

DataOutputStream and DataInputStream Class

12 September 2025 09:04

Java DataOutputStream class allows an application to write primitive Java data types to the output stream in a machine-independent way.

Java application generally uses the data output stream to write data that can later be read by a DataInputStream.

Constructors

Constructor	Description
<code>DataOutputStream(OutputStream out)</code>	Creates a new data output stream to write data to the specified underlying output stream.

Method Summary

All Methods	Instance Methods	Concrete Methods	Description
Modifier and Type	Method		
void	<code>flush()</code>		Flushes this data output stream.
final int	<code>size()</code>		Returns the current value of the counter <code>written</code> , the number of bytes written to this data output stream so far.
void	<code>write(byte[] b, int off, int len)</code>		Writes <code>len</code> bytes from the specified byte array starting at offset <code>off</code> to the underlying output stream.
void	<code>write(int b)</code>		Writes the specified byte (the low eight bits of the argument <code>b</code>) to the underlying output stream.
final void	<code>writeBoolean(boolean v)</code>		Writes a <code>boolean</code> to the underlying output stream as a 1-byte value.
final void	<code>writeByte(int v)</code>		Writes out a <code>byte</code> to the underlying output stream as a 1-byte value.
final void	<code>writeBytes(String s)</code>		Writes out the string to the underlying output stream as a sequence of bytes.
final void	<code>writeChar(int v)</code>		Writes a <code>char</code> to the underlying output stream as a 2-byte value, high byte first.
final void	<code>writeChars(String s)</code>		Writes a string to the underlying output stream as a sequence of characters.
final void	<code>writeDouble(double v)</code>		Converts the double argument to a <code>long</code> using the <code>doubleToLongBits</code> method in class <code>Double</code> , and then writes that <code>long</code> value to the underlying output stream as an 8-byte quantity, high byte first.
final void	<code>writeFloat(float v)</code>		Converts the float argument to an <code>int</code> using the <code>floatToIntBits</code> method in class <code>Float</code> , and then writes that <code>int</code> value to the underlying output stream as a 4-byte quantity, high byte first.
final void	<code>writeInt(int v)</code>		Writes an <code>int</code> to the underlying output stream as four bytes, high byte first.
final void	<code>writeLong(long v)</code>		Writes a <code>long</code> to the underlying output stream as eight bytes, high byte first.
final void	<code>writeShort(int v)</code>		Writes a <code>short</code> to the underlying output stream as two bytes, high byte first.
final void	<code>writeUTF(String str)</code>		Writes a string to the underlying output stream using modified UTF-8 encoding in a machine-independent manner.

```
package FileHandling;
```

```
import javax.swing.*;  
import java.io.*;
```

```
/**
```

```

* Write a description of class DataOutputStreamClassExample here.
*
* @author (your name)
* @version (a version number or a date)
*/
public class DataOutputStreamClassExample
{
    public static void main(String[] args){
        String path = "../JavaSem2/DataFiles/";
        String fileName;
        fileName = JOptionPane.showInputDialog(null, "Enter file name: ",
        "Input file name", JOptionPane.QUESTION_MESSAGE);

        try(FileOutputStream fos = new FileOutputStream(path+fileName);
            DataOutputStream dos = new DataOutputStream(fos)){

            int i = 65;
            boolean b = true;
            char x = 'Z';
            String str = "Hello";
            double d = 3.14;
            float f = 6.28f;
            long l = 45676543454678l;
            byte bt = 100;
            short s = 25981;

            dos.writeInt(i);
            dos.writeBoolean(b);
            dos.writeChar(x);
            dos.writeUTF(str);
            dos.writeDouble(d);
            dos.writeFloat(f);
            dos.writeLong(l);
            dos.writeByte(bt);
            dos.writeShort(s);

            dos.flush();

            JOptionPane.showMessageDialog(null, "File is created!",
            "Success", JOptionPane.INFORMATION_MESSAGE);
        }
        catch(FileNotFoundException fnfe){
            JOptionPane.showMessageDialog(null, "File not found on the specified path",
            "Error Message", JOptionPane.ERROR_MESSAGE);
        }
        catch(IOException ioe){
            JOptionPane.showMessageDialog(null, "File is used by other application",
            "IOException Error Message", JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

Output:

