| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ChoiceFormat.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/CharacterIterator.html)   [**NEXT CLASS**](http://docs.google.com/java/text/CollationElementIterator.html) | [**FRAMES**](http://docs.google.com/index.html?java/text/ChoiceFormat.html)    [**NO FRAMES**](http://docs.google.com/ChoiceFormat.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#tyjcwt) | [FIELD](#1t3h5sf) | [CONSTR](#4d34og8) | [METHOD](#2s8eyo1) | DETAIL: FIELD | [CONSTR](#lnxbz9) | [METHOD](#44sinio) |

## **java.text**

Class ChoiceFormat

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

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

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

A ChoiceFormat allows you to attach a format to a range of numbers. It is generally used in a MessageFormat for handling plurals. The choice is specified with an ascending list of doubles, where each item specifies a half-open interval up to the next item:

X matches j if and only if limit[j] <= X < limit[j+1]

If there is no match, then either the first or last index is used, depending on whether the number (X) is too low or too high. If the limit array is not in ascending order, the results of formatting will be incorrect. ChoiceFormat also accepts \u221E as equivalent to infinity(INF).

**Note:** ChoiceFormat differs from the other Format classes in that you create a ChoiceFormat object with a constructor (not with a getInstance style factory method). The factory methods aren't necessary because ChoiceFormat doesn't require any complex setup for a given locale. In fact, ChoiceFormat doesn't implement any locale specific behavior.

When creating a ChoiceFormat, you must specify an array of formats and an array of limits. The length of these arrays must be the same. For example,

* *limits* = {1,2,3,4,5,6,7}  
  *formats* = {"Sun","Mon","Tue","Wed","Thur","Fri","Sat"}
* *limits* = {0, 1, ChoiceFormat.nextDouble(1)}  
  *formats* = {"no files", "one file", "many files"}  
  (nextDouble can be used to get the next higher double, to make the half-open interval.)

Here is a simple example that shows formatting and parsing:

double[] limits = {1,2,3,4,5,6,7};  
 String[] dayOfWeekNames = {"Sun","Mon","Tue","Wed","Thur","Fri","Sat"};  
 ChoiceFormat form = new ChoiceFormat(limits, dayOfWeekNames);  
 ParsePosition status = new ParsePosition(0);  
 for (double i = 0.0; i <= 8.0; ++i) {  
 status.setIndex(0);  
 System.out.println(i + " -> " + form.format(i) + " -> "  
 + form.parse(form.format(i),status));  
 }

Here is a more complex example, with a pattern format:

double[] filelimits = {0,1,2};  
 String[] filepart = {"are no files","is one file","are {2} files"};  
 ChoiceFormat fileform = new ChoiceFormat(filelimits, filepart);  
 Format[] testFormats = {fileform, null, NumberFormat.getInstance()};  
 MessageFormat pattform = new MessageFormat("There {0} on {1}");  
 pattform.setFormats(testFormats);  
 Object[] testArgs = {null, "ADisk", null};  
 for (int i = 0; i < 4; ++i) {  
 testArgs[0] = new Integer(i);  
 testArgs[2] = testArgs[0];  
 System.out.println(pattform.format(testArgs));  
 }

Specifying a pattern for ChoiceFormat objects is fairly straightforward. For example:

ChoiceFormat fmt = new ChoiceFormat(  
 "-1#is negative| 0#is zero or fraction | 1#is one |1.0<is 1+ |2#is two |2<is more than 2.");  
 System.out.println("Formatter Pattern : " + fmt.toPattern());  
  
 System.out.println("Format with -INF : " + fmt.format(Double.NEGATIVE\_INFINITY));  
 System.out.println("Format with -1.0 : " + fmt.format(-1.0));  
 System.out.println("Format with 0 : " + fmt.format(0));  
 System.out.println("Format with 0.9 : " + fmt.format(0.9));  
 System.out.println("Format with 1.0 : " + fmt.format(1));  
 System.out.println("Format with 1.5 : " + fmt.format(1.5));  
 System.out.println("Format with 2 : " + fmt.format(2));  
 System.out.println("Format with 2.1 : " + fmt.format(2.1));  
 System.out.println("Format with NaN : " + fmt.format(Double.NaN));  
 System.out.println("Format with +INF : " + fmt.format(Double.POSITIVE\_INFINITY));

And the output result would be like the following:

Format with -INF : is negative  
 Format with -1.0 : is negative  
 Format with 0 : is zero or fraction  
 Format with 0.9 : is zero or fraction  
 Format with 1.0 : is one  
 Format with 1.5 : is 1+  
 Format with 2 : is two  
 Format with 2.1 : is more than 2.  
 Format with NaN : is negative  
 Format with +INF : is more than 2.

