compact1, compact2, compact3

java.io

Class PrintWriter

java.lang.Object
    java.io.Writer
        java.io.PrintWriter

All Implemented Interfaces:

`Closeable, Flushable, Appendable, AutoCloseable`

```
public class PrintWriter
extends Writer
```

Prints formatted representations of objects to a text-output stream. This class implements all of the `print` methods found in `PrintStream`. It does not contain methods for writing raw bytes, for which a program should use unencoded byte streams.

Unlike the `PrintStream` class, if automatic flushing is enabled it will be done only when one of the `println`, `printf`, or `format` methods is invoked, rather than whenever a newline character happens to be output. These methods use the platform's own notion of line separator rather than the newline character.

Methods in this class never throw I/O exceptions, although some of its constructors may. The client may inquire as to whether any errors have occurred by invoking `checkError()`.

Since:

`JDK1.1`

Field Summary

Fields

Modifier and Type	Field and Description
`protected Writer`	**`out`** The underlying character-output stream of this `PrintWriter`.

Fields inherited from class java.io.Writer

`lock`

Constructor Summary

Constructors

Constructor and Description
PrintWriter(**File** file) Creates a new PrintWriter, without automatic line flushing, with the specified file.
PrintWriter(**File** file, **String** csn) Creates a new PrintWriter, without automatic line flushing, with the specified file and charset.
PrintWriter(**OutputStream** out) Creates a new PrintWriter, without automatic line flushing, from an existing OutputStream.
PrintWriter(**OutputStream** out, boolean autoFlush) Creates a new PrintWriter from an existing OutputStream.
PrintWriter(**String** fileName) Creates a new PrintWriter, without automatic line flushing, with the specified file name.
PrintWriter(**String** fileName, **String** csn) Creates a new PrintWriter, without automatic line flushing, with the specified file name and charset.
PrintWriter(**Writer** out) Creates a new PrintWriter, without automatic line flushing.
PrintWriter(**Writer** out, boolean autoFlush) Creates a new PrintWriter.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
PrintWriter	**append**(char c) Appends the specified character to this writer.
PrintWriter	**append**(**CharSequence** csq) Appends the specified character sequence to this writer.
PrintWriter	**append**(**CharSequence** csq, int start, int end) Appends a subsequence of the specified character sequence to this writer.

boolean	**checkError**() Flushes the stream if it's not closed and checks its error state.
protected void	**clearError**() Clears the error state of this stream.
void	**close**() Closes the stream and releases any system resources associated with it.
void	**flush**() Flushes the stream.
PrintWriter	**format**(**Locale** l, **String** format, **Object**... args) Writes a formatted string to this writer using the specified format string and arguments.
PrintWriter	**format**(**String** format, **Object**... args) Writes a formatted string to this writer using the specified format string and arguments.
void	**print**(boolean b) Prints a boolean value.
void	**print**(char c) Prints a character.
void	**print**(char[] s) Prints an array of characters.
void	**print**(double d) Prints a double-precision floating-point number.
void	**print**(float f) Prints a floating-point number.
void	**print**(int i) Prints an integer.
void	**print**(long l) Prints a long integer.
void	**print**(**Object** obj) Prints an object.
void	**print**(**String** s) Prints a string.
PrintWriter	**printf**(**Locale** l, **String** format, **Object**... args) A convenience method to write a formatted string to this writer using the specified format string and arguments.

PrintWriter	**printf**(String format, Object... args) A convenience method to write a formatted string to this writer using the specified format string and arguments.
void	**println**() Terminates the current line by writing the line separator string.
void	**println**(boolean x) Prints a boolean value and then terminates the line.
void	**println**(char x) Prints a character and then terminates the line.
void	**println**(char[] x) Prints an array of characters and then terminates the line.
void	**println**(double x) Prints a double-precision floating-point number and then terminates the line.
void	**println**(float x) Prints a floating-point number and then terminates the line.
void	**println**(int x) Prints an integer and then terminates the line.
void	**println**(long x) Prints a long integer and then terminates the line.
void	**println**(Object x) Prints an Object and then terminates the line.
void	**println**(String x) Prints a String and then terminates the line.
protected void	**setError**() Indicates that an error has occurred.
void	**write**(char[] buf) Writes an array of characters.
void	**write**(char[] buf, int off, int len) Writes A Portion of an array of characters.
void	**write**(int c) Writes a single character.
void	**write**(String s) Writes a string.
void	**write**(String s, int off, int len) Writes a portion of a string.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

out

```
protected Writer out
```

The underlying character-output stream of this PrintWriter.

