| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Character.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/lang/Byte.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/Character.Subset.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/Character.html)    [**NO FRAMES**](http://docs.google.com/Character.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#tyjcwt) | [FIELD](#3dy6vkm) | [CONSTR](#1t3h5sf) | [METHOD](#4d34og8) | DETAIL: [FIELD](#17dp8vu) | [CONSTR](#4h042r0) | [METHOD](#1baon6m) |

## **java.lang**

Class Character

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

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

public final class **Character**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html), [Comparable](http://docs.google.com/java/lang/Comparable.html)<[Character](http://docs.google.com/java/lang/Character.html)>

The Character class wraps a value of the primitive type char in an object. An object of type Character contains a single field whose type is char.

In addition, this class provides several methods for determining a character's category (lowercase letter, digit, etc.) and for converting characters from uppercase to lowercase and vice versa.

Character information is based on the Unicode Standard, version 4.0.

The methods and data of class Character are defined by the information in the *UnicodeData* file that is part of the Unicode Character Database maintained by the Unicode Consortium. This file specifies various properties including name and general category for every defined Unicode code point or character range.

The file and its description are available from the Unicode Consortium at:

* <http://www.unicode.org>

#### Unicode Character Representations

The char data type (and therefore the value that a Character object encapsulates) are based on the original Unicode specification, which defined characters as fixed-width 16-bit entities. The Unicode standard has since been changed to allow for characters whose representation requires more than 16 bits. The range of legal *code point*s is now U+0000 to U+10FFFF, known as *Unicode scalar value*. (Refer to the  [*definition*](http://www.unicode.org/reports/tr27/#notation) of the U+*n* notation in the Unicode standard.)

The set of characters from U+0000 to U+FFFF is sometimes referred to as the *Basic Multilingual Plane (BMP)*. Characters whose code points are greater than U+FFFF are called *supplementary character*s. The Java 2 platform uses the UTF-16 representation in char arrays and in the String and StringBuffer classes. In this representation, supplementary characters are represented as a pair of char values, the first from the *high-surrogates* range, (\uD800-\uDBFF), the second from the *low-surrogates* range (\uDC00-\uDFFF).

A char value, therefore, represents Basic Multilingual Plane (BMP) code points, including the surrogate code points, or code units of the UTF-16 encoding. An int value represents all Unicode code points, including supplementary code points. The lower (least significant) 21 bits of int are used to represent Unicode code points and the upper (most significant) 11 bits must be zero. Unless otherwise specified, the behavior with respect to supplementary characters and surrogate char values is as follows:

* The methods that only accept a char value cannot support supplementary characters. They treat char values from the surrogate ranges as undefined characters. For example, Character.isLetter('\uD840') returns false, even though this specific value if followed by any low-surrogate value in a string would represent a letter.
* The methods that accept an int value support all Unicode characters, including supplementary characters. For example, Character.isLetter(0x2F81A) returns true because the code point value represents a letter (a CJK ideograph).

In the Java SE API documentation, *Unicode code point* is used for character values in the range between U+0000 and U+10FFFF, and *Unicode code unit* is used for 16-bit char values that are code units of the *UTF-16* encoding. For more information on Unicode terminology, refer to the [Unicode Glossary](http://www.unicode.org/glossary/).

**Since:** 1.0 **See Also:**[Serialized Form](http://docs.google.com/serialized-form.html#java.lang.Character)

| **Nested Class Summary** | |
| --- | --- |
| static class | [**Character.Subset**](http://docs.google.com/java/lang/Character.Subset.html)            Instances of this class represent particular subsets of the Unicode character set. |
| static class | [**Character.UnicodeBlock**](http://docs.google.com/java/lang/Character.UnicodeBlock.html)            A family of character subsets representing the character blocks in the Unicode specification. |