#### Synchronization

Choice 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:**[DecimalFormat](http://docs.google.com/java/text/DecimalFormat.html), [MessageFormat](http://docs.google.com/java/text/MessageFormat.html), [Serialized Form](http://docs.google.com/serialized-form.html#java.text.ChoiceFormat)

| **Nested Class Summary** | |
| --- | --- |

| **Nested classes/interfaces inherited from class java.text.**[**NumberFormat**](http://docs.google.com/java/text/NumberFormat.html) |
| --- |
| [NumberFormat.Field](http://docs.google.com/java/text/NumberFormat.Field.html) |

| **Field Summary** | |
| --- | --- |

| **Fields inherited from class java.text.**[**NumberFormat**](http://docs.google.com/java/text/NumberFormat.html) |
| --- |
| [FRACTION\_FIELD](http://docs.google.com/java/text/NumberFormat.html#FRACTION_FIELD), [INTEGER\_FIELD](http://docs.google.com/java/text/NumberFormat.html#INTEGER_FIELD) |

| **Constructor Summary** | |
| --- | --- |
| [**ChoiceFormat**](http://docs.google.com/java/text/ChoiceFormat.html#ChoiceFormat(double%5B%5D,%20java.lang.String%5B%5D))(double[] limits, [String](http://docs.google.com/java/lang/String.html)[] formats)            Constructs with the limits and the corresponding formats. |
| [**ChoiceFormat**](http://docs.google.com/java/text/ChoiceFormat.html#ChoiceFormat(java.lang.String))([String](http://docs.google.com/java/lang/String.html) newPattern)            Constructs with limits and corresponding formats based on the pattern. |

| **Method Summary** | |
| --- | --- |
| void | [**applyPattern**](http://docs.google.com/java/text/ChoiceFormat.html#applyPattern(java.lang.String))([String](http://docs.google.com/java/lang/String.html) newPattern)            Sets the pattern. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**clone**](http://docs.google.com/java/text/ChoiceFormat.html#clone())()            Overrides Cloneable |
| boolean | [**equals**](http://docs.google.com/java/text/ChoiceFormat.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Equality comparision between two |
| [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) | [**format**](http://docs.google.com/java/text/ChoiceFormat.html#format(double,%20java.lang.StringBuffer,%20java.text.FieldPosition))(double number, [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) toAppendTo, [FieldPosition](http://docs.google.com/java/text/FieldPosition.html) status)            Returns pattern with formatted double. |
| [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) | [**format**](http://docs.google.com/java/text/ChoiceFormat.html#format(long,%20java.lang.StringBuffer,%20java.text.FieldPosition))(long number, [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) toAppendTo, [FieldPosition](http://docs.google.com/java/text/FieldPosition.html) status)            Specialization of format. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**getFormats**](http://docs.google.com/java/text/ChoiceFormat.html#getFormats())()            Get the formats passed in the constructor. |
| double[] | [**getLimits**](http://docs.google.com/java/text/ChoiceFormat.html#getLimits())()            Get the limits passed in the constructor. |
| int | [**hashCode**](http://docs.google.com/java/text/ChoiceFormat.html#hashCode())()            Generates a hash code for the message format object. |
| static double | [**nextDouble**](http://docs.google.com/java/text/ChoiceFormat.html#nextDouble(double))(double d)            Finds the least double greater than d. |
| static double | [**nextDouble**](http://docs.google.com/java/text/ChoiceFormat.html#nextDouble(double,%20boolean))(double d, boolean positive)            Finds the least double greater than d (if positive == true), or the greatest double less than d (if positive == false). |
| [Number](http://docs.google.com/java/lang/Number.html) | [**parse**](http://docs.google.com/java/text/ChoiceFormat.html#parse(java.lang.String,%20java.text.ParsePosition))([String](http://docs.google.com/java/lang/String.html) text, [ParsePosition](http://docs.google.com/java/text/ParsePosition.html) status)            Parses a Number from the input text. |
| static double | [**previousDouble**](http://docs.google.com/java/text/ChoiceFormat.html#previousDouble(double))(double d)            Finds the greatest double less than d. |
| void | [**setChoices**](http://docs.google.com/java/text/ChoiceFormat.html#setChoices(double%5B%5D,%20java.lang.String%5B%5D))(double[] limits, [String](http://docs.google.com/java/lang/String.html)[] formats)            Set the choices to be used in formatting. |
| [String](http://docs.google.com/java/lang/String.html) | [**toPattern**](http://docs.google.com/java/text/ChoiceFormat.html#toPattern())()            Gets the pattern. |

