| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/MessageFormat.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/text/Format.Field.html)   [**NEXT CLASS**](http://docs.google.com/java/text/MessageFormat.Field.html) | [**FRAMES**](http://docs.google.com/index.html?java/text/MessageFormat.html)    [**NO FRAMES**](http://docs.google.com/MessageFormat.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#tyjcwt) | FIELD | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

## **java.text**

Class MessageFormat

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.text.Format](http://docs.google.com/java/text/Format.html)  
 **java.text.MessageFormat**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html)

public class **MessageFormat**extends [Format](http://docs.google.com/java/text/Format.html)

MessageFormat provides a means to produce concatenated messages in a language-neutral way. Use this to construct messages displayed for end users.

MessageFormat takes a set of objects, formats them, then inserts the formatted strings into the pattern at the appropriate places.

**Note:** MessageFormat differs from the other Format classes in that you create a MessageFormat object with one of its constructors (not with a getInstance style factory method). The factory methods aren't necessary because MessageFormat itself doesn't implement locale specific behavior. Any locale specific behavior is defined by the pattern that you provide as well as the subformats used for inserted arguments.

#### Patterns and Their Interpretation

MessageFormat uses patterns of the following form:

*MessageFormatPattern:*  
 *String*  
 *MessageFormatPattern* *FormatElement* *String*  
  
 *FormatElement:*  
 { *ArgumentIndex* }  
 { *ArgumentIndex* , *FormatType* }  
 { *ArgumentIndex* , *FormatType* , *FormatStyle* }  
  
 *FormatType: one of*   
 number date time choice  
  
 *FormatStyle:*  
 short  
 medium  
 long  
 full  
 integer  
 currency  
 percent  
 *SubformatPattern*  
  
 *String:*  
 *StringPartopt*  
 *String* *StringPart*  
  
 *StringPart:*  
 ''  
 ' *QuotedString* '  
 *UnquotedString*  
  
 *SubformatPattern:*  
 *SubformatPatternPartopt*  
 *SubformatPattern* *SubformatPatternPart*  
  
 *SubFormatPatternPart:*  
 ' *QuotedPattern* '  
 *UnquotedPattern*

Within a *String*, "''" represents a single quote. A *QuotedString* can contain arbitrary characters except single quotes; the surrounding single quotes are removed. An *UnquotedString* can contain arbitrary characters except single quotes and left curly brackets. Thus, a string that should result in the formatted message "'{0}'" can be written as "'''{'0}''" or "'''{0}'''".

Within a *SubformatPattern*, different rules apply. A *QuotedPattern* can contain arbitrary characters except single quotes; but the surrounding single quotes are **not** removed, so they may be interpreted by the subformat. For example, "{1,number,$'#',##}" will produce a number format with the pound-sign quoted, with a result such as: "$#31,45". An *UnquotedPattern* can contain arbitrary characters except single quotes, but curly braces within it must be balanced. For example, "ab {0} de" and "ab '}' de" are valid subformat patterns, but "ab {0'}' de" and "ab } de" are not.

**Warning:**The rules for using quotes within message format patterns unfortunately have shown to be somewhat confusing. In particular, it isn't always obvious to localizers whether single quotes need to be doubled or not. Make sure to inform localizers about the rules, and tell them (for example, by using comments in resource bundle source files) which strings will be processed by MessageFormat. Note that localizers may need to use single quotes in translated strings where the original version doesn't have them.

The *ArgumentIndex* value is a non-negative integer written using the digits '0' through '9', and represents an index into the arguments array passed to the format methods or the result array returned by the parse methods.