Since:

1.2

Constructor Detail

PrintWriter

```
public PrintWriter(Writer out)
```

Creates a new PrintWriter, without automatic line flushing.

Parameters:

out - A character-output stream

PrintWriter

```
public PrintWriter(Writer out,
                   boolean autoFlush)
```

Creates a new PrintWriter.

Parameters:

out - A character-output stream

autoFlush - A boolean; if true, the println, printf, or format methods will flush the output buffer

PrintWriter

```
public PrintWriter(OutputStream out)
```

Creates a new PrintWriter, without automatic line flushing, from an existing OutputStream. This convenience constructor creates the necessary intermediate OutputStreamWriter, which will convert characters into bytes using the default character encoding.

Parameters:

out - An output stream

See Also:

OutputStreamWriter.OutputStreamWriter(java.io.OutputStream)

PrintWriter

```
public PrintWriter(OutputStream out,
                   boolean autoFlush)
```

Creates a new PrintWriter from an existing OutputStream. This convenience constructor creates the necessary intermediate OutputStreamWriter, which will convert characters into bytes using the default character encoding.

Parameters:

out - An output stream

autoFlush - A boolean; if true, the println, printf, or format methods will flush the output buffer

See Also:

OutputStreamWriter.OutputStreamWriter(java.io.OutputStream)

PrintWriter

```
public PrintWriter(String fileName)
            throws FileNotFoundException
```

Creates a new PrintWriter, without automatic line flushing, with the specified file name. This convenience constructor creates the necessary intermediate OutputStreamWriter, which will encode characters using the default charset for this instance of the Java virtual machine.

Parameters:

fileName - The name of the file to use as the destination of this writer. If the file exists then it will be truncated to zero size; otherwise, a new file will be created. The output will be written to the file and is buffered.

Throws:

FileNotFoundException - If the given string does not denote an existing, writable regular file and a new regular file of that name cannot be created, or if some other error occurs while opening or creating the file

SecurityException - If a security manager is present and checkWrite(fileName) denies write access to the file

Since:

1.5

PrintWriter

```
public PrintWriter(String fileName,
                   String csn)
            throws FileNotFoundException,
                   UnsupportedEncodingException
```

Creates a new PrintWriter, without automatic line flushing, with the specified file name and charset. This convenience constructor creates the necessary intermediate `OutputStreamWriter`, which will encode characters using the provided charset.

Parameters:

`fileName` - The name of the file to use as the destination of this writer. If the file exists then it will be truncated to zero size; otherwise, a new file will be created. The output will be written to the file and is buffered.

`csn` - The name of a supported `charset`

Throws:

`FileNotFoundException` - If the given string does not denote an existing, writable regular file and a new regular file of that name cannot be created, or if some other error occurs while opening or creating the file

`SecurityException` - If a security manager is present and `checkWrite(fileName)` denies write access to the file

`UnsupportedEncodingException` - If the named charset is not supported

Since:

1.5

PrintWriter

```
public PrintWriter(File file)
            throws FileNotFoundException
```

Creates a new PrintWriter, without automatic line flushing, with the specified file. This convenience constructor creates the necessary intermediate `OutputStreamWriter`, which will encode characters using the default charset for this instance of the Java virtual machine.

Parameters:

`file` - The file to use as the destination of this writer. If the file exists then it will be truncated to zero size; otherwise, a new file will be created. The output will be written to the file and is buffered.

Throws:

FileNotFoundException - If the given file object does not denote an existing, writable regular file and a new regular file of that name cannot be created, or if some other error occurs while opening or creating the file

SecurityException - If a security manager is present and checkWrite(file.getPath()) denies write access to the file

Since:

1.5

PrintWriter

```
public PrintWriter(File file,
                   String csn)
            throws FileNotFoundException,
                   UnsupportedEncodingException
```

Creates a new PrintWriter, without automatic line flushing, with the specified file and charset. This convenience constructor creates the necessary intermediate OutputStreamWriter, which will encode characters using the provided charset.