| **Methods inherited from class java.text.**[**NumberFormat**](http://docs.google.com/java/text/NumberFormat.html) |
| --- |
| [format](http://docs.google.com/java/text/NumberFormat.html#format(double)), [format](http://docs.google.com/java/text/NumberFormat.html#format(long)), [format](http://docs.google.com/java/text/NumberFormat.html#format(java.lang.Object,%20java.lang.StringBuffer,%20java.text.FieldPosition)), [getAvailableLocales](http://docs.google.com/java/text/NumberFormat.html#getAvailableLocales()), [getCurrency](http://docs.google.com/java/text/NumberFormat.html#getCurrency()), [getCurrencyInstance](http://docs.google.com/java/text/NumberFormat.html#getCurrencyInstance()), [getCurrencyInstance](http://docs.google.com/java/text/NumberFormat.html#getCurrencyInstance(java.util.Locale)), [getInstance](http://docs.google.com/java/text/NumberFormat.html#getInstance()), [getInstance](http://docs.google.com/java/text/NumberFormat.html#getInstance(java.util.Locale)), [getIntegerInstance](http://docs.google.com/java/text/NumberFormat.html#getIntegerInstance()), [getIntegerInstance](http://docs.google.com/java/text/NumberFormat.html#getIntegerInstance(java.util.Locale)), [getMaximumFractionDigits](http://docs.google.com/java/text/NumberFormat.html#getMaximumFractionDigits()), [getMaximumIntegerDigits](http://docs.google.com/java/text/NumberFormat.html#getMaximumIntegerDigits()), [getMinimumFractionDigits](http://docs.google.com/java/text/NumberFormat.html#getMinimumFractionDigits()), [getMinimumIntegerDigits](http://docs.google.com/java/text/NumberFormat.html#getMinimumIntegerDigits()), [getNumberInstance](http://docs.google.com/java/text/NumberFormat.html#getNumberInstance()), [getNumberInstance](http://docs.google.com/java/text/NumberFormat.html#getNumberInstance(java.util.Locale)), [getPercentInstance](http://docs.google.com/java/text/NumberFormat.html#getPercentInstance()), [getPercentInstance](http://docs.google.com/java/text/NumberFormat.html#getPercentInstance(java.util.Locale)), [getRoundingMode](http://docs.google.com/java/text/NumberFormat.html#getRoundingMode()), [isGroupingUsed](http://docs.google.com/java/text/NumberFormat.html#isGroupingUsed()), [isParseIntegerOnly](http://docs.google.com/java/text/NumberFormat.html#isParseIntegerOnly()), [parse](http://docs.google.com/java/text/NumberFormat.html#parse(java.lang.String)), [parseObject](http://docs.google.com/java/text/NumberFormat.html#parseObject(java.lang.String,%20java.text.ParsePosition)), [setCurrency](http://docs.google.com/java/text/NumberFormat.html#setCurrency(java.util.Currency)), [setGroupingUsed](http://docs.google.com/java/text/NumberFormat.html#setGroupingUsed(boolean)), [setMaximumFractionDigits](http://docs.google.com/java/text/NumberFormat.html#setMaximumFractionDigits(int)), [setMaximumIntegerDigits](http://docs.google.com/java/text/NumberFormat.html#setMaximumIntegerDigits(int)), [setMinimumFractionDigits](http://docs.google.com/java/text/NumberFormat.html#setMinimumFractionDigits(int)), [setMinimumIntegerDigits](http://docs.google.com/java/text/NumberFormat.html#setMinimumIntegerDigits(int)), [setParseIntegerOnly](http://docs.google.com/java/text/NumberFormat.html#setParseIntegerOnly(boolean)), [setRoundingMode](http://docs.google.com/java/text/NumberFormat.html#setRoundingMode(java.math.RoundingMode)) |

| **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)), [formatToCharacterIterator](http://docs.google.com/java/text/Format.html#formatToCharacterIterator(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** |
| --- |

### ChoiceFormat

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

Constructs with limits and corresponding formats based on the pattern.

**See Also:**[applyPattern(java.lang.String)](http://docs.google.com/java/text/ChoiceFormat.html#applyPattern(java.lang.String))

### ChoiceFormat

public **ChoiceFormat**(double[] limits,  
 [String](http://docs.google.com/java/lang/String.html)[] formats)

Constructs with the limits and the corresponding formats.

**See Also:**[setChoices(double[], java.lang.String[])](http://docs.google.com/java/text/ChoiceFormat.html#setChoices(double%5B%5D,%20java.lang.String%5B%5D))

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

### applyPattern

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

Sets the pattern.

**Parameters:**newPattern - See the class description.

### toPattern

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

Gets the pattern.

### setChoices

public void **setChoices**(double[] limits,  
 [String](http://docs.google.com/java/lang/String.html)[] formats)

Set the choices to be used in formatting.