The *FormatType* and *FormatStyle* values are used to create a Format instance for the format element. The following table shows how the values map to Format instances. Combinations not shown in the table are illegal. A *SubformatPattern* must be a valid pattern string for the Format subclass used.

| Format Type | Format Style | Subformat Created |
| --- | --- | --- |
| *(none)* | *(none)* | null |
| number | *(none)* | NumberFormat.getInstance(getLocale()) |
| integer | NumberFormat.getIntegerInstance(getLocale()) |
| currency | NumberFormat.getCurrencyInstance(getLocale()) |
| percent | NumberFormat.getPercentInstance(getLocale()) |
| *SubformatPattern* | new DecimalFormat(subformatPattern, DecimalFormatSymbols.getInstance(getLocale())) |
| date | *(none)* | DateFormat.getDateInstance(DateFormat.DEFAULT, getLocale()) |
| short | DateFormat.getDateInstance(DateFormat.SHORT, getLocale()) |
| medium | DateFormat.getDateInstance(DateFormat.DEFAULT, getLocale()) |
| long | DateFormat.getDateInstance(DateFormat.LONG, getLocale()) |
| full | DateFormat.getDateInstance(DateFormat.FULL, getLocale()) |
| *SubformatPattern* | new SimpleDateFormat(subformatPattern, getLocale()) |
| time | *(none)* | DateFormat.getTimeInstance(DateFormat.DEFAULT, getLocale()) |
| short | DateFormat.getTimeInstance(DateFormat.SHORT, getLocale()) |
| medium | DateFormat.getTimeInstance(DateFormat.DEFAULT, getLocale()) |
| long | DateFormat.getTimeInstance(DateFormat.LONG, getLocale()) |
| full | DateFormat.getTimeInstance(DateFormat.FULL, getLocale()) |
| *SubformatPattern* | new SimpleDateFormat(subformatPattern, getLocale()) |
| choice | *SubformatPattern* | new ChoiceFormat(subformatPattern) |

#### Usage Information

Here are some examples of usage. In real internationalized programs, the message format pattern and other static strings will, of course, be obtained from resource bundles. Other parameters will be dynamically determined at runtime.

The first example uses the static method MessageFormat.format, which internally creates a MessageFormat for one-time use:

int planet = 7;  
 String event = "a disturbance in the Force";  
  
 String result = MessageFormat.format(  
 "At {1,time} on {1,date}, there was {2} on planet {0,number,integer}.",  
 planet, new Date(), event);

The output is:

At 12:30 PM on Jul 3, 2053, there was a disturbance in the Force on planet 7.

The following example creates a MessageFormat instance that can be used repeatedly:

int fileCount = 1273;  
 String diskName = "MyDisk";  
 Object[] testArgs = {new Long(fileCount), diskName};  
  
 MessageFormat form = new MessageFormat(  
 "The disk \"{1}\" contains {0} file(s).");  
  
 System.out.println(form.format(testArgs));

The output with different values for fileCount:

The disk "MyDisk" contains 0 file(s).  
 The disk "MyDisk" contains 1 file(s).  
 The disk "MyDisk" contains 1,273 file(s).

For more sophisticated patterns, you can use a ChoiceFormat to produce correct forms for singular and plural:

MessageFormat form = new MessageFormat("The disk \"{1}\" contains {0}.");  
 double[] filelimits = {0,1,2};  
 String[] filepart = {"no files","one file","{0,number} files"};  
 ChoiceFormat fileform = new ChoiceFormat(filelimits, filepart);  
 form.setFormatByArgumentIndex(0, fileform);  
  
 int fileCount = 1273;  
 String diskName = "MyDisk";  
 Object[] testArgs = {new Long(fileCount), diskName};  
  
 System.out.println(form.format(testArgs));

The output with different values for fileCount:

The disk "MyDisk" contains no files.  
 The disk "MyDisk" contains one file.  
 The disk "MyDisk" contains 1,273 files.

You can create the ChoiceFormat programmatically, as in the above example, or by using a pattern. See [ChoiceFormat](http://docs.google.com/java/text/ChoiceFormat.html) for more information.

form.applyPattern(  
 "There {0,choice,0#are no files|1#is one file|1<are {0,number,integer} files}.");

**Note:** As we see above, the string produced by a ChoiceFormat in MessageFormat is treated as special; occurrences of '{' are used to indicate subformats, and cause recursion. If you create both a MessageFormat and ChoiceFormat programmatically (instead of using the string patterns), then be careful not to produce a format that recurses on itself, which will cause an infinite loop.