| **Field Summary** | |
| --- | --- |
| static byte | [**COMBINING\_SPACING\_MARK**](http://docs.google.com/java/lang/Character.html#COMBINING_SPACING_MARK)            General category "Mc" in the Unicode specification. |
| static byte | [**CONNECTOR\_PUNCTUATION**](http://docs.google.com/java/lang/Character.html#CONNECTOR_PUNCTUATION)            General category "Pc" in the Unicode specification. |
| static byte | [**CONTROL**](http://docs.google.com/java/lang/Character.html#CONTROL)            General category "Cc" in the Unicode specification. |
| static byte | [**CURRENCY\_SYMBOL**](http://docs.google.com/java/lang/Character.html#CURRENCY_SYMBOL)            General category "Sc" in the Unicode specification. |
| static byte | [**DASH\_PUNCTUATION**](http://docs.google.com/java/lang/Character.html#DASH_PUNCTUATION)            General category "Pd" in the Unicode specification. |
| static byte | [**DECIMAL\_DIGIT\_NUMBER**](http://docs.google.com/java/lang/Character.html#DECIMAL_DIGIT_NUMBER)            General category "Nd" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_ARABIC\_NUMBER**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_ARABIC_NUMBER)            Weak bidirectional character type "AN" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_BOUNDARY\_NEUTRAL**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_BOUNDARY_NEUTRAL)            Weak bidirectional character type "BN" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_COMMON\_NUMBER\_SEPARATOR**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_COMMON_NUMBER_SEPARATOR)            Weak bidirectional character type "CS" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_EUROPEAN\_NUMBER**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_EUROPEAN_NUMBER)            Weak bidirectional character type "EN" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_EUROPEAN\_NUMBER\_SEPARATOR**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_EUROPEAN_NUMBER_SEPARATOR)            Weak bidirectional character type "ES" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_EUROPEAN\_NUMBER\_TERMINATOR**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_EUROPEAN_NUMBER_TERMINATOR)            Weak bidirectional character type "ET" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_LEFT\_TO\_RIGHT**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_LEFT_TO_RIGHT)            Strong bidirectional character type "L" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_LEFT\_TO\_RIGHT\_EMBEDDING**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_LEFT_TO_RIGHT_EMBEDDING)            Strong bidirectional character type "LRE" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_LEFT\_TO\_RIGHT\_OVERRIDE**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_LEFT_TO_RIGHT_OVERRIDE)            Strong bidirectional character type "LRO" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_NONSPACING\_MARK**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_NONSPACING_MARK)            Weak bidirectional character type "NSM" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_OTHER\_NEUTRALS**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_OTHER_NEUTRALS)            Neutral bidirectional character type "ON" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_PARAGRAPH\_SEPARATOR**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_PARAGRAPH_SEPARATOR)            Neutral bidirectional character type "B" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_POP\_DIRECTIONAL\_FORMAT**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_POP_DIRECTIONAL_FORMAT)            Weak bidirectional character type "PDF" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_RIGHT\_TO\_LEFT**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_RIGHT_TO_LEFT)            Strong bidirectional character type "R" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_RIGHT\_TO\_LEFT\_ARABIC**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_RIGHT_TO_LEFT_ARABIC)            Strong bidirectional character type "AL" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_RIGHT\_TO\_LEFT\_EMBEDDING**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_RIGHT_TO_LEFT_EMBEDDING)            Strong bidirectional character type "RLE" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_RIGHT\_TO\_LEFT\_OVERRIDE**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_RIGHT_TO_LEFT_OVERRIDE)            Strong bidirectional character type "RLO" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_SEGMENT\_SEPARATOR**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_SEGMENT_SEPARATOR)            Neutral bidirectional character type "S" in the Unicode specification. |
| static byte | [**DIRECTIONALITY\_UNDEFINED**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_UNDEFINED)            Undefined bidirectional character type. |
| static byte | [**DIRECTIONALITY\_WHITESPACE**](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_WHITESPACE)            Neutral bidirectional character type "WS" in the Unicode specification. |
| static byte | [**ENCLOSING\_MARK**](http://docs.google.com/java/lang/Character.html#ENCLOSING_MARK)            General category "Me" in the Unicode specification. |
| static byte | [**END\_PUNCTUATION**](http://docs.google.com/java/lang/Character.html#END_PUNCTUATION)            General category "Pe" in the Unicode specification. |
| static byte | [**FINAL\_QUOTE\_PUNCTUATION**](http://docs.google.com/java/lang/Character.html#FINAL_QUOTE_PUNCTUATION)            General category "Pf" in the Unicode specification. |
| static byte | [**FORMAT**](http://docs.google.com/java/lang/Character.html#FORMAT)            General category "Cf" in the Unicode specification. |
| static byte | [**INITIAL\_QUOTE\_PUNCTUATION**](http://docs.google.com/java/lang/Character.html#INITIAL_QUOTE_PUNCTUATION)            General category "Pi" in the Unicode specification. |
| static byte | [**LETTER\_NUMBER**](http://docs.google.com/java/lang/Character.html#LETTER_NUMBER)            General category "Nl" in the Unicode specification. |
| static byte | [**LINE\_SEPARATOR**](http://docs.google.com/java/lang/Character.html#LINE_SEPARATOR)            General category "Zl" in the Unicode specification. |
| static byte | [**LOWERCASE\_LETTER**](http://docs.google.com/java/lang/Character.html#LOWERCASE_LETTER)            General category "Ll" in the Unicode specification. |
| static byte | [**MATH\_SYMBOL**](http://docs.google.com/java/lang/Character.html#MATH_SYMBOL)            General category "Sm" in the Unicode specification. |
| static int | [**MAX\_CODE\_POINT**](http://docs.google.com/java/lang/Character.html#MAX_CODE_POINT)            The maximum value of a Unicode code point. |
| static char | [**MAX\_HIGH\_SURROGATE**](http://docs.google.com/java/lang/Character.html#MAX_HIGH_SURROGATE)            The maximum value of a Unicode high-surrogate code unit in the UTF-16 encoding. |
| static char | [**MAX\_LOW\_SURROGATE**](http://docs.google.com/java/lang/Character.html#MAX_LOW_SURROGATE)            The maximum value of a Unicode low-surrogate code unit in the UTF-16 encoding. |
| static int | [**MAX\_RADIX**](http://docs.google.com/java/lang/Character.html#MAX_RADIX)            The maximum radix available for conversion to and from strings. |
| static char | [**MAX\_SURROGATE**](http://docs.google.com/java/lang/Character.html#MAX_SURROGATE)            The maximum value of a Unicode surrogate code unit in the UTF-16 encoding. |
| static char | [**MAX\_VALUE**](http://docs.google.com/java/lang/Character.html#MAX_VALUE)            The constant value of this field is the largest value of type char, '\uFFFF'. |
| static int | [**MIN\_CODE\_POINT**](http://docs.google.com/java/lang/Character.html#MIN_CODE_POINT)            The minimum value of a Unicode code point. |
| static char | [**MIN\_HIGH\_SURROGATE**](http://docs.google.com/java/lang/Character.html#MIN_HIGH_SURROGATE)            The minimum value of a Unicode high-surrogate code unit in the UTF-16 encoding. |
| static char | [**MIN\_LOW\_SURROGATE**](http://docs.google.com/java/lang/Character.html#MIN_LOW_SURROGATE)            The minimum value of a Unicode low-surrogate code unit in the UTF-16 encoding. |
| static int | [**MIN\_RADIX**](http://docs.google.com/java/lang/Character.html#MIN_RADIX)            The minimum radix available for conversion to and from strings. |
| static int | [**MIN\_SUPPLEMENTARY\_CODE\_POINT**](http://docs.google.com/java/lang/Character.html#MIN_SUPPLEMENTARY_CODE_POINT)            The minimum value of a supplementary code point. |
| static char | [**MIN\_SURROGATE**](http://docs.google.com/java/lang/Character.html#MIN_SURROGATE)            The minimum value of a Unicode surrogate code unit in the UTF-16 encoding. |
| static char | [**MIN\_VALUE**](http://docs.google.com/java/lang/Character.html#MIN_VALUE)            The constant value of this field is the smallest value of type char, '\u0000'. |
| static byte | [**MODIFIER\_LETTER**](http://docs.google.com/java/lang/Character.html#MODIFIER_LETTER)            General category "Lm" in the Unicode specification. |
| static byte | [**MODIFIER\_SYMBOL**](http://docs.google.com/java/lang/Character.html#MODIFIER_SYMBOL)            General category "Sk" in the Unicode specification. |
| static byte | [**NON\_SPACING\_MARK**](http://docs.google.com/java/lang/Character.html#NON_SPACING_MARK)            General category "Mn" in the Unicode specification. |
| static byte | [**OTHER\_LETTER**](http://docs.google.com/java/lang/Character.html#OTHER_LETTER)            General category "Lo" in the Unicode specification. |
| static byte | [**OTHER\_NUMBER**](http://docs.google.com/java/lang/Character.html#OTHER_NUMBER)            General category "No" in the Unicode specification. |
| static byte | [**OTHER\_PUNCTUATION**](http://docs.google.com/java/lang/Character.html#OTHER_PUNCTUATION)            General category "Po" in the Unicode specification. |
| static byte | [**OTHER\_SYMBOL**](http://docs.google.com/java/lang/Character.html#OTHER_SYMBOL)            General category "So" in the Unicode specification. |
| static byte | [**PARAGRAPH\_SEPARATOR**](http://docs.google.com/java/lang/Character.html#PARAGRAPH_SEPARATOR)            General category "Zp" in the Unicode specification. |
| static byte | [**PRIVATE\_USE**](http://docs.google.com/java/lang/Character.html#PRIVATE_USE)            General category "Co" in the Unicode specification. |
| static int | [**SIZE**](http://docs.google.com/java/lang/Character.html#SIZE)            The number of bits used to represent a char value in unsigned binary form. |
| static byte | [**SPACE\_SEPARATOR**](http://docs.google.com/java/lang/Character.html#SPACE_SEPARATOR)            General category "Zs" in the Unicode specification. |
| static byte | [**START\_PUNCTUATION**](http://docs.google.com/java/lang/Character.html#START_PUNCTUATION)            General category "Ps" in the Unicode specification. |
| static byte | [**SURROGATE**](http://docs.google.com/java/lang/Character.html#SURROGATE)            General category "Cs" in the Unicode specification. |
| static byte | [**TITLECASE\_LETTER**](http://docs.google.com/java/lang/Character.html#TITLECASE_LETTER)            General category "Lt" in the Unicode specification. |
| static [Class](http://docs.google.com/java/lang/Class.html)<[Character](http://docs.google.com/java/lang/Character.html)> | [**TYPE**](http://docs.google.com/java/lang/Character.html#TYPE)            The Class instance representing the primitive type char. |
| static byte | [**UNASSIGNED**](http://docs.google.com/java/lang/Character.html#UNASSIGNED)            General category "Cn" in the Unicode specification. |
| static byte | [**UPPERCASE\_LETTER**](http://docs.google.com/java/lang/Character.html#UPPERCASE_LETTER)            General category "Lu" in the Unicode specification. |

| **Constructor Summary** | |
| --- | --- |
| [**Character**](http://docs.google.com/java/lang/Character.html#Character(char))(char value)            Constructs a newly allocated Character object that represents the specified char value. |