Parameters:

file - The file to use as the destination of this writer. If the file exists then it will be truncated to zero size; otherwise, a new file will be created. The output will be written to the file and is buffered.

csn - The name of a supported charset

Throws:

FileNotFoundException - If the given file object does not denote an existing, writable regular file and a new regular file of that name cannot be created, or if some other error occurs while opening or creating the file

SecurityException - If a security manager is present and checkWrite(file.getPath()) denies write access to the file

UnsupportedEncodingException - If the named charset is not supported

Since:

1.5

Method Detail

flush

```
public void flush()
```

Flushes the stream.

Specified by:

`flush` in interface `Flushable`

Specified by:

`flush` in class `Writer`

See Also:

`checkError()`

close

`public void close()`

Closes the stream and releases any system resources associated with it. Closing a previously closed stream has no effect.

Specified by:

`close` in interface `Closeable`

Specified by:

`close` in interface `AutoCloseable`

Specified by:

`close` in class `Writer`

See Also:

`checkError()`

checkError

`public boolean checkError()`

Flushes the stream if it's not closed and checks its error state.

Returns:

`true if the print stream has encountered an error, either on the underlying output stream or during a format conversion.`

setError

`protected void setError()`

Indicates that an error has occurred.

This method will cause subsequent invocations of `checkError()` to return `true` until `clearError()` is invoked.

clearError

```
protected void clearError()
```

Clears the error state of this stream.

This method will cause subsequent invocations of `checkError()` to return `false` until another write operation fails and invokes `setError()`.

Since:

1.6

write

```
public void write(int c)
```

Writes a single character.

Overrides:

`write` in class `Writer`

Parameters:

`c` - int specifying a character to be written.

write

```
public void write(char[] buf,
                  int off,
                  int len)
```

Writes A Portion of an array of characters.

Specified by:

`write` in class `Writer`

Parameters:

`buf` - Array of characters

`off` - Offset from which to start writing characters

`len` - Number of characters to write

write

```
public void write(char[] buf)
```

Writes an array of characters. This method cannot be inherited from the Writer class because it must suppress I/O exceptions.

Overrides:

`write` in class `Writer`

Parameters:

buf - Array of characters to be written

write

```
public void write(String s,
                  int off,
                  int len)
```

Writes a portion of a string.

Overrides:

`write` in class `Writer`

Parameters:

s - A String

off - Offset from which to start writing characters

len - Number of characters to write

write

```
public void write(String s)
```

Writes a string. This method cannot be inherited from the Writer class because it must suppress I/O exceptions.

Overrides:

`write` in class `Writer`

Parameters:

s - String to be written

print

```
public void print(boolean b)
```

Prints a boolean value. The string produced by `String.valueOf(boolean)` is translated into bytes according to the platform's default character encoding, and these bytes are written in exactly the manner of the `write(int)` method.

Parameters:

b - The boolean to be printed

print

```
public void print(char c)
```

Prints a character. The character is translated into one or more bytes according to the platform's default character encoding, and these bytes are written in exactly the manner of the `write(int)` method.

Parameters:

`c - The char to be printed`

print

```
public void print(int i)
```

Prints an integer. The string produced by `String.valueOf(int)` is translated into bytes according to the platform's default character encoding, and these bytes are written in exactly the manner of the `write(int)` method.

Parameters:

`i - The int to be printed`

See Also:

`Integer.toString(int)`

print

```
public void print(long l)
```

Prints a long integer. The string produced by `String.valueOf(long)` is translated into bytes according to the platform's default character encoding, and these bytes are written in exactly the manner of the `write(int)` method.

Parameters:

`l - The long to be printed`

See Also:

`Long.toString(long)`

print

```
public void print(float f)
```

Prints a floating-point number. The string produced by `String.valueOf(float)` is translated into bytes according to the platform's default character encoding, and these bytes are written in exactly the manner of the `write(int)` method.

Parameters:

`f - The float to be printed`

See Also:

`Float.toString(float)`

print

```
public void print(double d)
```

Prints a double-precision floating-point number. The string produced by `String.valueOf(double)` is translated into bytes according to the platform's default character encoding, and these bytes are written in exactly the manner of the `write(int)` method.

Parameters:

`d - The double to be printed`

See Also:

`Double.toString(double)`

print

```
public void print(char[] s)
```

Prints an array of characters. The characters are converted into bytes according to the platform's default character encoding, and these bytes are written in exactly the manner of the `write(int)` method.