When a single argument is parsed more than once in the string, the last match will be the final result of the parsing. For example,

MessageFormat mf = new MessageFormat("{0,number,#.##}, {0,number,#.#}");  
 Object[] objs = {new Double(3.1415)};  
 String result = mf.format( objs );  
 // result now equals "3.14, 3.1"  
 objs = null;  
 objs = mf.parse(result, new ParsePosition(0));  
 // objs now equals {new Double(3.1)}

Likewise, parsing with a MessageFormat object using patterns containing multiple occurrences of the same argument would return the last match. For example,

MessageFormat mf = new MessageFormat("{0}, {0}, {0}");  
 String forParsing = "x, y, z";  
 Object[] objs = mf.parse(forParsing, new ParsePosition(0));  
 // result now equals {new String("z")}

#### Synchronization

Message formats are not synchronized. It is recommended to create separate format instances for each thread. If multiple threads access a format concurrently, it must be synchronized externally.

**See Also:**[Locale](http://docs.google.com/java/util/Locale.html), [Format](http://docs.google.com/java/text/Format.html), [NumberFormat](http://docs.google.com/java/text/NumberFormat.html), [DecimalFormat](http://docs.google.com/java/text/DecimalFormat.html), [ChoiceFormat](http://docs.google.com/java/text/ChoiceFormat.html), [Serialized Form](http://docs.google.com/serialized-form.html#java.text.MessageFormat)

| **Nested Class Summary** | |
| --- | --- |
| static class | [**MessageFormat.Field**](http://docs.google.com/java/text/MessageFormat.Field.html)            Defines constants that are used as attribute keys in the AttributedCharacterIterator returned from MessageFormat.formatToCharacterIterator. |

| **Constructor Summary** | |
| --- | --- |
| [**MessageFormat**](http://docs.google.com/java/text/MessageFormat.html#MessageFormat(java.lang.String))([String](http://docs.google.com/java/lang/String.html) pattern)            Constructs a MessageFormat for the default locale and the specified pattern. |
| [**MessageFormat**](http://docs.google.com/java/text/MessageFormat.html#MessageFormat(java.lang.String,%20java.util.Locale))([String](http://docs.google.com/java/lang/String.html) pattern, [Locale](http://docs.google.com/java/util/Locale.html) locale)            Constructs a MessageFormat for the specified locale and pattern. |

