Java PrintWriter class

Java PrintWriter class is the implementation of [Writer](https://www.javatpoint.com/java-writer-class) class. It is used to print the formatted representation of [objects](https://www.javatpoint.com/object-and-class-in-java) to the text-output stream.

## Class declaration

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

**public** **class** PrintWriter **extends** Writer

## Methods of PrintWriter class

|  |  |
| --- | --- |
| **Method** | **Description** |
| void println(boolean x) | It is used to print the boolean value. |
| void println(char[] x) | It is used to print an [array](https://www.javatpoint.com/array-in-java) of characters. |
| void println(int x) | It is used to print an integer. |
| PrintWriter append(char c) | It is used to append the specified character to the writer. |
| PrintWriter append(CharSequence ch) | It is used to append the specified character sequence to the writer. |
| PrintWriter append(CharSequence ch, int start, int end) | It is used to append a subsequence of specified character to the writer. |
| boolean checkError() | It is used to flushes the stream and check its error state. |
| protected void setError() | It is used to indicate that an error occurs. |
| protected void clearError() | It is used to clear the error state of a stream. |
| PrintWriter format(String format, Object... args) | It is used to write a formatted [string](https://www.javatpoint.com/java-string) to the writer using specified arguments and format string. |
| void print(Object obj) | It is used to print an object. |
| void flush() | It is used to flushes the stream. |
| void close() | It is used to close the stream. |

## Java PrintWriter Example

Let's see the simple example of writing the data on a **console** and in a **text file testout.txt** using Java PrintWriter class.

1. **package** com.javatpoint;
3. **import** java.io.File;
4. **import** java.io.PrintWriter;
5. **public** **class** PrintWriterExample {
6. **public** **static** **void** main(String[] args) **throws** Exception {
7. //Data to write on Console using PrintWriter
8. PrintWriter writer = **new** PrintWriter(System.out);
9. writer.write("Javatpoint provides tutorials of all technology.");
10. writer.flush();
11. writer.close();
12. //Data to write in File using PrintWriter
13. PrintWriter writer1 =**null**;
14. writer1 = **new** PrintWriter(**new** File("D:\\testout.txt"));
15. writer1.write("Like Java, Spring, Hibernate, Android, PHP etc.");
16. writer1.flush();
17. writer1.close();
18. }
19. }

Outpt

Javatpoint provides tutorials of all technology.

The content of a text file **testout.txt** is set with the data **Like Java, Spring, Hibernate, Android, PHP etc.**

# Java Writer

It is an [abstract](https://www.javatpoint.com/abstract-class-in-java) class for writing to character streams. The methods that a subclass must implement are write(char[], int, int), flush(), and close(). Most subclasses will override some of the methods defined here to provide higher efficiency, functionality or both.

### Fields

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Field** | **Description** |
| protected Object | lock | The object used to synchronize operations on this stream. |

### Constructor

|  |  |  |
| --- | --- | --- |
| **Modifier** | **Constructor** | **Description** |
| protected | Writer() | It creates a new character-stream writer whose critical sections will synchronize on the writer itself. |
| protected | Writer(Object lock) | It creates a new character-stream writer whose critical sections will synchronize on the given [object](https://www.javatpoint.com/object-and-class-in-java). |

### Methods

|  |  |  |
| --- | --- | --- |
| Modifier and Type | **Method** | **Description** |
| Writer | append(char c) | It appends the specified character to this writer. |
| Writer | append(CharSequence csq) | It appends the specified character sequence to this writer |
| Writer | append(CharSequence csq, int start, int end) | It appends a subsequence of the specified character sequence to this writer. |
| abstract void | close() | It closes the stream, flushing it first. |
| abstract void | flush() | It flushes the stream. |
| void | write(char[] cbuf) | It writes an [array](https://www.javatpoint.com/array-in-java) of characters. |
| abstract void | write(char[] cbuf, int off, int len) | It writes a portion of an array of characters. |
| void | write(int c) | It writes a single character. |
| void | write(String str) | It writes a [string](https://www.javatpoint.com/java-string). |
| void | write(String str, int off, int len) | It writes a portion of a string. |

## Java Writer Example

1. **import** java.io.\*;
2. **public** **class** WriterExample {
3. **public** **static** **void** main(String[] args) {
4. **try** {
5. Writer w = **new** FileWriter("output.txt");
6. String content = "I love my country";
7. w.write(content);
8. w.close();
9. System.out.println("Done");
10. } **catch** (IOException e) {
11. e.printStackTrace();
12. }
13. }
14. }

Output:

Done

output.txt:

I love my country

# Java PrintStream Class

The PrintStream class provides methods to write data to another stream. The PrintStream [class](https://www.javatpoint.com/object-and-class-in-java) automatically flushes the data so there is no need to call flush() method. Moreover, its methods don't throw IOException.