**Parameters:**limits - contains the top value that you want parsed with that format,and should be in ascending sorted order. When formatting X, the choice will be the i, where limit[i] <= X < limit[i+1]. If the limit array is not in ascending order, the results of formatting will be incorrect.formats - are the formats you want to use for each limit. They can be either Format objects or Strings. When formatting with object Y, if the object is a NumberFormat, then ((NumberFormat) Y).format(X) is called. Otherwise Y.toString() is called.

### getLimits

public double[] **getLimits**()

Get the limits passed in the constructor.

**Returns:**the limits.

### getFormats

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

Get the formats passed in the constructor.

**Returns:**the formats.

### format

public [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) **format**(long number,  
 [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) toAppendTo,  
 [FieldPosition](http://docs.google.com/java/text/FieldPosition.html) status)

Specialization of format. This method really calls format(double, StringBuffer, FieldPosition) thus the range of longs that are supported is only equal to the range that can be stored by double. This will never be a practical limitation.

**Specified by:**[format](http://docs.google.com/java/text/NumberFormat.html#format(long,%20java.lang.StringBuffer,%20java.text.FieldPosition)) in class [NumberFormat](http://docs.google.com/java/text/NumberFormat.html) **See Also:**[Format.format(java.lang.Object)](http://docs.google.com/java/text/Format.html#format(java.lang.Object))

### format

public [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) **format**(double number,  
 [StringBuffer](http://docs.google.com/java/lang/StringBuffer.html) toAppendTo,  
 [FieldPosition](http://docs.google.com/java/text/FieldPosition.html) status)

Returns pattern with formatted double.

**Specified by:**[format](http://docs.google.com/java/text/NumberFormat.html#format(double,%20java.lang.StringBuffer,%20java.text.FieldPosition)) in class [NumberFormat](http://docs.google.com/java/text/NumberFormat.html) **Parameters:**number - number to be formatted & substituted.toAppendTo - where text is appended.status - ignore no useful status is returned.**See Also:**[Format.format(java.lang.Object)](http://docs.google.com/java/text/Format.html#format(java.lang.Object))

### parse

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

Parses a Number from the input text.

**Specified by:**[parse](http://docs.google.com/java/text/NumberFormat.html#parse(java.lang.String,%20java.text.ParsePosition)) in class [NumberFormat](http://docs.google.com/java/text/NumberFormat.html) **Parameters:**text - the source text.status - an input-output parameter. On input, the status.index field indicates the first character of the source text that should be parsed. On exit, if no error occured, status.index is set to the first unparsed character in the source text. On exit, if an error did occur, status.index is unchanged and status.errorIndex is set to the first index of the character that caused the parse to fail. **Returns:**A Number representing the value of the number parsed.**See Also:**[NumberFormat.isParseIntegerOnly()](http://docs.google.com/java/text/NumberFormat.html#isParseIntegerOnly()), [Format.parseObject(java.lang.String, java.text.ParsePosition)](http://docs.google.com/java/text/Format.html#parseObject(java.lang.String,%20java.text.ParsePosition))

### nextDouble

public static final double **nextDouble**(double d)

Finds the least double greater than d. If NaN, returns same value.

Used to make half-open intervals.

**See Also:**[previousDouble(double)](http://docs.google.com/java/text/ChoiceFormat.html#previousDouble(double))

### previousDouble

public static final double **previousDouble**(double d)

Finds the greatest double less than d. If NaN, returns same value.

**See Also:**[nextDouble(double)](http://docs.google.com/java/text/ChoiceFormat.html#nextDouble(double))

### clone

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

Overrides Cloneable

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

### hashCode

public int **hashCode**()

Generates a hash code for the message format object.

**Overrides:**[hashCode](http://docs.google.com/java/text/NumberFormat.html#hashCode()) in class [NumberFormat](http://docs.google.com/java/text/NumberFormat.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)

### equals

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

Equality comparision between two

**Overrides:**[equals](http://docs.google.com/java/text/NumberFormat.html#equals(java.lang.Object)) in class [NumberFormat](http://docs.google.com/java/text/NumberFormat.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)

### nextDouble

public static double **nextDouble**(double d,  
 boolean positive)

Finds the least double greater than d (if positive == true), or the greatest double less than d (if positive == false). If NaN, returns same value. Does not affect floating-point flags, provided these member functions do not: Double.longBitsToDouble(long) Double.doubleToLongBits(double) Double.isNaN(double)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ChoiceFormat.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/CharacterIterator.html)   [**NEXT CLASS**](http://docs.google.com/java/text/CollationElementIterator.html) | [**FRAMES**](http://docs.google.com/index.html?java/text/ChoiceFormat.html)    [**NO FRAMES**](http://docs.google.com/ChoiceFormat.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#tyjcwt) | [FIELD](#1t3h5sf) | [CONSTR](#4d34og8) | [METHOD](#2s8eyo1) | DETAIL: FIELD | [CONSTR](#lnxbz9) | [METHOD](#44sinio) |

[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).