| **Method Summary** | |
| --- | --- |
| static int | [**charCount**](http://docs.google.com/java/lang/Character.html#charCount(int))(int codePoint)            Determines the number of char values needed to represent the specified character (Unicode code point). |
| char | [**charValue**](http://docs.google.com/java/lang/Character.html#charValue())()            Returns the value of this Character object. |
| static int | [**codePointAt**](http://docs.google.com/java/lang/Character.html#codePointAt(char%5B%5D,%20int))(char[] a, int index)            Returns the code point at the given index of the char array. |
| static int | [**codePointAt**](http://docs.google.com/java/lang/Character.html#codePointAt(char%5B%5D,%20int,%20int))(char[] a, int index, int limit)            Returns the code point at the given index of the char array, where only array elements with index less than limit can be used. |
| static int | [**codePointAt**](http://docs.google.com/java/lang/Character.html#codePointAt(java.lang.CharSequence,%20int))([CharSequence](http://docs.google.com/java/lang/CharSequence.html) seq, int index)            Returns the code point at the given index of the CharSequence. |
| static int | [**codePointBefore**](http://docs.google.com/java/lang/Character.html#codePointBefore(char%5B%5D,%20int))(char[] a, int index)            Returns the code point preceding the given index of the char array. |
| static int | [**codePointBefore**](http://docs.google.com/java/lang/Character.html#codePointBefore(char%5B%5D,%20int,%20int))(char[] a, int index, int start)            Returns the code point preceding the given index of the char array, where only array elements with index greater than or equal to start can be used. |
| static int | [**codePointBefore**](http://docs.google.com/java/lang/Character.html#codePointBefore(java.lang.CharSequence,%20int))([CharSequence](http://docs.google.com/java/lang/CharSequence.html) seq, int index)            Returns the code point preceding the given index of the CharSequence. |
| static int | [**codePointCount**](http://docs.google.com/java/lang/Character.html#codePointCount(char%5B%5D,%20int,%20int))(char[] a, int offset, int count)            Returns the number of Unicode code points in a subarray of the char array argument. |
| static int | [**codePointCount**](http://docs.google.com/java/lang/Character.html#codePointCount(java.lang.CharSequence,%20int,%20int))([CharSequence](http://docs.google.com/java/lang/CharSequence.html) seq, int beginIndex, int endIndex)            Returns the number of Unicode code points in the text range of the specified char sequence. |
| int | [**compareTo**](http://docs.google.com/java/lang/Character.html#compareTo(java.lang.Character))([Character](http://docs.google.com/java/lang/Character.html) anotherCharacter)            Compares two Character objects numerically. |
| static int | [**digit**](http://docs.google.com/java/lang/Character.html#digit(char,%20int))(char ch, int radix)            Returns the numeric value of the character ch in the specified radix. |
| static int | [**digit**](http://docs.google.com/java/lang/Character.html#digit(int,%20int))(int codePoint, int radix)            Returns the numeric value of the specified character (Unicode code point) in the specified radix. |
| boolean | [**equals**](http://docs.google.com/java/lang/Character.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Compares this object against the specified object. |
| static char | [**forDigit**](http://docs.google.com/java/lang/Character.html#forDigit(int,%20int))(int digit, int radix)            Determines the character representation for a specific digit in the specified radix. |
| static byte | [**getDirectionality**](http://docs.google.com/java/lang/Character.html#getDirectionality(char))(char ch)            Returns the Unicode directionality property for the given character. |
| static byte | [**getDirectionality**](http://docs.google.com/java/lang/Character.html#getDirectionality(int))(int codePoint)            Returns the Unicode directionality property for the given character (Unicode code point). |
| static int | [**getNumericValue**](http://docs.google.com/java/lang/Character.html#getNumericValue(char))(char ch)            Returns the int value that the specified Unicode character represents. |
| static int | [**getNumericValue**](http://docs.google.com/java/lang/Character.html#getNumericValue(int))(int codePoint)            Returns the int value that the specified character (Unicode code point) represents. |
| static int | [**getType**](http://docs.google.com/java/lang/Character.html#getType(char))(char ch)            Returns a value indicating a character's general category. |
| static int | [**getType**](http://docs.google.com/java/lang/Character.html#getType(int))(int codePoint)            Returns a value indicating a character's general category. |
| int | [**hashCode**](http://docs.google.com/java/lang/Character.html#hashCode())()            Returns a hash code for this Character. |
| static boolean | [**isDefined**](http://docs.google.com/java/lang/Character.html#isDefined(char))(char ch)            Determines if a character is defined in Unicode. |
| static boolean | [**isDefined**](http://docs.google.com/java/lang/Character.html#isDefined(int))(int codePoint)            Determines if a character (Unicode code point) is defined in Unicode. |
| static boolean | [**isDigit**](http://docs.google.com/java/lang/Character.html#isDigit(char))(char ch)            Determines if the specified character is a digit. |
| static boolean | [**isDigit**](http://docs.google.com/java/lang/Character.html#isDigit(int))(int codePoint)            Determines if the specified character (Unicode code point) is a digit. |
| static boolean | [**isHighSurrogate**](http://docs.google.com/java/lang/Character.html#isHighSurrogate(char))(char ch)            Determines if the given char value is a high-surrogate code unit (also known as *leading-surrogate code unit*). |
| static boolean | [**isIdentifierIgnorable**](http://docs.google.com/java/lang/Character.html#isIdentifierIgnorable(char))(char ch)            Determines if the specified character should be regarded as an ignorable character in a Java identifier or a Unicode identifier. |
| static boolean | [**isIdentifierIgnorable**](http://docs.google.com/java/lang/Character.html#isIdentifierIgnorable(int))(int codePoint)            Determines if the specified character (Unicode code point) should be regarded as an ignorable character in a Java identifier or a Unicode identifier. |
| static boolean | [**isISOControl**](http://docs.google.com/java/lang/Character.html#isISOControl(char))(char ch)            Determines if the specified character is an ISO control character. |
| static boolean | [**isISOControl**](http://docs.google.com/java/lang/Character.html#isISOControl(int))(int codePoint)            Determines if the referenced character (Unicode code point) is an ISO control character. |
| static boolean | [**isJavaIdentifierPart**](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(char))(char ch)            Determines if the specified character may be part of a Java identifier as other than the first character. |
| static boolean | [**isJavaIdentifierPart**](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(int))(int codePoint)            Determines if the character (Unicode code point) may be part of a Java identifier as other than the first character. |
| static boolean | [**isJavaIdentifierStart**](http://docs.google.com/java/lang/Character.html#isJavaIdentifierStart(char))(char ch)            Determines if the specified character is permissible as the first character in a Java identifier. |
| static boolean | [**isJavaIdentifierStart**](http://docs.google.com/java/lang/Character.html#isJavaIdentifierStart(int))(int codePoint)            Determines if the character (Unicode code point) is permissible as the first character in a Java identifier. |
| static boolean | [**isJavaLetter**](http://docs.google.com/java/lang/Character.html#isJavaLetter(char))(char ch)  **Deprecated.** *Replaced by isJavaIdentifierStart(char).* |
| static boolean | [**isJavaLetterOrDigit**](http://docs.google.com/java/lang/Character.html#isJavaLetterOrDigit(char))(char ch)  **Deprecated.** *Replaced by isJavaIdentifierPart(char).* |
| static boolean | [**isLetter**](http://docs.google.com/java/lang/Character.html#isLetter(char))(char ch)            Determines if the specified character is a letter. |
| static boolean | [**isLetter**](http://docs.google.com/java/lang/Character.html#isLetter(int))(int codePoint)            Determines if the specified character (Unicode code point) is a letter. |
| static boolean | [**isLetterOrDigit**](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(char))(char ch)            Determines if the specified character is a letter or digit. |
| static boolean | [**isLetterOrDigit**](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(int))(int codePoint)            Determines if the specified character (Unicode code point) is a letter or digit. |
| static boolean | [**isLowerCase**](http://docs.google.com/java/lang/Character.html#isLowerCase(char))(char ch)            Determines if the specified character is a lowercase character. |
| static boolean | [**isLowerCase**](http://docs.google.com/java/lang/Character.html#isLowerCase(int))(int codePoint)            Determines if the specified character (Unicode code point) is a lowercase character. |
| static boolean | [**isLowSurrogate**](http://docs.google.com/java/lang/Character.html#isLowSurrogate(char))(char ch)            Determines if the given char value is a low-surrogate code unit (also known as *trailing-surrogate code unit*). |
| static boolean | [**isMirrored**](http://docs.google.com/java/lang/Character.html#isMirrored(char))(char ch)            Determines whether the character is mirrored according to the Unicode specification. |
| static boolean | [**isMirrored**](http://docs.google.com/java/lang/Character.html#isMirrored(int))(int codePoint)            Determines whether the specified character (Unicode code point) is mirrored according to the Unicode specification. |
| static boolean | [**isSpace**](http://docs.google.com/java/lang/Character.html#isSpace(char))(char ch)  **Deprecated.** *Replaced by isWhitespace(char).* |
| static boolean | [**isSpaceChar**](http://docs.google.com/java/lang/Character.html#isSpaceChar(char))(char ch)            Determines if the specified character is a Unicode space character. |
| static boolean | [**isSpaceChar**](http://docs.google.com/java/lang/Character.html#isSpaceChar(int))(int codePoint)            Determines if the specified character (Unicode code point) is a Unicode space character. |
| static boolean | [**isSupplementaryCodePoint**](http://docs.google.com/java/lang/Character.html#isSupplementaryCodePoint(int))(int codePoint)            Determines whether the specified character (Unicode code point) is in the supplementary character range. |
| static boolean | [**isSurrogatePair**](http://docs.google.com/java/lang/Character.html#isSurrogatePair(char,%20char))(char high, char low)            Determines whether the specified pair of char values is a valid surrogate pair. |
| static boolean | [**isTitleCase**](http://docs.google.com/java/lang/Character.html#isTitleCase(char))(char ch)            Determines if the specified character is a titlecase character. |
| static boolean | [**isTitleCase**](http://docs.google.com/java/lang/Character.html#isTitleCase(int))(int codePoint)            Determines if the specified character (Unicode code point) is a titlecase character. |
| static boolean | [**isUnicodeIdentifierPart**](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierPart(char))(char ch)            Determines if the specified character may be part of a Unicode identifier as other than the first character. |
| static boolean | [**isUnicodeIdentifierPart**](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierPart(int))(int codePoint)            Determines if the specified character (Unicode code point) may be part of a Unicode identifier as other than the first character. |
| static boolean | [**isUnicodeIdentifierStart**](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierStart(char))(char ch)            Determines if the specified character is permissible as the first character in a Unicode identifier. |
| static boolean | [**isUnicodeIdentifierStart**](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierStart(int))(int codePoint)            Determines if the specified character (Unicode code point) is permissible as the first character in a Unicode identifier. |
| static boolean | [**isUpperCase**](http://docs.google.com/java/lang/Character.html#isUpperCase(char))(char ch)            Determines if the specified character is an uppercase character. |
| static boolean | [**isUpperCase**](http://docs.google.com/java/lang/Character.html#isUpperCase(int))(int codePoint)            Determines if the specified character (Unicode code point) is an uppercase character. |
| static boolean | [**isValidCodePoint**](http://docs.google.com/java/lang/Character.html#isValidCodePoint(int))(int codePoint)            Determines whether the specified code point is a valid Unicode code point value in the range of 0x0000 to 0x10FFFF inclusive. |
| static boolean | [**isWhitespace**](http://docs.google.com/java/lang/Character.html#isWhitespace(char))(char ch)            Determines if the specified character is white space according to Java. |
| static boolean | [**isWhitespace**](http://docs.google.com/java/lang/Character.html#isWhitespace(int))(int codePoint)            Determines if the specified character (Unicode code point) is white space according to Java. |
| static int | [**offsetByCodePoints**](http://docs.google.com/java/lang/Character.html#offsetByCodePoints(char%5B%5D,%20int,%20int,%20int,%20int))(char[] a, int start, int count, int index, int codePointOffset)            Returns the index within the given char subarray that is offset from the given index by codePointOffset code points. |
| static int | [**offsetByCodePoints**](http://docs.google.com/java/lang/Character.html#offsetByCodePoints(java.lang.CharSequence,%20int,%20int))([CharSequence](http://docs.google.com/java/lang/CharSequence.html) seq, int index, int codePointOffset)            Returns the index within the given char sequence that is offset from the given index by codePointOffset code points. |
| static char | [**reverseBytes**](http://docs.google.com/java/lang/Character.html#reverseBytes(char))(char ch)            Returns the value obtained by reversing the order of the bytes in the specified char value. |
| static char[] | [**toChars**](http://docs.google.com/java/lang/Character.html#toChars(int))(int codePoint)            Converts the specified character (Unicode code point) to its UTF-16 representation stored in a char array. |
| static int | [**toChars**](http://docs.google.com/java/lang/Character.html#toChars(int,%20char%5B%5D,%20int))(int codePoint, char[] dst, int dstIndex)            Converts the specified character (Unicode code point) to its UTF-16 representation. |
| static int | [**toCodePoint**](http://docs.google.com/java/lang/Character.html#toCodePoint(char,%20char))(char high, char low)            Converts the specified surrogate pair to its supplementary code point value. |
| static char | [**toLowerCase**](http://docs.google.com/java/lang/Character.html#toLowerCase(char))(char ch)            Converts the character argument to lowercase using case mapping information from the UnicodeData file. |
| static int | [**toLowerCase**](http://docs.google.com/java/lang/Character.html#toLowerCase(int))(int codePoint)            Converts the character (Unicode code point) argument to lowercase using case mapping information from the UnicodeData file. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/lang/Character.html#toString())()            Returns a String object representing this Character's value. |
| static [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/lang/Character.html#toString(char))(char c)            Returns a String object representing the specified char. |
| static char | [**toTitleCase**](http://docs.google.com/java/lang/Character.html#toTitleCase(char))(char ch)            Converts the character argument to titlecase using case mapping information from the UnicodeData file. |
| static int | [**toTitleCase**](http://docs.google.com/java/lang/Character.html#toTitleCase(int))(int codePoint)            Converts the character (Unicode code point) argument to titlecase using case mapping information from the UnicodeData file. |
| static char | [**toUpperCase**](http://docs.google.com/java/lang/Character.html#toUpperCase(char))(char ch)            Converts the character argument to uppercase using case mapping information from the UnicodeData file. |
| static int | [**toUpperCase**](http://docs.google.com/java/lang/Character.html#toUpperCase(int))(int codePoint)            Converts the character (Unicode code point) argument to uppercase using case mapping information from the UnicodeData file. |
| static [Character](http://docs.google.com/java/lang/Character.html) | [**valueOf**](http://docs.google.com/java/lang/Character.html#valueOf(char))(char c)            Returns a Character instance representing the specified char value. |