## Class declaration

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

1. **public** **class** PrintStream **extends** FilterOutputStream **implements** Closeable. Appendable

## Methods of PrintStream class

|  |  |
| --- | --- |
| **Method** | **Description** |
| void print(boolean b) | It prints the specified boolean value. |
| void print(char c) | It prints the specified char value. |
| void print(char[] c) | It prints the specified character [array](https://www.javatpoint.com/array-in-java) values. |
| void print(int i) | It prints the specified int value. |
| void print(long l) | It prints the specified long value. |
| void print(float f) | It prints the specified float value. |
| void print(double d) | It prints the specified double value. |
| void print(String s) | It prints the specified [string](https://www.javatpoint.com/java-string) value. |
| void print(Object obj) | It prints the specified object value. |
| void println(boolean b) | It prints the specified boolean value and terminates the line. |
| void println(char c) | It prints the specified char value and terminates the line. |
| void println(char[] c) | It prints the specified character array values and terminates the line. |
| void println(int i) | It prints the specified int value and terminates the line. |
| void println(long l) | It prints the specified long value and terminates the line. |
| void println(float f) | It prints the specified float value and terminates the line. |
| void println(double d) | It prints the specified double value and terminates the line. |
| void println(String s) | It prints the specified string value and terminates the line. |
| void println(Object obj) | It prints the specified object value and terminates the line. |
| void println() | It terminates the line only. |
| void printf(Object format, Object... args) | It writes the formatted string to the current stream. |
| void printf(Locale l, Object format, Object... args) | It writes the formatted string to the current stream. |
| void format(Object format, Object... args) | It writes the formatted string to the current stream using specified format. |
| void format(Locale l, Object format, Object... args) | It writes the formatted string to the current stream using specified format. |

## Example of java PrintStream class

In this example, we are simply printing integer and string value.

1. **package** com.javatpoint;
3. **import** java.io.FileOutputStream;
4. **import** java.io.PrintStream;
5. **public** **class** PrintStreamTest{
6. **public** **static** **void** main(String args[])**throws** Exception{
7. FileOutputStream fout=**new** FileOutputStream("D:\\testout.txt ");
8. PrintStream pout=**new** PrintStream(fout);
9. pout.println(2016);
10. pout.println("Hello Java");
11. pout.println("Welcome to Java");
12. pout.close();
13. fout.close();
14. System.out.println("Success?");
15. }
16. }

Output

Success...

The content of a text file **testout.txt** is set with the below data

2016

Hello Java

Welcome to Java

## Example of printf() method using java PrintStream class:

Let's see the simple example of printing integer value by format specifier using **printf()** method of **java.io.PrintStream** class.

1. **class** PrintStreamTest{
2. **public** **static** **void** main(String args[]){
3. **int** a=19;
4. System.out.printf("%d",a); //Note: out is the object of printstream
5. }
6. }

Output

19

# Java OutputStreamWriter

OutputStreamWriter is a [class](https://www.javatpoint.com/object-and-class-in-java) which is used to convert character stream to byte stream, the characters are encoded into byte using a specified charset. write() method calls the encoding converter which converts the character into bytes. The resulting bytes are then accumulated in a buffer before being written into the underlying output stream. The characters passed to write() methods are not buffered. We optimize the performance of OutputStreamWriter by using it with in a BufferedWriter so that to avoid frequent converter invocation.

### Constructor

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| OutputStreamWriter(OutputStream out) | It creates an OutputStreamWriter that uses the default character encoding. |
| OutputStreamWriter(OutputStream out, Charset cs) | It creates an OutputStreamWriter that uses the given charset. |
| OutputStreamWriter(OutputStream out, CharsetEncoder enc) | It creates an OutputStreamWriter that uses the given charset encoder. |
| OutputStreamWriter(OutputStream out, String charsetName) | It creates an OutputStreamWriter that uses the named charset. |

### Methods

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Method** | **Description** |
| void | close() | It closes the stream, flushing it first. |
| void | flush() | It flushes the stream. |
| String | getEncoding() | It returns the name of the character encoding being used by this stream. |
| void | write(char[] cbuf, int off, int len) | It writes a portion of an [array](https://www.javatpoint.com/array-in-java) of characters. |
| void | write(int c) | It writes a single character. |
| void | write(String str, int off, int len) | It writes a portion of a [string](https://www.javatpoint.com/java-string). |