```
File Edit View  
| A@ Z @Hello@ @, Që...@ÈöÃ )Šæ%Öde} |
```

data is in encoded form .

Java DataInputStream Class

Java **DataInputStream** class allows an application to read primitive data from the input stream in a machine independent way.

Java application generally uses the data output stream to write data that can later be read by a data input stream.

Constructors					
Constructor	Description				
<code>DataInputStream(InputStream in)</code> Creates a DataInputStream that uses the specified underlying InputStream.					
Method Summary					
All Methods	Static Methods	Instance Methods	Concrete Methods		
Modifier and Type	Method	Description			
final int	<code>read(byte[] b)</code>	Reads some number of bytes from the contained input stream and stores them into the buffer array <code>b</code> .			
final int	<code>read(byte[] b, int off, int len)</code>	Reads up to <code>len</code> bytes of data from the contained input stream into an array of bytes.			
final boolean	<code>readBoolean()</code>	See the general contract of the <code>readBoolean</code> method of <code>DataInput</code> .			
final byte	<code>readByte()</code>	See the general contract of the <code>readByte</code> method of <code>DataInput</code> .			
final char	<code>readChar()</code>	See the general contract of the <code>readChar</code> method of <code>DataInput</code> .			
final double	<code>readDouble()</code>	See the general contract of the <code>readDouble</code> method of <code>DataInput</code> .			
final float	<code>readFloat()</code>	See the general contract of the <code>readFloat</code> method of <code>DataInput</code> .			
final void	<code>readFully(byte[] b)</code>	See the general contract of the <code>readFully</code> method of <code>DataInput</code> .			
final void	<code>readFully(byte[] b, int off, int len)</code>	See the general contract of the <code>readFully</code> method of <code>DataInput</code> .			
final int	<code>readInt()</code>	See the general contract of the <code>readInt</code> method of <code>DataInput</code> .			
final String	<code>readLine()</code>	Deprecated. This method does not properly convert bytes to characters.			
final long	<code>readLong()</code>	See the general contract of the <code>readLong</code> method of <code>DataInput</code> .			
final short	<code>readShort()</code>	See the general contract of the <code>readShort</code> method of <code>DataInput</code> .			
final int	<code>readUnsignedByte()</code>	See the general contract of the <code>readUnsignedByte</code> method of <code>DataInput</code> .			

Method Summary

All Methods	Static Methods	Instance Methods	Concrete Methods	Deprecated Methods
Modifier and Type	Method	Description		
final int	<code>read(byte[] b)</code>	Reads some number of bytes from the contained input stream and stores them into the buffer array <code>b</code> .		
final int	<code>read(byte[] b, int off, int len)</code>	Reads up to <code>len</code> bytes of data from the contained input stream into an array of bytes.		
final boolean	<code>readBoolean()</code>	See the general contract of the <code>readBoolean</code> method of <code>DataInput</code> .		
final byte	<code>readByte()</code>	See the general contract of the <code>readByte</code> method of <code>DataInput</code> .		
final char	<code>readChar()</code>	See the general contract of the <code>readChar</code> method of <code>DataInput</code> .		
final double	<code>readDouble()</code>	See the general contract of the <code>readDouble</code> method of <code>DataInput</code> .		
final float	<code>readFloat()</code>	See the general contract of the <code>readFloat</code> method of <code>DataInput</code> .		
final void	<code>readFully(byte[] b)</code>	See the general contract of the <code>readFully</code> method of <code>DataInput</code> .		
final void	<code>readFully(byte[] b, int off, int len)</code>	See the general contract of the <code>readFully</code> method of <code>DataInput</code> .		
final int	<code>readInt()</code>	See the general contract of the <code>readInt</code> method of <code>DataInput</code> .		
final String	<code>readLine()</code>	Deprecated. This method does not properly convert bytes to characters.		
final long	<code>readLong()</code>	See the general contract of the <code>readLong</code> method of <code>DataInput</code> .		
final short	<code>readShort()</code>	See the general contract of the <code>readShort</code> method of <code>DataInput</code> .		
final int	<code>readUnsignedByte()</code>	See the general contract of the <code>readUnsignedByte</code> method of <code>DataInput</code> .		
final int	<code>readUnsignedShort()</code>	See the general contract of the <code>readUnsignedShort</code> method of <code>DataInput</code> .		
final String	<code>readUTF()</code>	See the general contract of the <code>readUTF</code> method of <code>DataInput</code> .		
static final String	<code>readUTF(DataInput in)</code>	Reads from the stream <code>in</code> a representation of a Unicode character string encoded in modified UTF-8 format; this string of characters is then returned as a <code>String</code> .		
final int	<code>skipBytes(int n)</code>	See the general contract of the <code>skipBytes</code> method of <code>DataInput</code> .		

```

int   available()
void  close()
boolean equals(Object)
FClass<?> getClass()
int   hashCode()
void  mark(int)
boolean markSupported()
FVoid  notify()
void  notifyAll()
InputsS.. nullInputStream()
Eint   read()
Fint   read(byte[])
Eint   read(byte[], int, int)
byte[]  readAllBytes()
    
```

`int available()`

Returns an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking by the next caller of a method for this input stream. The next caller might be the same thread or another thread. A single read or skip of this many bytes will not block, but may read or skip fewer bytes.

implSpec - This method returns the result of `in.available()`.
return - an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking.
throws - `IOException` {@inheritDoc}

```

int    read ()
int    read (byte[])
int    read (byte[], int, int)
byte[]  readAllBytes()
boolean readBoolean()
byte   readByte()
char   readChar()
double readDouble()
float  readFloat()
void   readFully(byte[])
void   readFully(byte[], int, int)
int    readInt()
String  readLine()
long   readLong()

```

int read(byte[] b)

Reads some number of bytes from the contained input stream and stores them into the buffer array `b`. The number of bytes actually read is returned as an integer. This method blocks until input data is available, end of file is detected, or an exception is thrown.

If `b` is null, a `NullPointerException` is thrown. If the length of `b` is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into `b`.

The first byte read is stored into element `b[0]`, the next one into `b[1]`, and so on. The number of bytes read is, at most, equal to the length of `b`. Let `k` be the number of bytes actually read; these bytes will be stored in elements `b[0]` through `b[k-1]`, leaving elements `b[k]` through `b[b.length-1]` unaffected.

