.google.com/java/io/InputStream.html) in)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Creates an ObjectInputStream that reads from the specified InputStream. A serialization stream header is read from the stream and verified. This constructor will block until the corresponding ObjectOutputStream has written and flushed 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 ObjectInputStream.readFields or ObjectInputStream.readUnshared methods.

**Parameters:**in - input stream to read from **Throws:** [StreamCorruptedException](http://docs.google.com/java/io/StreamCorruptedException.html) - if the stream header is incorrect [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs while reading 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 in is null**See Also:**[ObjectInputStream()](http://docs.google.com/java/io/ObjectInputStream.html#ObjectInputStream()), [readFields()](http://docs.google.com/java/io/ObjectInputStream.html#readFields()), [ObjectOutputStream.ObjectOutputStream(OutputStream)](http://docs.google.com/java/io/ObjectOutputStream.html#ObjectOutputStream(java.io.OutputStream))

### ObjectInputStream

protected **ObjectInputStream**()  
 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 ObjectInputStream to not have to allocate private data just used by this implementation of ObjectInputStream.

If there is a security manager installed, this method first calls the security manager's checkPermission method with the 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** |
| --- |

### readObject

public final [Object](http://docs.google.com/java/lang/Object.html) **readObject**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html)

Read an object from the ObjectInputStream. 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 read. Default deserializing for a class can be overriden using the writeObject and readObject methods. Objects referenced by this object are read transitively so that a complete equivalent graph of objects is reconstructed by readObject.

The root object is completely restored when all of its fields and the objects it references are completely restored. At this point the object validation callbacks are executed in order based on their registered priorities. The callbacks are registered by objects (in the readObject special methods) as they are individually restored.

Exceptions are thrown for problems with the InputStream and for classes that should not be deserialized. All exceptions are fatal to the InputStream and leave it in an indeterminate state; it is up to the caller to ignore or recover the stream state.

**Specified by:**[readObject](http://docs.google.com/java/io/ObjectInput.html#readObject()) in interface [ObjectInput](http://docs.google.com/java/io/ObjectInput.html) **Returns:**the object read from the stream **Throws:** [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - Class of a serialized object cannot be found. [InvalidClassException](http://docs.google.com/java/io/InvalidClassException.html) - Something is wrong with a class used by serialization. [StreamCorruptedException](http://docs.google.com/java/io/StreamCorruptedException.html) - Control information in the stream is inconsistent. [OptionalDataException](http://docs.google.com/java/io/OptionalDataException.html) - Primitive data was found in the stream instead of objects. [IOException](http://docs.google.com/java/io/IOException.html) - Any of the usual Input/Output related exceptions.

### readObjectOverride

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

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

**Returns:**the Object read from the stream. **Throws:** [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - Class definition of a serialized object cannot be found. [OptionalDataException](http://docs.google.com/java/io/OptionalDataException.html) - Primitive data was found in the stream instead of objects. [IOException](http://docs.google.com/java/io/IOException.html) - if I/O errors occurred while reading from the underlying stream**Since:** 1.2 **See Also:**[ObjectInputStream()](http://docs.google.com/java/io/ObjectInputStream.html#ObjectInputStream()), [readObject()](http://docs.google.com/java/io/ObjectInputStream.html#readObject())

### readUnshared

public [Object](http://docs.google.com/java/lang/Object.html) **readUnshared**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html)

Reads an "unshared" object from the ObjectInputStream. This method is identical to readObject, except that it prevents subsequent calls to readObject and readUnshared from returning additional references to the deserialized instance obtained via this call. Specifically:

* If readUnshared is called to deserialize a back-reference (the stream representation of an object which has been written previously to the stream), an ObjectStreamException will be thrown.
* If readUnshared returns successfully, then any subsequent attempts to deserialize back-references to the stream handle deserialized by readUnshared will cause an ObjectStreamException to be thrown.