## Example

1. **public** **class** OutputStreamWriterExample {
2. **public** **static** **void** main(String[] args) {
4. **try** {
5. OutputStream outputStream = **new** FileOutputStream("output.txt");
6. Writer outputStreamWriter = **new** OutputStreamWriter(outputStream);
8. outputStreamWriter.write("Hello World");
10. outputStreamWriter.close();
11. } **catch** (Exception e) {
12. e.getMessage();
13. }
14. }
15. }

Output:

output.txt file will contains text "Hello World"

# Java Reader

[Java](https://www.javatpoint.com/java-tutorial) Reader is an [abstract class](https://www.javatpoint.com/abstract-class-in-java) for reading character [streams](https://www.javatpoint.com/java-8-stream). The only methods that a subclass must implement are read(char[], int, int) and close(). Most subclasses, however, will [override](https://www.javatpoint.com/method-overriding-in-java) some of the methods to provide higher efficiency, additional functionality, or both.

Some of the implementation [class](https://www.javatpoint.com/object-class) are [BufferedReader](https://www.javatpoint.com/java-bufferedreader-class), [CharArrayReader](https://www.javatpoint.com/java-chararrayreader-class), [FilterReader](https://www.javatpoint.com/java-filterreader-class), [InputStreamReader](https://www.javatpoint.com/Input-from-keyboard-by-InputStreamReader), PipedReader, [StringReader](https://www.javatpoint.com/java-stringreader-class)

## Example

1. **import** java.io.\*;
2. **public** **class** ReaderExample {
3. **public** **static** **void** main(String[] args) {
4. **try** {
5. Reader reader = **new** FileReader("file.txt");
6. **int** data = reader.read();
7. **while** (data != -1) {
8. System.out.print((**char**) data);
9. data = reader.read();
10. }
11. reader.close();
12. } **catch** (Exception ex) {
13. System.out.println(ex.getMessage());
14. }
15. }
16. }

**file.txt:**

**I love my country**

**Output:**

**I love my country**

# FileReader Class

Java FileReader class is used to read data from the file. It returns data in byte format like [FileInputStream](https://www.javatpoint.com/java-fileinputstream-class) class.

It is character-oriented class which is used for [file](https://www.javatpoint.com/java-file-class) handling in [java](https://www.javatpoint.com/java-tutorial).

## Java FileReader class declaration

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

**public** **class** FileReader **extends** InputStreamReader

## Java FileReader Example

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

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

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

**Welcome to javaTpoint.**

**Output:**

**Welcome to javaTpoint.**

# 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:

**public** **class** BufferedReader **extends** Reader

## 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 PrintStream Class

The PrintStream class provides methods to write data to another stream. The PrintStream [class](https://www.javatpoint.com/object-and-class-in-java) automatically flushes the data so there is no need to call flush() method. Moreover, its methods don't throw IOException.

## Class declaration

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

**public** **class** PrintStream **extends** FilterOutputStream **implements** Closeable. Appendable

## Example of java PrintStream class

In this example, we are simply printing integer and string value.

1. **package** com.javatpoint;
3. **import** java.io.FileOutputStream;
4. **import** java.io.PrintStream;
5. **public** **class** PrintStreamTest{
6. **public** **static** **void** main(String args[])**throws** Exception{
7. FileOutputStream fout=**new** FileOutputStream("D:\\testout.txt ");
8. PrintStream pout=**new** PrintStream(fout);
9. pout.println(2016);
10. pout.println("Hello Java");
11. pout.println("Welcome to Java");
12. pout.close();
13. fout.close();
14. System.out.println("Success?");
15. }
16. }

**Output**

**Success...**

**The content of a text file testout.txt is set with the below data**

**2016**

**Hello Java**

**Welcome to Jav**

Java InputStreamReader

An InputStreamReader is a bridge from byte streams to character streams: It reads bytes and decodes them into characters using a specified charset. The charset that it uses may be specified by name or may be given explicitly, or the platform's default charset may be accepted.

## Example

1. **public** **class** InputStreamReaderExample {
2. **public** **static** **void** main(String[] args) {
3. **try**  {
4. InputStream stream = **new** FileInputStream("file.txt");
5. Reader reader = **new** InputStreamReader(stream);
6. **int** data = reader.read();
7. **while** (data != -1) {
8. System.out.print((**char**) data);
9. data = reader.read();
10. }
11. } **catch** (Exception e) {
12. e.printStackTrace();
13. }
14. }
15. }

Output:

I love my country

The file.txt contains text "I love my country" the InputStreamReader

reads Character by character from the file