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

| **Field Detail** |
| --- |

### MIN\_RADIX

public static final int **MIN\_RADIX**

The minimum radix available for conversion to and from strings. The constant value of this field is the smallest value permitted for the radix argument in radix-conversion methods such as the digit method, the forDigit method, and the toString method of class Integer.

**See Also:**[digit(char, int)](http://docs.google.com/java/lang/Character.html#digit(char,%20int)), [forDigit(int, int)](http://docs.google.com/java/lang/Character.html#forDigit(int,%20int)), [Integer.toString(int, int)](http://docs.google.com/java/lang/Integer.html#toString(int,%20int)), [Integer.valueOf(java.lang.String)](http://docs.google.com/java/lang/Integer.html#valueOf(java.lang.String)), [Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MIN_RADIX)

### MAX\_RADIX

public static final int **MAX\_RADIX**

The maximum radix available for conversion to and from strings. The constant value of this field is the largest value permitted for the radix argument in radix-conversion methods such as the digit method, the forDigit method, and the toString method of class Integer.

**See Also:**[digit(char, int)](http://docs.google.com/java/lang/Character.html#digit(char,%20int)), [forDigit(int, int)](http://docs.google.com/java/lang/Character.html#forDigit(int,%20int)), [Integer.toString(int, int)](http://docs.google.com/java/lang/Integer.html#toString(int,%20int)), [Integer.valueOf(java.lang.String)](http://docs.google.com/java/lang/Integer.html#valueOf(java.lang.String)), [Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MAX_RADIX)

### MIN\_VALUE

public static final char **MIN\_VALUE**

The constant value of this field is the smallest value of type char, '\u0000'.

**Since:** 1.0.2 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MIN_VALUE)

### MAX\_VALUE

public static final char **MAX\_VALUE**

The constant value of this field is the largest value of type char, '\uFFFF'.

**Since:** 1.0.2 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MAX_VALUE)

### TYPE

public static final [Class](http://docs.google.com/java/lang/Class.html)<[Character](http://docs.google.com/java/lang/Character.html)> **TYPE**

The Class instance representing the primitive type char.