| **Method Summary** | |
| --- | --- |
| void | [**applyPattern**](http://docs.google.com/java/text/MessageFormat.html#applyPattern(java.lang.String))([String](http://docs.google.com/java/lang/String.html) pattern)            Sets the pattern used by this message format. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**clone**](http://docs.google.com/java/text/MessageFormat.html#clone())()            Creates and returns a copy of this object. |
| boolean | [**equals**](http://docs.google.com/java/text/MessageFormat.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Equality comparison between two message format objects |
| [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) | [**format**](http://docs.google.com/java/text/MessageFormat.html#format(java.lang.Object%5B%5D,%20java.lang.StringBuffer,%20java.text.FieldPosition))([Object](http://docs.google.com/java/lang/Object.html)[] arguments, [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) result, [FieldPosition](http://docs.google.com/java/text/FieldPosition.html) pos)            Formats an array of objects and appends the MessageFormat's pattern, with format elements replaced by the formatted objects, to the provided StringBuffer. |
| [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) | [**format**](http://docs.google.com/java/text/MessageFormat.html#format(java.lang.Object,%20java.lang.StringBuffer,%20java.text.FieldPosition))([Object](http://docs.google.com/java/lang/Object.html) arguments, [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) result, [FieldPosition](http://docs.google.com/java/text/FieldPosition.html) pos)            Formats an array of objects and appends the MessageFormat's pattern, with format elements replaced by the formatted objects, to the provided StringBuffer. |
| static [String](http://docs.google.com/java/lang/String.html) | [**format**](http://docs.google.com/java/text/MessageFormat.html#format(java.lang.String,%20java.lang.Object...))([String](http://docs.google.com/java/lang/String.html) pattern, [Object](http://docs.google.com/java/lang/Object.html)... arguments)            Creates a MessageFormat with the given pattern and uses it to format the given arguments. |
| [AttributedCharacterIterator](http://docs.google.com/java/text/AttributedCharacterIterator.html) | [**formatToCharacterIterator**](http://docs.google.com/java/text/MessageFormat.html#formatToCharacterIterator(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) arguments)            Formats an array of objects and inserts them into the MessageFormat's pattern, producing an AttributedCharacterIterator. |
| [Format](http://docs.google.com/java/text/Format.html)[] | [**getFormats**](http://docs.google.com/java/text/MessageFormat.html#getFormats())()            Gets the formats used for the format elements in the previously set pattern string. |
| [Format](http://docs.google.com/java/text/Format.html)[] | [**getFormatsByArgumentIndex**](http://docs.google.com/java/text/MessageFormat.html#getFormatsByArgumentIndex())()            Gets the formats used for the values passed into format methods or returned from parse methods. |
| [Locale](http://docs.google.com/java/util/Locale.html) | [**getLocale**](http://docs.google.com/java/text/MessageFormat.html#getLocale())()            Gets the locale that's used when creating or comparing subformats. |
| int | [**hashCode**](http://docs.google.com/java/text/MessageFormat.html#hashCode())()            Generates a hash code for the message format object. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**parse**](http://docs.google.com/java/text/MessageFormat.html#parse(java.lang.String))([String](http://docs.google.com/java/lang/String.html) source)            Parses text from the beginning of the given string to produce an object array. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**parse**](http://docs.google.com/java/text/MessageFormat.html#parse(java.lang.String,%20java.text.ParsePosition))([String](http://docs.google.com/java/lang/String.html) source, [ParsePosition](http://docs.google.com/java/text/ParsePosition.html) pos)            Parses the string. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**parseObject**](http://docs.google.com/java/text/MessageFormat.html#parseObject(java.lang.String,%20java.text.ParsePosition))([String](http://docs.google.com/java/lang/String.html) source, [ParsePosition](http://docs.google.com/java/text/ParsePosition.html) pos)            Parses text from a string to produce an object array. |
| void | [**setFormat**](http://docs.google.com/java/text/MessageFormat.html#setFormat(int,%20java.text.Format))(int formatElementIndex, [Format](http://docs.google.com/java/text/Format.html) newFormat)            Sets the format to use for the format element with the given format element index within the previously set pattern string. |
| void | [**setFormatByArgumentIndex**](http://docs.google.com/java/text/MessageFormat.html#setFormatByArgumentIndex(int,%20java.text.Format))(int argumentIndex, [Format](http://docs.google.com/java/text/Format.html) newFormat)            Sets the format to use for the format elements within the previously set pattern string that use the given argument index. |
| void | [**setFormats**](http://docs.google.com/java/text/MessageFormat.html#setFormats(java.text.Format%5B%5D))([Format](http://docs.google.com/java/text/Format.html)[] newFormats)            Sets the formats to use for the format elements in the previously set pattern string. |
| void | [**setFormatsByArgumentIndex**](http://docs.google.com/java/text/MessageFormat.html#setFormatsByArgumentIndex(java.text.Format%5B%5D))([Format](http://docs.google.com/java/text/Format.html)[] newFormats)            Sets the formats to use for the values passed into format methods or returned from parse methods. |
| void | [**setLocale**](http://docs.google.com/java/text/MessageFormat.html#setLocale(java.util.Locale))([Locale](http://docs.google.com/java/util/Locale.html) locale)            Sets the locale to be used when creating or comparing subformats. |
| [String](http://docs.google.com/java/lang/String.html) | [**toPattern**](http://docs.google.com/java/text/MessageFormat.html#toPattern())()            Returns a pattern representing the current state of the message format. |

| **Methods inherited from class java.text.**[**Format**](http://docs.google.com/java/text/Format.html) |
| --- |
| [format](http://docs.google.com/java/text/Format.html#format(java.lang.Object)), [parseObject](http://docs.google.com/java/text/Format.html#parseObject(java.lang.String)) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [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)) |

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

### MessageFormat

public **MessageFormat**([String](http://docs.google.com/java/lang/String.html) pattern)

Constructs a MessageFormat for the default locale and the specified pattern. The constructor first sets the locale, then parses the pattern and creates a list of subformats for the format elements contained in it. Patterns and their interpretation are specified in the [class description](#_3znysh7).