Deserializing an object via readUnshared invalidates the stream handle associated with the returned object. Note that this in itself does not always guarantee that the reference returned by readUnshared is unique; the deserialized object may define a readResolve method which returns an object visible to other parties, or readUnshared may return a Class object or enum constant obtainable elsewhere in the stream or through external means. If the deserialized object defines a readResolve method and the invocation of that method returns an array, then readUnshared returns a shallow clone of that array; this guarantees that the returned array object is unique and cannot be obtained a second time from an invocation of readObject or readUnshared on the ObjectInputStream, even if the underlying data stream has been manipulated.

ObjectInputStream 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.

**Returns:**reference to deserialized object **Throws:** [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - if class of an object to deserialize cannot be found [StreamCorruptedException](http://docs.google.com/java/io/StreamCorruptedException.html) - if control information in the stream is inconsistent [ObjectStreamException](http://docs.google.com/java/io/ObjectStreamException.html) - if object to deserialize has already appeared in stream [OptionalDataException](http://docs.google.com/java/io/OptionalDataException.html) - if primitive data is next in stream [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs during deserialization**Since:** 1.4

### defaultReadObject

public void **defaultReadObject**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html)

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

**Throws:** [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - if the class of a serialized object could not be found. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs. [NotActiveException](http://docs.google.com/java/io/NotActiveException.html) - if the stream is not currently reading objects.

### readFields

public [ObjectInputStream.GetField](http://docs.google.com/java/io/ObjectInputStream.GetField.html) **readFields**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html)

Reads the persistent fields from the stream and makes them available by name.

**Returns:**the GetField object representing the persistent fields of the object being deserialized **Throws:** [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - if the class of a serialized object could not be found. [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O error occurs. [NotActiveException](http://docs.google.com/java/io/NotActiveException.html) - if the stream is not currently reading objects.**Since:** 1.2

### registerValidation

public void **registerValidation**([ObjectInputValidation](http://docs.google.com/java/io/ObjectInputValidation.html) obj,  
 int prio)  
 throws [NotActiveException](http://docs.google.com/java/io/NotActiveException.html),  
 [InvalidObjectException](http://docs.google.com/java/io/InvalidObjectException.html)

Register an object to be validated before the graph is returned. While similar to resolveObject these validations are called after the entire graph has been reconstituted. Typically, a readObject method will register the object with the stream so that when all of the objects are restored a final set of validations can be performed.

**Parameters:**obj - the object to receive the validation callback.prio - controls the order of callbacks;zero is a good default. Use higher numbers to be called back earlier, lower numbers for later callbacks. Within a priority, callbacks are processed in no particular order. **Throws:** [NotActiveException](http://docs.google.com/java/io/NotActiveException.html) - The stream is not currently reading objects so it is invalid to register a callback. [InvalidObjectException](http://docs.google.com/java/io/InvalidObjectException.html) - The validation object is null.

### resolveClass

protected [Class](http://docs.google.com/java/lang/Class.html)<?> **resolveClass**([ObjectStreamClass](http://docs.google.com/java/io/ObjectStreamClass.html) desc)  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html)

Load the local class equivalent of the specified stream class description. Subclasses may implement this method to allow classes to be fetched from an alternate source.

The corresponding method in ObjectOutputStream is annotateClass. This method will be invoked only once for each unique class in the stream. This method can be implemented by subclasses to use an alternate loading mechanism but must return a Class object. Once returned, if the class is not an array class, its serialVersionUID is compared to the serialVersionUID of the serialized class, and if there is a mismatch, the deserialization fails and an [InvalidClassException](http://docs.google.com/java/io/InvalidClassException.html) is thrown.

The default implementation of this method in ObjectInputStream returns the result of calling

Class.forName(desc.getName(), false, loader)

where loader is determined as follows: if there is a method on the current thread's stack whose declaring class was defined by a user-defined class loader (and was not a generated to implement reflective invocations), then loader is class loader corresponding to the closest such method to the currently executing frame; otherwise, loader is null. If this call results in a ClassNotFoundException and the name of the passed ObjectStreamClass instance is the Java language keyword for a primitive type or void, then the Class object representing that primitive type or void will be returned (e.g., an ObjectStreamClass with the name "int" will be resolved to Integer.TYPE). Otherwise, the ClassNotFoundException will be thrown to the caller of this method.

