Java BufferedReader Class

Java BufferedReader class is used to read the text from a character-based input stream. It can be used to read data line by line by readLine() method. It makes the performance fast. It inherits [Reader](https://www.javatpoint.com/java-reader-class) [class](https://www.javatpoint.com/object-and-class-in-java).

Java BufferedReader class declaration

Let's see the declaration for Java.io.BufferedReader class:

1. **public** **class** BufferedReader **extends** Reader

Java BufferedReader class constructors

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| BufferedReader(Reader rd) | It is used to create a buffered character input stream that uses the default size for an input buffer. |
| BufferedReader(Reader rd, int size) | It is used to create a buffered character input stream that uses the specified size for an input buffer. |

Java BufferedReader class methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| int read() | It is used for reading a single character. |
| int read(char[] cbuf, int off, int len) | It is used for reading characters into a portion of an [array](https://www.javatpoint.com/array-in-java). |
| boolean markSupported() | It is used to test the input stream support for the mark and reset method. |
| String readLine() | It is used for reading a line of text. |
| boolean ready() | It is used to test whether the input stream is ready to be read. |
| long skip(long n) | It is used for skipping the characters. |
| void reset() | It repositions the [stream](https://www.javatpoint.com/java-8-stream) at a position the mark method was last called on this input stream. |
| void mark(int readAheadLimit) | It is used for marking the present position in a stream. |
| void close() | It closes the input stream and releases any of the system resources associated with the stream. |

Java BufferedReader Example

In this example, we are reading the data from the text file **testout.txt** using Java BufferedReader class.

1. **package** com.javatpoint;
2. **import** java.io.\*;
3. **public** **class** BufferedReaderExample {
4. **public** **static** **void** main(String args[])**throws** Exception{
5. FileReader fr=**new** FileReader("D:\\testout.txt");
6. BufferedReader br=**new** BufferedReader(fr);
8. **int** i;
9. **while**((i=br.read())!=-1){
10. System.out.print((**char**)i);
11. }
12. br.close();
13. fr.close();
14. }
15. }

Here, we are assuming that you have following data in "testout.txt" file:

Welcome to javaTpoint.

Output:

Welcome to javaTpoint.

Reading data from console by InputStreamReader and BufferedReader

In this example, we are connecting the BufferedReader stream with the [InputStreamReader](https://www.javatpoint.com/Input-from-keyboard-by-InputStreamReader) stream for reading the line by line data from the keyboard.

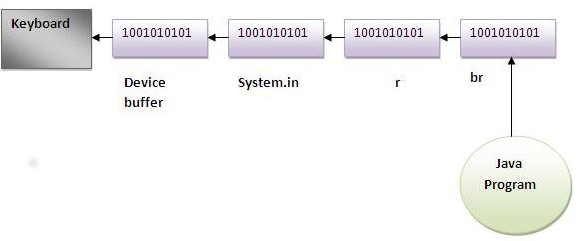
1. **package** com.javatpoint;
2. **import** java.io.\*;
3. **public** **class** BufferedReaderExample{
4. **public** **static** **void** main(String args[])**throws** Exception{
5. InputStreamReader r=**new** InputStreamReader(System.in);
6. BufferedReader br=**new** BufferedReader(r);
7. System.out.println("Enter your name");
8. String name=br.readLine();
9. System.out.println("Welcome "+name);
10. }
11. }

Output:

Enter your name

Nakul Jain

Welcome Nakul Jain



Another example of reading data from console until user writes stop

In this example, we are reading and printing the data until the user prints stop.

1. **package** com.javatpoint;
2. **import** java.io.\*;
3. **public** **class** BufferedReaderExample{
4. **public** **static** **void** main(String args[])**throws** Exception{
5. InputStreamReader r=**new** InputStreamReader(System.in);
6. BufferedReader br=**new** BufferedReader(r);
7. String name="";
8. **while**(!name.equals("stop")){
9. System.out.println("Enter data: ");
10. name=br.readLine();
11. System.out.println("data is: "+name);
12. }
13. br.close();
14. r.close();
15. }
16. }

Output:

Enter data: Nakul

data is: Nakul

Enter data: 12

data is: 12

Enter data: stop

data is: stop

Java BufferedWriter Class

Java BufferedWriter class is used to provide buffering for Writer instances. It makes the performance fast. It inherits [Writer](https://www.javatpoint.com/java-writer-class) class. The buffering characters are used for providing the efficient writing of single [arrays](https://www.javatpoint.com/array-in-java), characters, and [strings](https://www.javatpoint.com/java-string).

Class declaration

Let's see the declaration for Java.io.BufferedWriter class:

1. **public** **class** BufferedWriter **extends** Writer

Class constructors

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| BufferedWriter(Writer wrt) | It is used to create a buffered character output stream that uses the default size for an output buffer. |
| BufferedWriter(Writer wrt, int size) | It is used to create a buffered character output stream that uses the specified size for an output buffer. |

Class methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| void newLine() | It is used to add a new line by writing a line separator. |
| void write(int c) | It is used to write a single character. |
| void write(char[] cbuf, int off, int len) | It is used to write a portion of an array of characters. |
| void write(String s, int off, int len) | It is used to write a portion of a string. |
| void flush() | It is used to flushes the input stream. |
| void close() | It is used to closes the input stream |

Example of Java BufferedWriter

Let's see the simple example of writing the data to a text file **testout.txt** using Java BufferedWriter.