```

String  toString(boolean[])
String  toString(byte[])
String  toString(char[])
String  toString(double[])
String  toString(float[])
String  toString(int[])
String  toString(Object[])
String  toString(long[])
String  toString(short[])

```

String toString(byte[] a)

Returns a string representation of the contents of the specified array. The string representation consists of a list of the array's elements, enclosed in square brackets ("[]"). Adjacent elements are separated by the characters ", " (a comma followed by a space). Elements are converted to strings as by `String.valueOf(byte)`. Returns "null" if `a` is null.

Parameters

a - the array whose string representation to return

return - a string representation of a
since - 1.5

```

package FileHandling;
import java.io.*;
import javax.swing.*;

```

```

/**
 * Write a description of class ReadingTextFileUsingDataInputStream here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class ReadingTextFileUsingDataInputStream
{
    public static void main(String[] args){
        String path = "../JavaSem2/DataFiles/";
        String fileName;
        fileName = JOptionPane.showInputDialog(null, "Enter file name: ",
            "Input file name", JOptionPane.QUESTION_MESSAGE);

        try(FileInputStream fos = new FileInputStream(path+fileName);
            DataInputStream dis = new DataInputStream(fos)){
            int count = dis.available();
            byte[] any = new byte[count];
            dis.read(any);
            String _message_ = "";
            for(byte c: any){
                _message_ += (char)c;
            }
            JOptionPane.showMessageDialog(null, _message_);
        }
    }
}

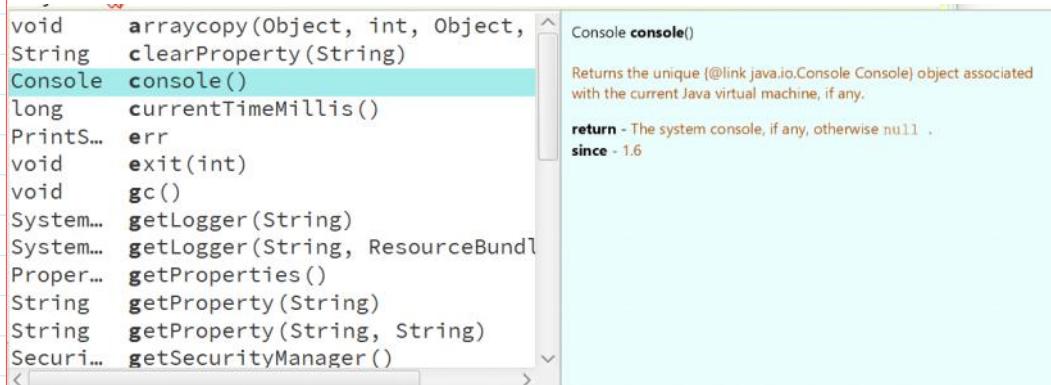
```

```

        "Content", JOptionPane.INFORMATION_MESSAGE);
    }
    catch(FileNotFoundException fnfe){
        JOptionPane.showMessageDialog(null, "File not found on the specified path",
            "Error Message", JOptionPane.ERROR_MESSAGE);
    }
    catch(IOException ioe){
        JOptionPane.showMessageDialog(null, "File is used by other application",
            "IOException Error Message", JOptionPane.ERROR_MESSAGE);
    }
}
}

```

Console class:



Method Summary

Methods	Modifier and Type	Method and Description
	void	flush() Flushes the console and forces any buffered output to be written immediately .
	Console	format(String fmt, Object... args) Writes a formated string to this console's output stream using the specified format string and arguments.
	Console	printf(String format, Object... args) A convenience method to write a formated string to this console's output stream using the specified format string and arguments.
	Reader	reader() Retrieves the unique Reader object associated with this console.
	String	readLine() Reads a single line of text from the console.
	String	readLine(String fmt, Object... args) Provides a formated prompt, then reads a single line of text from the console.
	char[]	readPassword() Reads a password or passphrase from the console with echoing disabled
	char[]	readPassword(String fmt, Object... args) Provides a formated prompt, then reads a password or passphrase from the console with echoing disabled.
	PrintWriter	writer() Retrieves the unique PrintWriter object associated with this console.

```

import java.io.Console;
public class ConsoleClass {
    public static void main(String args[]){
        Console c = System.console();
        if(c == null){
            System.out.print("No console available");
        }
        String name = c.readLine("Enter your name: ");
    }
}

```

```
c.printf("%s", name);
c.printf("%s", "\nEnter your password: ");
char pass[] = c.readPassword();
String password = String.valueOf(pass);
c.printf("%s%s", "Password is: ",password);
}
}
```

Output:

```
Enter your name: Satyender
Satyender
Enter your password:

Password is: password
```