**Since:** 1.1

### UNASSIGNED

public static final byte **UNASSIGNED**

General category "Cn" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.UNASSIGNED)

### UPPERCASE\_LETTER

public static final byte **UPPERCASE\_LETTER**

General category "Lu" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.UPPERCASE_LETTER)

### LOWERCASE\_LETTER

public static final byte **LOWERCASE\_LETTER**

General category "Ll" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.LOWERCASE_LETTER)

### TITLECASE\_LETTER

public static final byte **TITLECASE\_LETTER**

General category "Lt" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.TITLECASE_LETTER)

### MODIFIER\_LETTER

public static final byte **MODIFIER\_LETTER**

General category "Lm" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MODIFIER_LETTER)

### OTHER\_LETTER

public static final byte **OTHER\_LETTER**

General category "Lo" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.OTHER_LETTER)

### NON\_SPACING\_MARK

public static final byte **NON\_SPACING\_MARK**

General category "Mn" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.NON_SPACING_MARK)

### ENCLOSING\_MARK

public static final byte **ENCLOSING\_MARK**

General category "Me" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.ENCLOSING_MARK)

### COMBINING\_SPACING\_MARK

public static final byte **COMBINING\_SPACING\_MARK**

General category "Mc" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.COMBINING_SPACING_MARK)

### DECIMAL\_DIGIT\_NUMBER

public static final byte **DECIMAL\_DIGIT\_NUMBER**

General category "Nd" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DECIMAL_DIGIT_NUMBER)

### LETTER\_NUMBER

public static final byte **LETTER\_NUMBER**

General category "Nl" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.LETTER_NUMBER)

### OTHER\_NUMBER

public static final byte **OTHER\_NUMBER**

General category "No" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.OTHER_NUMBER)

### SPACE\_SEPARATOR

public static final byte **SPACE\_SEPARATOR**

General category "Zs" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.SPACE_SEPARATOR)

### LINE\_SEPARATOR

public static final byte **LINE\_SEPARATOR**

General category "Zl" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.LINE_SEPARATOR)

### PARAGRAPH\_SEPARATOR

public static final byte **PARAGRAPH\_SEPARATOR**

General category "Zp" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.PARAGRAPH_SEPARATOR)

### CONTROL

public static final byte **CONTROL**

General category "Cc" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.CONTROL)

### FORMAT

public static final byte **FORMAT**

General category "Cf" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.FORMAT)

### PRIVATE\_USE

public static final byte **PRIVATE\_USE**

General category "Co" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.PRIVATE_USE)

### SURROGATE

public static final byte **SURROGATE**

General category "Cs" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.SURROGATE)

### DASH\_PUNCTUATION

public static final byte **DASH\_PUNCTUATION**

General category "Pd" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DASH_PUNCTUATION)

### START\_PUNCTUATION

public static final byte **START\_PUNCTUATION**

General category "Ps" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.START_PUNCTUATION)

### END\_PUNCTUATION

public static final byte **END\_PUNCTUATION**

General category "Pe" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.END_PUNCTUATION)

### CONNECTOR\_PUNCTUATION

public static final byte **CONNECTOR\_PUNCTUATION**

General category "Pc" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.CONNECTOR_PUNCTUATION)

### OTHER\_PUNCTUATION

public static final byte **OTHER\_PUNCTUATION**

General category "Po" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.OTHER_PUNCTUATION)

### MATH\_SYMBOL

public static final byte **MATH\_SYMBOL**

General category "Sm" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MATH_SYMBOL)

### CURRENCY\_SYMBOL

public static final byte **CURRENCY\_SYMBOL**

General category "Sc" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.CURRENCY_SYMBOL)

### MODIFIER\_SYMBOL

public static final byte **MODIFIER\_SYMBOL**

General category "Sk" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MODIFIER_SYMBOL)

### OTHER\_SYMBOL

public static final byte **OTHER\_SYMBOL**

General category "So" in the Unicode specification.

**Since:** 1.1 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.OTHER_SYMBOL)

### INITIAL\_QUOTE\_PUNCTUATION

public static final byte **INITIAL\_QUOTE\_PUNCTUATION**

General category "Pi" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.INITIAL_QUOTE_PUNCTUATION)

### FINAL\_QUOTE\_PUNCTUATION

public static final byte **FINAL\_QUOTE\_PUNCTUATION**

General category "Pf" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.FINAL_QUOTE_PUNCTUATION)

### DIRECTIONALITY\_UNDEFINED

public static final byte **DIRECTIONALITY\_UNDEFINED**

Undefined bidirectional character type. Undefined char values have undefined directionality in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_UNDEFINED)

### DIRECTIONALITY\_LEFT\_TO\_RIGHT

public static final byte **DIRECTIONALITY\_LEFT\_TO\_RIGHT**

Strong bidirectional character type "L" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_LEFT_TO_RIGHT)

### DIRECTIONALITY\_RIGHT\_TO\_LEFT

public static final byte **DIRECTIONALITY\_RIGHT\_TO\_LEFT**

Strong bidirectional character type "R" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_RIGHT_TO_LEFT)

### DIRECTIONALITY\_RIGHT\_TO\_LEFT\_ARABIC

public static final byte **DIRECTIONALITY\_RIGHT\_TO\_LEFT\_ARABIC**

Strong bidirectional character type "AL" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_RIGHT_TO_LEFT_ARABIC)

### DIRECTIONALITY\_EUROPEAN\_NUMBER

public static final byte **DIRECTIONALITY\_EUROPEAN\_NUMBER**

Weak bidirectional character type "EN" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_EUROPEAN_NUMBER)

### DIRECTIONALITY\_EUROPEAN\_NUMBER\_SEPARATOR

public static final byte **DIRECTIONALITY\_EUROPEAN\_NUMBER\_SEPARATOR**

Weak bidirectional character type "ES" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_EUROPEAN_NUMBER_SEPARATOR)

### DIRECTIONALITY\_EUROPEAN\_NUMBER\_TERMINATOR

public static final byte **DIRECTIONALITY\_EUROPEAN\_NUMBER\_TERMINATOR**

Weak bidirectional character type "ET" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_EUROPEAN_NUMBER_TERMINATOR)

### DIRECTIONALITY\_ARABIC\_NUMBER

public static final byte **DIRECTIONALITY\_ARABIC\_NUMBER**

Weak bidirectional character type "AN" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_ARABIC_NUMBER)

### DIRECTIONALITY\_COMMON\_NUMBER\_SEPARATOR

public static final byte **DIRECTIONALITY\_COMMON\_NUMBER\_SEPARATOR**

Weak bidirectional character type "CS" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_COMMON_NUMBER_SEPARATOR)

### DIRECTIONALITY\_NONSPACING\_MARK

public static final byte **DIRECTIONALITY\_NONSPACING\_MARK**

Weak bidirectional character type "NSM" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_NONSPACING_MARK)

### DIRECTIONALITY\_BOUNDARY\_NEUTRAL

public static final byte **DIRECTIONALITY\_BOUNDARY\_NEUTRAL**

Weak bidirectional character type "BN" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_BOUNDARY_NEUTRAL)

### DIRECTIONALITY\_PARAGRAPH\_SEPARATOR

public static final byte **DIRECTIONALITY\_PARAGRAPH\_SEPARATOR**

Neutral bidirectional character type "B" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_PARAGRAPH_SEPARATOR)

### DIRECTIONALITY\_SEGMENT\_SEPARATOR

public static final byte **DIRECTIONALITY\_SEGMENT\_SEPARATOR**

Neutral bidirectional character type "S" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_SEGMENT_SEPARATOR)

### DIRECTIONALITY\_WHITESPACE

public static final byte **DIRECTIONALITY\_WHITESPACE**

Neutral bidirectional character type "WS" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_WHITESPACE)

### DIRECTIONALITY\_OTHER\_NEUTRALS

public static final byte **DIRECTIONALITY\_OTHER\_NEUTRALS**

Neutral bidirectional character type "ON" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_OTHER_NEUTRALS)

### DIRECTIONALITY\_LEFT\_TO\_RIGHT\_EMBEDDING

public static final byte **DIRECTIONALITY\_LEFT\_TO\_RIGHT\_EMBEDDING**

Strong bidirectional character type "LRE" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_LEFT_TO_RIGHT_EMBEDDING)

### DIRECTIONALITY\_LEFT\_TO\_RIGHT\_OVERRIDE

public static final byte **DIRECTIONALITY\_LEFT\_TO\_RIGHT\_OVERRIDE**

Strong bidirectional character type "LRO" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_LEFT_TO_RIGHT_OVERRIDE)

### DIRECTIONALITY\_RIGHT\_TO\_LEFT\_EMBEDDING

public static final byte **DIRECTIONALITY\_RIGHT\_TO\_LEFT\_EMBEDDING**

Strong bidirectional character type "RLE" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_RIGHT_TO_LEFT_EMBEDDING)

### DIRECTIONALITY\_RIGHT\_TO\_LEFT\_OVERRIDE

public static final byte **DIRECTIONALITY\_RIGHT\_TO\_LEFT\_OVERRIDE**

Strong bidirectional character type "RLO" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_RIGHT_TO_LEFT_OVERRIDE)

### DIRECTIONALITY\_POP\_DIRECTIONAL\_FORMAT

public static final byte **DIRECTIONALITY\_POP\_DIRECTIONAL\_FORMAT**

Weak bidirectional character type "PDF" in the Unicode specification.