**Parameters:**desc - an instance of class ObjectStreamClass **Returns:**a Class object corresponding to desc **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - any of the usual Input/Output exceptions. [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - if class of a serialized object cannot be found.

### resolveProxyClass

protected [Class](http://docs.google.com/java/lang/Class.html)<?> **resolveProxyClass**([String](http://docs.google.com/java/lang/String.html)[] interfaces)  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html)

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.

This method is called exactly once for each unique proxy class descriptor in the stream.

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

The default implementation of this method in ObjectInputStream returns the result of calling Proxy.getProxyClass with the list of Class objects for the interfaces that are named in the interfaces parameter. The Class object for each interface name i is the value returned by calling

Class.forName(i, false, loader)

where loader is that of the first non-null class loader up the execution stack, or null if no non-null class loaders are on the stack (the same class loader choice used by the resolveClass method). Unless any of the resolved interfaces are non-public, this same value of loader is also the class loader passed to Proxy.getProxyClass; if non-public interfaces are present, their class loader is passed instead (if more than one non-public interface class loader is encountered, an IllegalAccessError is thrown). If Proxy.getProxyClass throws an IllegalArgumentException, resolveProxyClass will throw a ClassNotFoundException containing the IllegalArgumentException.

**Parameters:**interfaces - the list of interface names that were deserialized in the proxy class descriptor **Returns:**a proxy class for the specified interfaces **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - any exception thrown by the underlying InputStream [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - if the proxy class or any of the named interfaces could not be found**Since:** 1.3 **See Also:**[ObjectOutputStream.annotateProxyClass(Class)](http://docs.google.com/java/io/ObjectOutputStream.html#annotateProxyClass(java.lang.Class))

### resolveObject

protected [Object](http://docs.google.com/java/lang/Object.html) **resolveObject**([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 ObjectInputStream to substitute one object for another during deserialization. Replacing objects is disabled until enableResolveObject is called. The enableResolveObject method checks that the stream requesting to resolve object can be trusted. Every reference to serializable objects is passed to resolveObject. To insure that the private state of objects is not unintentionally exposed only trusted streams may use resolveObject.

This method is called after an object has been read but before it is returned from readObject. The default resolveObject method just returns the same object.

When a subclass is replacing objects it must insure 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.

**Parameters:**obj - object to be substituted **Returns:**the substituted object **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - Any of the usual Input/Output exceptions.

### enableResolveObject

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

Enable the stream to allow objects read from the stream to be replaced. When enabled, the resolveObject method is called for every object being deserialized.

If *enable* is true, and there is a security manager installed, this method first calls the security manager's checkPermission method with the SerializablePermission("enableSubstitution") permission to ensure it's ok to enable the stream to allow objects read from the stream to be replaced.

**Parameters:**enable - true for enabling use of resolveObject for every object being deserialized **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 allow objects read from the stream to be replaced.**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)

### readStreamHeader

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

The readStreamHeader method is provided to allow subclasses to read and verify their own stream headers. It reads and verifies the magic number and version number.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if there are I/O errors while reading from the underlying InputStream [StreamCorruptedException](http://docs.google.com/java/io/StreamCorruptedException.html) - if control information in the stream is inconsistent

### readClassDescriptor

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

Read a class descriptor from the serialization stream. This method is called when the ObjectInputStream expects a class descriptor as the next item in the serialization stream. Subclasses of ObjectInputStream may override this method to read in class descriptors that have been written in non-standard formats (by subclasses of ObjectOutputStream which have overridden the writeClassDescriptor method). By default, this method reads class descriptors according to the format defined in the Object Serialization specification.