**Parameters:**pattern - the pattern for this message format **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the pattern is invalid

### MessageFormat

public **MessageFormat**([String](http://docs.google.com/java/lang/String.html) pattern,  
 [Locale](http://docs.google.com/java/util/Locale.html) locale)

Constructs a MessageFormat for the specified locale and pattern. The constructor first sets the locale, then parses the pattern and creates a list of subformats for the format elements contained in it. Patterns and their interpretation are specified in the [class description](#_3znysh7).

**Parameters:**pattern - the pattern for this message formatlocale - the locale for this message format **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the pattern is invalid**Since:** 1.4

| **Method Detail** |
| --- |

### setLocale

public void **setLocale**([Locale](http://docs.google.com/java/util/Locale.html) locale)

Sets the locale to be used when creating or comparing subformats. This affects subsequent calls

* to the [applyPattern](http://docs.google.com/java/text/MessageFormat.html#applyPattern(java.lang.String)) and [toPattern](http://docs.google.com/java/text/MessageFormat.html#toPattern()) methods if format elements specify a format type and therefore have the subformats created in the applyPattern method, as well as
* to the format and [formatToCharacterIterator](http://docs.google.com/java/text/MessageFormat.html#formatToCharacterIterator(java.lang.Object)) methods if format elements do not specify a format type and therefore have the subformats created in the formatting methods.

Subformats that have already been created are not affected.

**Parameters:**locale - the locale to be used when creating or comparing subformats

### getLocale

public [Locale](http://docs.google.com/java/util/Locale.html) **getLocale**()

Gets the locale that's used when creating or comparing subformats.

**Returns:**the locale used when creating or comparing subformats

### applyPattern

public void **applyPattern**([String](http://docs.google.com/java/lang/String.html) pattern)

Sets the pattern used by this message format. The method parses the pattern and creates a list of subformats for the format elements contained in it. Patterns and their interpretation are specified in the [class description](#_3znysh7).

**Parameters:**pattern - the pattern for this message format **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the pattern is invalid

### toPattern

public [String](http://docs.google.com/java/lang/String.html) **toPattern**()

Returns a pattern representing the current state of the message format. The string is constructed from internal information and therefore does not necessarily equal the previously applied pattern.

**Returns:**a pattern representing the current state of the message format

### setFormatsByArgumentIndex

public void **setFormatsByArgumentIndex**([Format](http://docs.google.com/java/text/Format.html)[] newFormats)

Sets the formats to use for the values passed into format methods or returned from parse methods. The indices of elements in newFormats correspond to the argument indices used in the previously set pattern string. The order of formats in newFormats thus corresponds to the order of elements in the arguments array passed to the format methods or the result array returned by the parse methods.

If an argument index is used for more than one format element in the pattern string, then the corresponding new format is used for all such format elements. If an argument index is not used for any format element in the pattern string, then the corresponding new format is ignored. If fewer formats are provided than needed, then only the formats for argument indices less than newFormats.length are replaced.

**Parameters:**newFormats - the new formats to use **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if newFormats is null**Since:** 1.4

### setFormats

public void **setFormats**([Format](http://docs.google.com/java/text/Format.html)[] newFormats)

Sets the formats to use for the format elements in the previously set pattern string. The order of formats in newFormats corresponds to the order of format elements in the pattern string.

If more formats are provided than needed by the pattern string, the remaining ones are ignored. If fewer formats are provided than needed, then only the first newFormats.length formats are replaced.