Parameters:

`s - The array of chars to be printed`

Throws:

`NullPointerException - If s is null`

print

```
public void print(String s)
```

Prints a string. If the argument is `null` then the string `"null"` is printed. Otherwise, the string's characters are converted into bytes according to the platform's default character encoding, and these bytes are written in exactly the manner of the `write(int)` method.

Parameters:

`s - The String to be printed`

print

```
public void print(Object obj)
```

Prints an object. The string produced by the `String.valueOf(Object)` method is translated into bytes according to the platform's default character encoding, and these bytes are written in exactly the manner of the `write(int)` method.

Parameters:

`obj - The Object to be printed`

See Also:

`Object.toString()`

println

`public void println()`

Terminates the current line by writing the line separator string. The line separator string is defined by the system property `line.separator`, and is not necessarily a single newline character (`'\n'`).

println

`public void println(boolean x)`

Prints a boolean value and then terminates the line. This method behaves as though it invokes `print(boolean)` and then `println()`.

Parameters:

`x - the boolean value to be printed`

println

`public void println(char x)`

Prints a character and then terminates the line. This method behaves as though it invokes `print(char)` and then `println()`.

Parameters:

`x - the char value to be printed`

println

`public void println(int x)`

Prints an integer and then terminates the line. This method behaves as though it invokes `print(int)` and then `println()`.

Parameters:

`x - the int value to be printed`

println

`public void println(long x)`

Prints a long integer and then terminates the line. This method behaves as though it invokes `print(long)` and then `println()`.

Parameters:

`x - the long value to be printed`

println

`public void println(float x)`

Prints a floating-point number and then terminates the line. This method behaves as though it invokes `print(float)` and then `println()`.

Parameters:

`x - the float value to be printed`

println

`public void println(double x)`

Prints a double-precision floating-point number and then terminates the line. This method behaves as though it invokes `print(double)` and then `println()`.

Parameters:

`x - the double value to be printed`

println

`public void println(char[] x)`

Prints an array of characters and then terminates the line. This method behaves as though it invokes `print(char[])` and then `println()`.

Parameters:

`x - the array of char values to be printed`

println

`public void println(String x)`

Prints a String and then terminates the line. This method behaves as though it invokes `print(String)` and then `println()`.

Parameters:

x - the String value to be printed

println

```
public void println(Object x)
```

Prints an Object and then terminates the line. This method calls at first String.valueOf(x) to get the printed object's string value, then behaves as though it invokes `print(String)` and then `println()`.

Parameters:

x - The Object to be printed.

printf

```
public PrintWriter printf(String format,
                          Object... args)
```

A convenience method to write a formatted string to this writer using the specified format string and arguments. If automatic flushing is enabled, calls to this method will flush the output buffer.

An invocation of this method of the form `out.printf(format, args)` behaves in exactly the same way as the invocation

```
out.format(format, args)
```

Parameters:

format - A format string as described in Format string syntax.

args - Arguments referenced by the format specifiers in the format string. If there are more arguments than format specifiers, the extra arguments are ignored. The number of arguments is variable and may be zero. The maximum number of arguments is limited by the maximum dimension of a Java array as defined by *The Java™ Virtual Machine Specification*. The behaviour on a null argument depends on the conversion.

Returns:

This writer

Throws:

IllegalFormatException - If a format string contains an illegal syntax, a format specifier that is incompatible with the given arguments, insufficient arguments given the format string, or other illegal conditions. For specification of all possible formatting errors, see the Details section of the formatter class specification.

NullPointerException - If the format is null

Since:

1.5

printf

```
public PrintWriter printf(Locale l,
                          String format,
                          Object... args)
```

A convenience method to write a formatted string to this writer using the specified format string and arguments. If automatic flushing is enabled, calls to this method will flush the output buffer.

An invocation of this method of the form `out.printf(l, format, args)` behaves in exactly the same way as the invocation

```
out.format(l, format, args)
```

Parameters:

`l` - The `locale` to apply during formatting. If `l` is null then no localization is applied.

`format` - A format string as described in `Format string syntax`.

`args` - Arguments referenced by the format specifiers in the format string. If there are more arguments than format specifiers, the extra arguments are ignored. The number of arguments is variable and may be zero. The maximum number of arguments is limited by the maximum dimension of a Java array as defined by *The Java™ Virtual Machine Specification*. The behaviour on a null argument depends on the `conversion`.