**Returns:**the class descriptor read **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error has occurred. [ClassNotFoundException](http://docs.google.com/java/lang/ClassNotFoundException.html) - If the Class of a serialized object used in the class descriptor representation cannot be found**Since:** 1.3 **See Also:**[ObjectOutputStream.writeClassDescriptor(java.io.ObjectStreamClass)](http://docs.google.com/java/io/ObjectOutputStream.html#writeClassDescriptor(java.io.ObjectStreamClass))

### read

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

Reads a byte of data. This method will block if no input is available.

**Specified by:**[read](http://docs.google.com/java/io/ObjectInput.html#read()) in interface [ObjectInput](http://docs.google.com/java/io/ObjectInput.html)**Specified by:**[read](http://docs.google.com/java/io/InputStream.html#read()) in class [InputStream](http://docs.google.com/java/io/InputStream.html) **Returns:**the byte read, or -1 if the end of the stream is reached. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error has occurred.

### read

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

Reads into an array of bytes. This method will block until some input is available. Consider using java.io.DataInputStream.readFully to read exactly 'length' bytes.

**Specified by:**[read](http://docs.google.com/java/io/ObjectInput.html#read(byte%5B%5D,%20int,%20int)) in interface [ObjectInput](http://docs.google.com/java/io/ObjectInput.html)**Overrides:**[read](http://docs.google.com/java/io/InputStream.html#read(byte%5B%5D,%20int,%20int)) in class [InputStream](http://docs.google.com/java/io/InputStream.html) **Parameters:**buf - the buffer into which the data is readoff - the start offset of the datalen - the maximum number of bytes read **Returns:**the actual number of bytes read, -1 is returned when the end of the stream is reached. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error has occurred.**See Also:**[DataInputStream.readFully(byte[],int,int)](http://docs.google.com/java/io/DataInputStream.html#readFully(byte%5B%5D,%20int,%20int))

### available

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

Returns the number of bytes that can be read without blocking.

**Specified by:**[available](http://docs.google.com/java/io/ObjectInput.html#available()) in interface [ObjectInput](http://docs.google.com/java/io/ObjectInput.html)**Overrides:**[available](http://docs.google.com/java/io/InputStream.html#available()) in class [InputStream](http://docs.google.com/java/io/InputStream.html) **Returns:**the number of available bytes. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if there are I/O errors while reading from the underlying InputStream

### close

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

Closes the input stream. 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/ObjectInput.html#close()) in interface [ObjectInput](http://docs.google.com/java/io/ObjectInput.html)**Overrides:**[close](http://docs.google.com/java/io/InputStream.html#close()) in class [InputStream](http://docs.google.com/java/io/InputStream.html) **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error has occurred.

### readBoolean

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

Reads in a boolean.

**Specified by:**[readBoolean](http://docs.google.com/java/io/DataInput.html#readBoolean()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**the boolean read. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - If end of file is reached. [IOException](http://docs.google.com/java/io/IOException.html) - If other I/O error has occurred.

### readByte

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

Reads an 8 bit byte.

**Specified by:**[readByte](http://docs.google.com/java/io/DataInput.html#readByte()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**the 8 bit byte read. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - If end of file is reached. [IOException](http://docs.google.com/java/io/IOException.html) - If other I/O error has occurred.

### readUnsignedByte

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

Reads an unsigned 8 bit byte.

**Specified by:**[readUnsignedByte](http://docs.google.com/java/io/DataInput.html#readUnsignedByte()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**the 8 bit byte read. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - If end of file is reached. [IOException](http://docs.google.com/java/io/IOException.html) - If other I/O error has occurred.

### readChar

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

Reads a 16 bit char.

**Specified by:**[readChar](http://docs.google.com/java/io/DataInput.html#readChar()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**the 16 bit char read. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - If end of file is reached. [IOException](http://docs.google.com/java/io/IOException.html) - If other I/O error has occurred.

### readShort

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

Reads a 16 bit short.

**Specified by:**[readShort](http://docs.google.com/java/io/DataInput.html#readShort()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**the 16 bit short read. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - If end of file is reached. [IOException](http://docs.google.com/java/io/IOException.html) - If other I/O error has occurred.