**Since:** 1.4 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.DIRECTIONALITY_POP_DIRECTIONAL_FORMAT)

### MIN\_HIGH\_SURROGATE

public static final char **MIN\_HIGH\_SURROGATE**

The minimum value of a Unicode high-surrogate code unit in the UTF-16 encoding. A high-surrogate is also known as a *leading-surrogate*.

**Since:** 1.5 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MIN_HIGH_SURROGATE)

### MAX\_HIGH\_SURROGATE

public static final char **MAX\_HIGH\_SURROGATE**

The maximum value of a Unicode high-surrogate code unit in the UTF-16 encoding. A high-surrogate is also known as a *leading-surrogate*.

**Since:** 1.5 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MAX_HIGH_SURROGATE)

### MIN\_LOW\_SURROGATE

public static final char **MIN\_LOW\_SURROGATE**

The minimum value of a Unicode low-surrogate code unit in the UTF-16 encoding. A low-surrogate is also known as a *trailing-surrogate*.

**Since:** 1.5 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MIN_LOW_SURROGATE)

### MAX\_LOW\_SURROGATE

public static final char **MAX\_LOW\_SURROGATE**

The maximum value of a Unicode low-surrogate code unit in the UTF-16 encoding. A low-surrogate is also known as a *trailing-surrogate*.

**Since:** 1.5 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MAX_LOW_SURROGATE)

### MIN\_SURROGATE

public static final char **MIN\_SURROGATE**

The minimum value of a Unicode surrogate code unit in the UTF-16 encoding.

**Since:** 1.5 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MIN_SURROGATE)

### MAX\_SURROGATE

public static final char **MAX\_SURROGATE**

The maximum value of a Unicode surrogate code unit in the UTF-16 encoding.

**Since:** 1.5 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MAX_SURROGATE)

### MIN\_SUPPLEMENTARY\_CODE\_POINT

public static final int **MIN\_SUPPLEMENTARY\_CODE\_POINT**

The minimum value of a supplementary code point.

**Since:** 1.5 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MIN_SUPPLEMENTARY_CODE_POINT)

### MIN\_CODE\_POINT

public static final int **MIN\_CODE\_POINT**

The minimum value of a Unicode code point.

**Since:** 1.5 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MIN_CODE_POINT)

### MAX\_CODE\_POINT

public static final int **MAX\_CODE\_POINT**

The maximum value of a Unicode code point.

**Since:** 1.5 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.MAX_CODE_POINT)

### SIZE

public static final int **SIZE**

The number of bits used to represent a char value in unsigned binary form.

**Since:** 1.5 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Character.SIZE)

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

### Character

public **Character**(char value)

Constructs a newly allocated Character object that represents the specified char value.

**Parameters:**value - the value to be represented by the Character object.

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

### valueOf

public static [Character](http://docs.google.com/java/lang/Character.html) **valueOf**(char c)

Returns a Character instance representing the specified char value. If a new Character instance is not required, this method should generally be used in preference to the constructor [Character(char)](http://docs.google.com/java/lang/Character.html#Character(char)), as this method is likely to yield significantly better space and time performance by caching frequently requested values.

**Parameters:**c - a char value. **Returns:**a Character instance representing c.**Since:** 1.5

### charValue

public char **charValue**()

Returns the value of this Character object.

**Returns:**the primitive char value represented by this object.

### hashCode

public int **hashCode**()

Returns a hash code for this Character.

**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)

### equals

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

Compares this object against the specified object. The result is true if and only if the argument is not null and is a Character object that represents the same char value as this object.

**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 object to compare with. **Returns:**true if the objects are the same; false otherwise.**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### toString

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

Returns a String object representing this Character's value. The result is a string of length 1 whose sole component is the primitive char value represented by this Character object.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a string representation of this object.

### toString

public static [String](http://docs.google.com/java/lang/String.html) **toString**(char c)

Returns a String object representing the specified char. The result is a string of length 1 consisting solely of the specified char.

**Parameters:**c - the char to be converted **Returns:**the string representation of the specified char**Since:** 1.4

### isValidCodePoint

public static boolean **isValidCodePoint**(int codePoint)

Determines whether the specified code point is a valid Unicode code point value in the range of 0x0000 to 0x10FFFF inclusive. This method is equivalent to the expression:

codePoint >= 0x0000 && codePoint <= 0x10FFFF

**Parameters:**codePoint - the Unicode code point to be tested **Returns:**true if the specified code point value is a valid code point value; false otherwise.**Since:** 1.5

### isSupplementaryCodePoint

public static boolean **isSupplementaryCodePoint**(int codePoint)

Determines whether the specified character (Unicode code point) is in the supplementary character range. The method call is equivalent to the expression:

codePoint >= 0x10000 && codePoint <= 0x10FFFF

**Parameters:**codePoint - the character (Unicode code point) to be tested **Returns:**true if the specified character is in the Unicode supplementary character range; false otherwise.**Since:** 1.5

### isHighSurrogate

public static boolean **isHighSurrogate**(char ch)

Determines if the given char value is a high-surrogate code unit (also known as *leading-surrogate code unit*). Such values do not represent characters by themselves, but are used in the representation of [supplementary characters](#2et92p0) in the UTF-16 encoding.

This method returns true if and only if

ch >= '\uD800' && ch <= '\uDBFF'

is true.

**Parameters:**ch - the char value to be tested. **Returns:**true if the char value is between '\uD800' and '\uDBFF' inclusive; false otherwise.**Since:** 1.5 **See Also:**[isLowSurrogate(char)](http://docs.google.com/java/lang/Character.html#isLowSurrogate(char)), [Character.UnicodeBlock.of(int)](http://docs.google.com/java/lang/Character.UnicodeBlock.html#of(int))

### isLowSurrogate

public static boolean **isLowSurrogate**(char ch)

Determines if the given char value is a low-surrogate code unit (also known as *trailing-surrogate code unit*). Such values do not represent characters by themselves, but are used in the representation of [supplementary characters](#2et92p0) in the UTF-16 encoding.

This method returns true if and only if

ch >= '\uDC00' && ch <= '\uDFFF'

is true.

**Parameters:**ch - the char value to be tested. **Returns:**true if the char value is between '\uDC00' and '\uDFFF' inclusive; false otherwise.**Since:** 1.5 **See Also:**[isHighSurrogate(char)](http://docs.google.com/java/lang/Character.html#isHighSurrogate(char))

### isSurrogatePair

public static boolean **isSurrogatePair**(char high,  
 char low)

Determines whether the specified pair of char values is a valid surrogate pair. This method is equivalent to the expression:

isHighSurrogate(high) && isLowSurrogate(low)

**Parameters:**high - the high-surrogate code value to be testedlow - the low-surrogate code value to be tested **Returns:**true if the specified high and low-surrogate code values represent a valid surrogate pair; false otherwise.**Since:** 1.5

### charCount

public static int **charCount**(int codePoint)

Determines the number of char values needed to represent the specified character (Unicode code point). If the specified character is equal to or greater than 0x10000, then the method returns 2. Otherwise, the method returns 1.

This method doesn't validate the specified character to be a valid Unicode code point. The caller must validate the character value using [isValidCodePoint](http://docs.google.com/java/lang/Character.html#isValidCodePoint(int)) if necessary.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**2 if the character is a valid supplementary character; 1 otherwise.**Since:** 1.5 **See Also:**[isSupplementaryCodePoint(int)](http://docs.google.com/java/lang/Character.html#isSupplementaryCodePoint(int))

### toCodePoint

public static int **toCodePoint**(char high,  
 char low)

Converts the specified surrogate pair to its supplementary code point value. This method does not validate the specified surrogate pair. The caller must validate it using [isSurrogatePair](http://docs.google.com/java/lang/Character.html#isSurrogatePair(char,%20char)) if necessary.