Returns:

This writer

Throws:

`IllegalFormatException` - If a format string contains an illegal syntax, a format specifier that is incompatible with the given arguments, insufficient arguments given the format string, or other illegal conditions. For specification of all possible formatting errors, see the `Details` section of the formatter class specification.

`NullPointerException` - If the format is null

Since:

1.5

format

```
public PrintWriter format(String format,
                          Object... args)
```

Writes a formatted string to this writer using the specified format string and arguments. If automatic flushing is enabled, calls to this method will flush the output buffer.

The locale always used is the one returned by `Locale.getDefault()`, regardless of any previous invocations of other formatting methods on this object.

Parameters:

`format` - A format string as described in Format string syntax.

`args` - Arguments referenced by the format specifiers in the format string. If there are more arguments than format specifiers, the extra arguments are ignored. The number of arguments is variable and may be zero. The maximum number of arguments is limited by the maximum dimension of a Java array as defined by *The Java™ Virtual Machine Specification*. The behaviour on a null argument depends on the conversion.

Returns:

This writer

Throws:

`IllegalFormatException` - If a format string contains an illegal syntax, a format specifier that is incompatible with the given arguments, insufficient arguments given the format string, or other illegal conditions. For specification of all possible formatting errors, see the Details section of the Formatter class specification.

`NullPointerException` - If the format is null

Since:

1.5

format

```
public PrintWriter format(Locale l,
                          String format,
                          Object... args)
```

Writes a formatted string to this writer using the specified format string and arguments. If automatic flushing is enabled, calls to this method will flush the output buffer.

Parameters:

`l` - The locale to apply during formatting. If l is null then no localization is applied.

`format` - A format string as described in Format string syntax.

`args` - Arguments referenced by the format specifiers in the format string. If there are more arguments than format specifiers, the extra arguments are ignored. The number of arguments is variable and may be zero. The maximum number of arguments is limited by the maximum dimension of a Java array as defined by *The Java™ Virtual Machine Specification*. The behaviour on a null argument depends on the conversion.

Returns:

This writer

Throws:

IllegalFormatException - If a format string contains an illegal syntax, a format specifier that is incompatible with the given arguments, insufficient arguments given the format string, or other illegal conditions. For specification of all possible formatting errors, see the Details section of the formatter class specification.

NullPointerException - If the format is null

Since:

1.5

append

```
public PrintWriter append(CharSequence csq)
```

Appends the specified character sequence to this writer.

An invocation of this method of the form `out.append(csq)` behaves in exactly the same way as the invocation

```
out.write(csq.toString())
```

Depending on the specification of `toString` for the character sequence `csq`, the entire sequence may not be appended. For instance, invoking the `toString` method of a character buffer will return a subsequence whose content depends upon the buffer's position and limit.

Specified by:

append in interface Appendable

Overrides:

append in class Writer

Parameters:

csq - The character sequence to append. If csq is null, then the four characters "null" are appended to this writer.

Returns:

This writer

Since:

1.5

append

```
public PrintWriter append(CharSequence csq,
                          int start,
```

int end)

Appends a subsequence of the specified character sequence to this writer.

An invocation of this method of the form `out.append(csq, start, end)` when `csq` is not `null`, behaves in exactly the same way as the invocation

```
out.write(csq.subSequence(start, end).toString())
```

Specified by:

`append` in interface `Appendable`

Overrides:

`append` in class `Writer`

Parameters:

`csq` - The character sequence from which a subsequence will be appended. If csq is null, then characters will be appended as if csq contained the four characters "null".

`start` - The index of the first character in the subsequence

`end` - The index of the character following the last character in the subsequence

Returns:

This writer

Throws:

`IndexOutOfBoundsException` - If start or end are negative, start is greater than end, or end is greater than csq.length()

Since:

1.5

append

`public PrintWriter append(char c)`

Appends the specified character to this writer.

An invocation of this method of the form `out.append(c)` behaves in exactly the same way as the invocation

```
out.write(c)
```

Specified by:

`append` in interface `Appendable`

Overrides:

`append` in class `Writer`

Parameters:

`c` - The 16-bit character to append

Returns:

This writer

Since:

1.5