Since the order of format elements in a pattern string often changes during localization, it is generally better to use the [setFormatsByArgumentIndex](http://docs.google.com/java/text/MessageFormat.html#setFormatsByArgumentIndex(java.text.Format%5B%5D)) method, which assumes an order of formats corresponding to the order of elements in the arguments array passed to the format methods or the result array returned by the parse methods.

**Parameters:**newFormats - the new formats to use **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if newFormats is null

### setFormatByArgumentIndex

public void **setFormatByArgumentIndex**(int argumentIndex,  
 [Format](http://docs.google.com/java/text/Format.html) newFormat)

Sets the format to use for the format elements within the previously set pattern string that use the given argument index. The argument index is part of the format element definition and represents an index into the arguments array passed to the format methods or the result array returned by the parse methods.

If the argument index is used for more than one format element in the pattern string, then the new format is used for all such format elements. If the argument index is not used for any format element in the pattern string, then the new format is ignored.

**Parameters:**argumentIndex - the argument index for which to use the new formatnewFormat - the new format to use**Since:** 1.4

### setFormat

public void **setFormat**(int formatElementIndex,  
 [Format](http://docs.google.com/java/text/Format.html) newFormat)

Sets the format to use for the format element with the given format element index within the previously set pattern string. The format element index is the zero-based number of the format element counting from the start of the pattern string.

Since the order of format elements in a pattern string often changes during localization, it is generally better to use the [setFormatByArgumentIndex](http://docs.google.com/java/text/MessageFormat.html#setFormatByArgumentIndex(int,%20java.text.Format)) method, which accesses format elements based on the argument index they specify.

**Parameters:**formatElementIndex - the index of a format element within the patternnewFormat - the format to use for the specified format element **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if formatElementIndex is equal to or larger than the number of format elements in the pattern string

### getFormatsByArgumentIndex

public [Format](http://docs.google.com/java/text/Format.html)[] **getFormatsByArgumentIndex**()

Gets the formats used for the values passed into format methods or returned from parse methods. The indices of elements in the returned array correspond to the argument indices used in the previously set pattern string. The order of formats in the returned array thus corresponds to the order of elements in the arguments array passed to the format methods or the result array returned by the parse methods.

If an argument index is used for more than one format element in the pattern string, then the format used for the last such format element is returned in the array. If an argument index is not used for any format element in the pattern string, then null is returned in the array.

**Returns:**the formats used for the arguments within the pattern**Since:** 1.4

### getFormats

public [Format](http://docs.google.com/java/text/Format.html)[] **getFormats**()

Gets the formats used for the format elements in the previously set pattern string. The order of formats in the returned array corresponds to the order of format elements in the pattern string.

Since the order of format elements in a pattern string often changes during localization, it's generally better to use the [getFormatsByArgumentIndex](http://docs.google.com/java/text/MessageFormat.html#getFormatsByArgumentIndex()) method, which assumes an order of formats corresponding to the order of elements in the arguments array passed to the format methods or the result array returned by the parse methods.

**Returns:**the formats used for the format elements in the pattern

### format

public final [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) **format**([Object](http://docs.google.com/java/lang/Object.html)[] arguments,  
 [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) result,  
 [FieldPosition](http://docs.google.com/java/text/FieldPosition.html) pos)

Formats an array of objects and appends the MessageFormat's pattern, with format elements replaced by the formatted objects, to the provided StringBuffer.

The text substituted for the individual format elements is derived from the current subformat of the format element and the arguments element at the format element's argument index as indicated by the first matching line of the following table. An argument is *unavailable* if arguments is null or has fewer than argumentIndex+1 elements.

| Subformat | Argument | Formatted Text |
| --- | --- | --- |
| *any* | *unavailable* | "{" + argumentIndex + "}" |
| *any* | null | "null" |
| instanceof ChoiceFormat | *any* | subformat.format(argument).indexOf('{') >= 0 ?  (new MessageFormat(subformat.format(argument), getLocale())).format(argument) : subformat.format(argument) |
| != null | *any* | subformat.format(argument) |
| null | instanceof Number | NumberFormat.getInstance(getLocale()).format(argument) |
| null | instanceof Date | DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.SHORT, getLocale()).format(argument) |
| null | instanceof String | argument |
| null | *any* | argument.toString() |

If pos is non-null, and refers to Field.ARGUMENT, the location of the first formatted string will be returned.