**Parameters:**high - the high-surrogate code unitlow - the low-surrogate code unit **Returns:**the supplementary code point composed from the specified surrogate pair.**Since:** 1.5

### codePointAt

public static int **codePointAt**([CharSequence](http://docs.google.com/java/lang/CharSequence.html) seq,  
 int index)

Returns the code point at the given index of the CharSequence. If the char value at the given index in the CharSequence is in the high-surrogate range, the following index is less than the length of the CharSequence, and the char value at the following index is in the low-surrogate range, then the supplementary code point corresponding to this surrogate pair is returned. Otherwise, the char value at the given index is returned.

**Parameters:**seq - a sequence of char values (Unicode code units)index - the index to the char values (Unicode code units) in seq to be converted **Returns:**the Unicode code point at the given index **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if seq is null. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the value index is negative or not less than [seq.length()](http://docs.google.com/java/lang/CharSequence.html#length()).**Since:** 1.5

### codePointAt

public static int **codePointAt**(char[] a,  
 int index)

Returns the code point at the given index of the char array. If the char value at the given index in the char array is in the high-surrogate range, the following index is less than the length of the char array, and the char value at the following index is in the low-surrogate range, then the supplementary code point corresponding to this surrogate pair is returned. Otherwise, the char value at the given index is returned.

**Parameters:**a - the char arrayindex - the index to the char values (Unicode code units) in the char array to be converted **Returns:**the Unicode code point at the given index **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if a is null. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the value index is negative or not less than the length of the char array.**Since:** 1.5

### codePointAt

public static int **codePointAt**(char[] a,  
 int index,  
 int limit)

Returns the code point at the given index of the char array, where only array elements with index less than limit can be used. If the char value at the given index in the char array is in the high-surrogate range, the following index is less than the limit, and the char value at the following index is in the low-surrogate range, then the supplementary code point corresponding to this surrogate pair is returned. Otherwise, the char value at the given index is returned.

**Parameters:**a - the char arrayindex - the index to the char values (Unicode code units) in the char array to be convertedlimit - the index after the last array element that can be used in the char array **Returns:**the Unicode code point at the given index **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if a is null. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the index argument is negative or not less than the limit argument, or if the limit argument is negative or greater than the length of the char array.**Since:** 1.5

### codePointBefore

public static int **codePointBefore**([CharSequence](http://docs.google.com/java/lang/CharSequence.html) seq,  
 int index)

Returns the code point preceding the given index of the CharSequence. If the char value at (index - 1) in the CharSequence is in the low-surrogate range, (index - 2) is not negative, and the char value at (index - 2) in the CharSequence is in the high-surrogate range, then the supplementary code point corresponding to this surrogate pair is returned. Otherwise, the char value at (index - 1) is returned.

**Parameters:**seq - the CharSequence instanceindex - the index following the code point that should be returned **Returns:**the Unicode code point value before the given index. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if seq is null. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the index argument is less than 1 or greater than [seq.length()](http://docs.google.com/java/lang/CharSequence.html#length()).**Since:** 1.5

### codePointBefore

public static int **codePointBefore**(char[] a,  
 int index)

Returns the code point preceding the given index of the char array. If the char value at (index - 1) in the char array is in the low-surrogate range, (index - 2) is not negative, and the char value at (index - 2) in the char array is in the high-surrogate range, then the supplementary code point corresponding to this surrogate pair is returned. Otherwise, the char value at (index - 1) is returned.

**Parameters:**a - the char arrayindex - the index following the code point that should be returned **Returns:**the Unicode code point value before the given index. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if a is null. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the index argument is less than 1 or greater than the length of the char array**Since:** 1.5

### codePointBefore

public static int **codePointBefore**(char[] a,  
 int index,  
 int start)

Returns the code point preceding the given index of the char array, where only array elements with index greater than or equal to start can be used. If the char value at (index - 1) in the char array is in the low-surrogate range, (index - 2) is not less than start, and the char value at (index - 2) in the char array is in the high-surrogate range, then the supplementary code point corresponding to this surrogate pair is returned. Otherwise, the char value at (index - 1) is returned.

**Parameters:**a - the char arrayindex - the index following the code point that should be returnedstart - the index of the first array element in the char array **Returns:**the Unicode code point value before the given index. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if a is null. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the index argument is not greater than the start argument or is greater than the length of the char array, or if the start argument is negative or not less than the length of the char array.**Since:** 1.5

### toChars

public static int **toChars**(int codePoint,  
 char[] dst,  
 int dstIndex)

Converts the specified character (Unicode code point) to its UTF-16 representation. If the specified code point is a BMP (Basic Multilingual Plane or Plane 0) value, the same value is stored in dst[dstIndex], and 1 is returned. If the specified code point is a supplementary character, its surrogate values are stored in dst[dstIndex] (high-surrogate) and dst[dstIndex+1] (low-surrogate), and 2 is returned.

**Parameters:**codePoint - the character (Unicode code point) to be converted.dst - an array of char in which the codePoint's UTF-16 value is stored.dstIndex - the start index into the dst array where the converted value is stored. **Returns:**1 if the code point is a BMP code point, 2 if the code point is a supplementary code point. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the specified codePoint is not a valid Unicode code point. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified dst is null. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if dstIndex is negative or not less than dst.length, or if dst at dstIndex doesn't have enough array element(s) to store the resulting char value(s). (If dstIndex is equal to dst.length-1 and the specified codePoint is a supplementary character, the high-surrogate value is not stored in dst[dstIndex].)**Since:** 1.5

### toChars

public static char[] **toChars**(int codePoint)

Converts the specified character (Unicode code point) to its UTF-16 representation stored in a char array. If the specified code point is a BMP (Basic Multilingual Plane or Plane 0) value, the resulting char array has the same value as codePoint. If the specified code point is a supplementary code point, the resulting char array has the corresponding surrogate pair.

**Parameters:**codePoint - a Unicode code point **Returns:**a char array having codePoint's UTF-16 representation. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the specified codePoint is not a valid Unicode code point.**Since:** 1.5

### codePointCount

public static int **codePointCount**([CharSequence](http://docs.google.com/java/lang/CharSequence.html) seq,  
 int beginIndex,  
 int endIndex)

Returns the number of Unicode code points in the text range of the specified char sequence. The text range begins at the specified beginIndex and extends to the char at index endIndex - 1. Thus the length (in chars) of the text range is endIndex-beginIndex. Unpaired surrogates within the text range count as one code point each.

**Parameters:**seq - the char sequencebeginIndex - the index to the first char of the text range.endIndex - the index after the last char of the text range. **Returns:**the number of Unicode code points in the specified text range **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if seq is null. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the beginIndex is negative, or endIndex is larger than the length of the given sequence, or beginIndex is larger than endIndex.**Since:** 1.5

### codePointCount

public static int **codePointCount**(char[] a,  
 int offset,  
 int count)

Returns the number of Unicode code points in a subarray of the char array argument. The offset argument is the index of the first char of the subarray and the count argument specifies the length of the subarray in chars. Unpaired surrogates within the subarray count as one code point each.

**Parameters:**a - the char arrayoffset - the index of the first char in the given char arraycount - the length of the subarray in chars **Returns:**the number of Unicode code points in the specified subarray **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if a is null. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if offset or count is negative, or if offset + count is larger than the length of the given array.**Since:** 1.5

### offsetByCodePoints

public static int **offsetByCodePoints**([CharSequence](http://docs.google.com/java/lang/CharSequence.html) seq,  
 int index,  
 int codePointOffset)

Returns the index within the given char sequence that is offset from the given index by codePointOffset code points. Unpaired surrogates within the text range given by index and codePointOffset count as one code point each.

**Parameters:**seq - the char sequenceindex - the index to be offsetcodePointOffset - the offset in code points **Returns:**the index within the char sequence **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if seq is null. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if index is negative or larger then the length of the char sequence, or if codePointOffset is positive and the subsequence starting with index has fewer than codePointOffset code points, or if codePointOffset is negative and the subsequence before index has fewer than the absolute value of codePointOffset code points.**Since:** 1.5

### offsetByCodePoints

public static int **offsetByCodePoints**(char[] a,  
 int start,  
 int count,  
 int index,  
 int codePointOffset)

Returns the index within the given char subarray that is offset from the given index by codePointOffset code points. The start and count arguments specify a subarray of the char array. Unpaired surrogates within the text range given by index and codePointOffset count as one code point each.