1. **package** com.javatpoint;
2. **import** java.io.\*;
3. **public** **class** BufferedWriterExample {
4. **public** **static** **void** main(String[] args) **throws** Exception {
5. FileWriter writer = **new** FileWriter("D:\\testout.txt");
6. BufferedWriter buffer = **new** BufferedWriter(writer);
7. buffer.write("Welcome to javaTpoint.");
8. buffer.close();
9. System.out.println("Success");
10. }
11. }

Output:

success

testout.txt:

Welcome to javaTpoint.

# Java DataOutputStream Class

Java DataOutputStream [class](https://www.javatpoint.com/object-and-class-in-java) allows an application to write primitive [Java](https://www.javatpoint.com/java-tutorial) 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 data input stream.

## Java DataOutputStream class declaration

Let's see the declaration for java.io.DataOutputStream class:

1. **public** **class** DataOutputStream **extends** FilterOutputStream **implements** DataOutput

## Java DataOutputStream class methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| int size() | It is used to return the number of bytes written to the data output stream. |
| void write(int b) | It is used to write the specified byte to the underlying output stream. |
| void write(byte[] b, int off, int len) | It is used to write len bytes of data to the output stream. |
| void writeBoolean(boolean v) | It is used to write Boolean to the output stream as a 1-byte value. |
| void writeChar(int v) | It is used to write char to the output stream as a 2-byte value. |
| void writeChars(String s) | It is used to write [string](https://www.javatpoint.com/java-string) to the output stream as a sequence of characters. |
| void writeByte(int v) | It is used to write a byte to the output stream as a 1-byte value. |
| void writeBytes(String s) | It is used to write string to the output stream as a sequence of bytes. |
| void writeInt(int v) | It is used to write an int to the output stream |
| void writeShort(int v) | It is used to write a short to the output stream. |
| void writeShort(int v) | It is used to write a short to the output stream. |
| void writeLong(long v) | It is used to write a long to the output stream. |
| void writeUTF(String str) | It is used to write a string to the output stream using UTF-8 encoding in portable manner. |
| void flush() | It is used to flushes the data output stream. |

### Example of DataOutputStream class

In this example, we are writing the data to a text file **testout.txt** using DataOutputStream class.

1. **package** com.javatpoint;
3. **import** java.io.\*;
4. **public** **class** OutputExample {
5. **public** **static** **void** main(String[] args) **throws** IOException {
6. FileOutputStream file = **new** FileOutputStream(D:\\testout.txt);
7. DataOutputStream data = **new** DataOutputStream(file);
8. data.writeInt(65);
9. data.flush();
10. data.close();
11. System.out.println("Succcess...");
12. }
13. }

Output:

Succcess...

testout.txt:

A

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**next →**](https://www.javatpoint.com/java-filteroutputstream-class)[**← prev**](https://www.javatpoint.com/java-dataoutputstream-class)  Java DataInputStream Class  Java DataInputStream [class](https://www.javatpoint.com/object-and-class-in-java) 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.  Java DataInputStream class declaration  Let's see the declaration for java.io.DataInputStream class:   1. **public** **class** DataInputStream **extends** FilterInputStream **implements** DataInput   Java DataInputStream class Methods   |  |  | | --- | --- | | **Method** | **Description** | | int read(byte[] b) | It is used to read the number of bytes from the input stream. | | int read(byte[] b, int off, int len) | It is used to read **len** bytes of data from the input stream. | | int readInt() | It is used to read input bytes and return an int value. | | byte readByte() | It is used to read and return the one input byte. | | char readChar() | It is used to read two input bytes and returns a char value. | | double readDouble() | It is used to read eight input bytes and returns a double value. | | boolean readBoolean() | It is used to read one input byte and return true if byte is non zero, false if byte is zero. | | int skipBytes(int x) | It is used to skip over x bytes of data from the input stream. | | String readUTF() | It is used to read a [string](https://www.javatpoint.com/java-string) that has been encoded using the UTF-8 format. | | void readFully(byte[] b) | It is used to read bytes from the input stream and store them into the buffer [array](https://www.javatpoint.com/array-in-java). | | void readFully(byte[] b, int off, int len) | It is used to read **len** bytes from the input stream. |   Example of DataInputStream class  In this example, we are reading the data from the file testout.txt file.   1. **package** com.javatpoint; 2. **import** java.io.\*; 3. **public** **class** DataStreamExample { 4. **public** **static** **void** main(String[] args) **throws** IOException { 5. InputStream input = **new** FileInputStream("D:\\testout.txt"); 6. DataInputStream inst = **new** DataInputStream(input); 7. **int** count = input.available(); 8. **byte**[] ary = **new** **byte**[count]; 9. inst.read(ary); 10. **for** (**byte** bt : ary) { 11. **char** k = (**char**) bt; 12. System.out.print(k+"-"); 13. } 14. } 15. }   Here, we are assuming that you have following data in **"testout.txt"** file:  JAVA  Output:  J-A-V-A |