**Parameters:**arguments - an array of objects to be formatted and substituted.result - where text is appended.pos - On input: an alignment field, if desired. On output: the offsets of the alignment field. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if an argument in the arguments array is not of the type expected by the format element(s) that use it.

### format

public static [String](http://docs.google.com/java/lang/String.html) **format**([String](http://docs.google.com/java/lang/String.html) pattern,  
 [Object](http://docs.google.com/java/lang/Object.html)... arguments)

Creates a MessageFormat with the given pattern and uses it to format the given arguments. This is equivalent to(new [MessageFormat](http://docs.google.com/java/text/MessageFormat.html#MessageFormat(java.lang.String))(pattern)).[format](http://docs.google.com/java/text/MessageFormat.html#format(java.lang.Object%5B%5D,%20java.lang.StringBuffer,%20java.text.FieldPosition))(arguments, new StringBuffer(), null).toString()

**Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the pattern is invalid, or if an argument in the arguments array is not of the type expected by the format element(s) that use it.

### format

public final [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) **format**([Object](http://docs.google.com/java/lang/Object.html) arguments,  
 [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) result,  
 [FieldPosition](http://docs.google.com/java/text/FieldPosition.html) pos)

Formats an array of objects and appends the MessageFormat's pattern, with format elements replaced by the formatted objects, to the provided StringBuffer. This is equivalent to[format](http://docs.google.com/java/text/MessageFormat.html#format(java.lang.Object%5B%5D,%20java.lang.StringBuffer,%20java.text.FieldPosition))((Object[]) arguments, result, pos)

**Specified by:**[format](http://docs.google.com/java/text/Format.html#format(java.lang.Object,%20java.lang.StringBuffer,%20java.text.FieldPosition)) in class [Format](http://docs.google.com/java/text/Format.html) **Parameters:**arguments - an array of objects to be formatted and substituted.result - where text is appended.pos - On input: an alignment field, if desired. On output: the offsets of the alignment field. **Returns:**the string buffer passed in as toAppendTo, with formatted text appended **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if an argument in the arguments array is not of the type expected by the format element(s) that use it.

### formatToCharacterIterator

public [AttributedCharacterIterator](http://docs.google.com/java/text/AttributedCharacterIterator.html) **formatToCharacterIterator**([Object](http://docs.google.com/java/lang/Object.html) arguments)

Formats an array of objects and inserts them into the MessageFormat's pattern, producing an AttributedCharacterIterator. You can use the returned AttributedCharacterIterator to build the resulting String, as well as to determine information about the resulting String.

The text of the returned AttributedCharacterIterator is the same that would be returned by

[format](http://docs.google.com/java/text/MessageFormat.html#format(java.lang.Object%5B%5D,%20java.lang.StringBuffer,%20java.text.FieldPosition))(arguments, new StringBuffer(), null).toString()

In addition, the AttributedCharacterIterator contains at least attributes indicating where text was generated from an argument in the arguments array. The keys of these attributes are of type MessageFormat.Field, their values are Integer objects indicating the index in the arguments array of the argument from which the text was generated.

The attributes/value from the underlying Format instances that MessageFormat uses will also be placed in the resulting AttributedCharacterIterator. This allows you to not only find where an argument is placed in the resulting String, but also which fields it contains in turn.

**Overrides:**[formatToCharacterIterator](http://docs.google.com/java/text/Format.html#formatToCharacterIterator(java.lang.Object)) in class [Format](http://docs.google.com/java/text/Format.html) **Parameters:**arguments - an array of objects to be formatted and substituted. **Returns:**AttributedCharacterIterator describing the formatted value. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if arguments is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if an argument in the arguments array is not of the type expected by the format element(s) that use it.**Since:** 1.4

### parse

public [Object](http://docs.google.com/java/lang/Object.html)[] **parse**([String](http://docs.google.com/java/lang/String.html) source,  
 [ParsePosition](http://docs.google.com/java/text/ParsePosition.html) pos)

Parses the string.