### readUnsignedShort

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

Reads an unsigned 16 bit short.

**Specified by:**[readUnsignedShort](http://docs.google.com/java/io/DataInput.html#readUnsignedShort()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**the 16 bit short read. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - If end of file is reached. [IOException](http://docs.google.com/java/io/IOException.html) - If other I/O error has occurred.

### readInt

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

Reads a 32 bit int.

**Specified by:**[readInt](http://docs.google.com/java/io/DataInput.html#readInt()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**the 32 bit integer read. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - If end of file is reached. [IOException](http://docs.google.com/java/io/IOException.html) - If other I/O error has occurred.

### readLong

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

Reads a 64 bit long.

**Specified by:**[readLong](http://docs.google.com/java/io/DataInput.html#readLong()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**the read 64 bit long. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - If end of file is reached. [IOException](http://docs.google.com/java/io/IOException.html) - If other I/O error has occurred.

### readFloat

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

Reads a 32 bit float.

**Specified by:**[readFloat](http://docs.google.com/java/io/DataInput.html#readFloat()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**the 32 bit float read. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - If end of file is reached. [IOException](http://docs.google.com/java/io/IOException.html) - If other I/O error has occurred.

### readDouble

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

Reads a 64 bit double.

**Specified by:**[readDouble](http://docs.google.com/java/io/DataInput.html#readDouble()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**the 64 bit double read. **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - If end of file is reached. [IOException](http://docs.google.com/java/io/IOException.html) - If other I/O error has occurred.

### readFully

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

Reads bytes, blocking until all bytes are read.

**Specified by:**[readFully](http://docs.google.com/java/io/DataInput.html#readFully(byte%5B%5D)) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Parameters:**buf - the buffer into which the data is read **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - If end of file is reached. [IOException](http://docs.google.com/java/io/IOException.html) - If other I/O error has occurred.

### readFully

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

Reads bytes, blocking until all bytes are read.

**Specified by:**[readFully](http://docs.google.com/java/io/DataInput.html#readFully(byte%5B%5D,%20int,%20int)) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Parameters:**buf - the buffer into which the data is readoff - the start offset of the datalen - the maximum number of bytes to read **Throws:** [EOFException](http://docs.google.com/java/io/EOFException.html) - If end of file is reached. [IOException](http://docs.google.com/java/io/IOException.html) - If other I/O error has occurred.

### skipBytes

public int **skipBytes**(int len)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Skips bytes.

**Specified by:**[skipBytes](http://docs.google.com/java/io/DataInput.html#skipBytes(int)) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Parameters:**len - the number of bytes to be skipped **Returns:**the actual number of bytes skipped. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - If an I/O error has occurred.

### readLine

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public [String](http://docs.google.com/java/lang/String.html) **readLine**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

**Deprecated.** *This method does not properly convert bytes to characters. see DataInputStream for the details and alternatives.*

Reads in a line that has been terminated by a \n, \r, \r\n or EOF.

**Specified by:**[readLine](http://docs.google.com/java/io/DataInput.html#readLine()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**a String copy of the line. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if there are I/O errors while reading from the underlying InputStream

### readUTF

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

Reads a String in [modified UTF-8](http://docs.google.com/DataInput.html#modified-utf-8) format.

**Specified by:**[readUTF](http://docs.google.com/java/io/DataInput.html#readUTF()) in interface [DataInput](http://docs.google.com/java/io/DataInput.html) **Returns:**the String. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if there are I/O errors while reading from the underlying InputStream [UTFDataFormatException](http://docs.google.com/java/io/UTFDataFormatException.html) - if read bytes do not represent a valid modified UTF-8 encoding of a string

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ObjectInputStream.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/ObjectInput.html)   [**NEXT CLASS**](http://docs.google.com/java/io/ObjectInputStream.GetField.html) | [**FRAMES**](http://docs.google.com/index.html?java/io/ObjectInputStream.html)    [**NO FRAMES**](http://docs.google.com/ObjectInputStream.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#35nkun2) |

[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).