Caveats: The parse may fail in a number of circumstances. For example:

* If one of the arguments does not occur in the pattern.
* If the format of an argument loses information, such as with a choice format where a large number formats to "many".
* Does not yet handle recursion (where the substituted strings contain {n} references.)
* Will not always find a match (or the correct match) if some part of the parse is ambiguous. For example, if the pattern "{1},{2}" is used with the string arguments {"a,b", "c"}, it will format as "a,b,c". When the result is parsed, it will return {"a", "b,c"}.
* If a single argument is parsed more than once in the string, then the later parse wins.

When the parse fails, use ParsePosition.getErrorIndex() to find out where in the string the parsing failed. The returned error index is the starting offset of the sub-patterns that the string is comparing with. For example, if the parsing string "AAA {0} BBB" is comparing against the pattern "AAD {0} BBB", the error index is 0. When an error occurs, the call to this method will return null. If the source is null, return an empty array.

### parse

public [Object](http://docs.google.com/java/lang/Object.html)[] **parse**([String](http://docs.google.com/java/lang/String.html) source)  
 throws [ParseException](http://docs.google.com/java/text/ParseException.html)

Parses text from the beginning of the given string to produce an object array. The method may not use the entire text of the given string.

See the [parse(String, ParsePosition)](http://docs.google.com/java/text/MessageFormat.html#parse(java.lang.String,%20java.text.ParsePosition)) method for more information on message parsing.

**Parameters:**source - A String whose beginning should be parsed. **Returns:**An Object array parsed from the string. **Throws:** [ParseException](http://docs.google.com/java/text/ParseException.html) - if the beginning of the specified string cannot be parsed.

### parseObject

public [Object](http://docs.google.com/java/lang/Object.html) **parseObject**([String](http://docs.google.com/java/lang/String.html) source,  
 [ParsePosition](http://docs.google.com/java/text/ParsePosition.html) pos)

Parses text from a string to produce an object array.

The method attempts to parse text starting at the index given by pos. If parsing succeeds, then the index of pos is updated to the index after the last character used (parsing does not necessarily use all characters up to the end of the string), and the parsed object array is returned. The updated pos can be used to indicate the starting point for the next call to this method. If an error occurs, then the index of pos is not changed, the error index of pos is set to the index of the character where the error occurred, and null is returned.

See the [parse(String, ParsePosition)](http://docs.google.com/java/text/MessageFormat.html#parse(java.lang.String,%20java.text.ParsePosition)) method for more information on message parsing.

**Specified by:**[parseObject](http://docs.google.com/java/text/Format.html#parseObject(java.lang.String,%20java.text.ParsePosition)) in class [Format](http://docs.google.com/java/text/Format.html) **Parameters:**source - A String, part of which should be parsed.pos - A ParsePosition object with index and error index information as described above. **Returns:**An Object array parsed from the string. In case of error, returns null. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if pos is null.

### clone

public [Object](http://docs.google.com/java/lang/Object.html) **clone**()

Creates and returns a copy of this object.

**Overrides:**[clone](http://docs.google.com/java/text/Format.html#clone()) in class [Format](http://docs.google.com/java/text/Format.html) **Returns:**a clone of this instance.**See Also:**[Cloneable](http://docs.google.com/java/lang/Cloneable.html)

### equals

public boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) obj)

Equality comparison between two message format objects

**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**obj - the reference object with which to compare. **Returns:**true if this object is the same as the obj argument; false otherwise.**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### hashCode

public int **hashCode**()

Generates a hash code for the message format object.

**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a hash code value for this object.**See Also:**[Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/MessageFormat.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/text/Format.Field.html)   [**NEXT CLASS**](http://docs.google.com/java/text/MessageFormat.Field.html) | [**FRAMES**](http://docs.google.com/index.html?java/text/MessageFormat.html)    [**NO FRAMES**](http://docs.google.com/MessageFormat.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#tyjcwt) | FIELD | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

[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).