**Parameters:**a - the char arraystart - the index of the first char of the subarraycount - the length of the subarray in charsindex - the index to be offsetcodePointOffset - the offset in code points **Returns:**the index within the subarray **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if a is null. [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if start or count is negative, or if start + count is larger than the length of the given array, or if index is less than start or larger then start + count, or if codePointOffset is positive and the text range starting with index and ending with start + count - 1 has fewer than codePointOffset code points, or if codePointOffset is negative and the text range starting with start and ending with index - 1 has fewer than the absolute value of codePointOffset code points.**Since:** 1.5

### isLowerCase

public static boolean **isLowerCase**(char ch)

Determines if the specified character is a lowercase character.

A character is lowercase if its general category type, provided by Character.getType(ch), is LOWERCASE\_LETTER.

The following are examples of lowercase characters:

a b c d e f g h i j k l m n o p q r s t u v w x y z  
 '\u00DF' '\u00E0' '\u00E1' '\u00E2' '\u00E3' '\u00E4' '\u00E5' '\u00E6'   
 '\u00E7' '\u00E8' '\u00E9' '\u00EA' '\u00EB' '\u00EC' '\u00ED' '\u00EE'  
 '\u00EF' '\u00F0' '\u00F1' '\u00F2' '\u00F3' '\u00F4' '\u00F5' '\u00F6'  
 '\u00F8' '\u00F9' '\u00FA' '\u00FB' '\u00FC' '\u00FD' '\u00FE' '\u00FF'

Many other Unicode characters are lowercase too.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isLowerCase(int)](http://docs.google.com/java/lang/Character.html#isLowerCase(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character is lowercase; false otherwise.**See Also:**[isLowerCase(char)](http://docs.google.com/java/lang/Character.html#isLowerCase(char)), [isTitleCase(char)](http://docs.google.com/java/lang/Character.html#isTitleCase(char)), [toLowerCase(char)](http://docs.google.com/java/lang/Character.html#toLowerCase(char)), [getType(char)](http://docs.google.com/java/lang/Character.html#getType(char))

### isLowerCase

public static boolean **isLowerCase**(int codePoint)

Determines if the specified character (Unicode code point) is a lowercase character.

A character is lowercase if its general category type, provided by [getType(codePoint)](http://docs.google.com/java/lang/Character.html#getType(char)), is LOWERCASE\_LETTER.

The following are examples of lowercase characters:

a b c d e f g h i j k l m n o p q r s t u v w x y z  
 '\u00DF' '\u00E0' '\u00E1' '\u00E2' '\u00E3' '\u00E4' '\u00E5' '\u00E6'   
 '\u00E7' '\u00E8' '\u00E9' '\u00EA' '\u00EB' '\u00EC' '\u00ED' '\u00EE'  
 '\u00EF' '\u00F0' '\u00F1' '\u00F2' '\u00F3' '\u00F4' '\u00F5' '\u00F6'  
 '\u00F8' '\u00F9' '\u00FA' '\u00FB' '\u00FC' '\u00FD' '\u00FE' '\u00FF'

Many other Unicode characters are lowercase too.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character is lowercase; false otherwise.**Since:** 1.5 **See Also:**[isLowerCase(int)](http://docs.google.com/java/lang/Character.html#isLowerCase(int)), [isTitleCase(int)](http://docs.google.com/java/lang/Character.html#isTitleCase(int)), [toLowerCase(int)](http://docs.google.com/java/lang/Character.html#toLowerCase(int)), [getType(int)](http://docs.google.com/java/lang/Character.html#getType(int))

### isUpperCase

public static boolean **isUpperCase**(char ch)

Determines if the specified character is an uppercase character.

A character is uppercase if its general category type, provided by Character.getType(ch), is UPPERCASE\_LETTER.

The following are examples of uppercase characters:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z  
 '\u00C0' '\u00C1' '\u00C2' '\u00C3' '\u00C4' '\u00C5' '\u00C6' '\u00C7'  
 '\u00C8' '\u00C9' '\u00CA' '\u00CB' '\u00CC' '\u00CD' '\u00CE' '\u00CF'  
 '\u00D0' '\u00D1' '\u00D2' '\u00D3' '\u00D4' '\u00D5' '\u00D6' '\u00D8'  
 '\u00D9' '\u00DA' '\u00DB' '\u00DC' '\u00DD' '\u00DE'

Many other Unicode characters are uppercase too.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isUpperCase(int)](http://docs.google.com/java/lang/Character.html#isUpperCase(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character is uppercase; false otherwise.**Since:** 1.0 **See Also:**[isLowerCase(char)](http://docs.google.com/java/lang/Character.html#isLowerCase(char)), [isTitleCase(char)](http://docs.google.com/java/lang/Character.html#isTitleCase(char)), [toUpperCase(char)](http://docs.google.com/java/lang/Character.html#toUpperCase(char)), [getType(char)](http://docs.google.com/java/lang/Character.html#getType(char))

### isUpperCase

public static boolean **isUpperCase**(int codePoint)

Determines if the specified character (Unicode code point) is an uppercase character.

A character is uppercase if its general category type, provided by [getType(codePoint)](http://docs.google.com/java/lang/Character.html#getType(int)), is UPPERCASE\_LETTER.

The following are examples of uppercase characters:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z  
 '\u00C0' '\u00C1' '\u00C2' '\u00C3' '\u00C4' '\u00C5' '\u00C6' '\u00C7'  
 '\u00C8' '\u00C9' '\u00CA' '\u00CB' '\u00CC' '\u00CD' '\u00CE' '\u00CF'  
 '\u00D0' '\u00D1' '\u00D2' '\u00D3' '\u00D4' '\u00D5' '\u00D6' '\u00D8'  
 '\u00D9' '\u00DA' '\u00DB' '\u00DC' '\u00DD' '\u00DE'

Many other Unicode characters are uppercase too.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character is uppercase; false otherwise.**Since:** 1.5 **See Also:**[isLowerCase(int)](http://docs.google.com/java/lang/Character.html#isLowerCase(int)), [isTitleCase(int)](http://docs.google.com/java/lang/Character.html#isTitleCase(int)), [toUpperCase(int)](http://docs.google.com/java/lang/Character.html#toUpperCase(int)), [getType(int)](http://docs.google.com/java/lang/Character.html#getType(int))

### isTitleCase

public static boolean **isTitleCase**(char ch)

Determines if the specified character is a titlecase character.

A character is a titlecase character if its general category type, provided by Character.getType(ch), is TITLECASE\_LETTER.

Some characters look like pairs of Latin letters. For example, there is an uppercase letter that looks like "LJ" and has a corresponding lowercase letter that looks like "lj". A third form, which looks like "Lj", is the appropriate form to use when rendering a word in lowercase with initial capitals, as for a book title.

These are some of the Unicode characters for which this method returns true:

* LATIN CAPITAL LETTER D WITH SMALL LETTER Z WITH CARON
* LATIN CAPITAL LETTER L WITH SMALL LETTER J
* LATIN CAPITAL LETTER N WITH SMALL LETTER J
* LATIN CAPITAL LETTER D WITH SMALL LETTER Z

Many other Unicode characters are titlecase too.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isTitleCase(int)](http://docs.google.com/java/lang/Character.html#isTitleCase(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character is titlecase; false otherwise.**Since:** 1.0.2 **See Also:**[isLowerCase(char)](http://docs.google.com/java/lang/Character.html#isLowerCase(char)), [isUpperCase(char)](http://docs.google.com/java/lang/Character.html#isUpperCase(char)), [toTitleCase(char)](http://docs.google.com/java/lang/Character.html#toTitleCase(char)), [getType(char)](http://docs.google.com/java/lang/Character.html#getType(char))

### isTitleCase

public static boolean **isTitleCase**(int codePoint)

Determines if the specified character (Unicode code point) is a titlecase character.

A character is a titlecase character if its general category type, provided by [getType(codePoint)](http://docs.google.com/java/lang/Character.html#getType(int)), is TITLECASE\_LETTER.

Some characters look like pairs of Latin letters. For example, there is an uppercase letter that looks like "LJ" and has a corresponding lowercase letter that looks like "lj". A third form, which looks like "Lj", is the appropriate form to use when rendering a word in lowercase with initial capitals, as for a book title.

These are some of the Unicode characters for which this method returns true:

* LATIN CAPITAL LETTER D WITH SMALL LETTER Z WITH CARON
* LATIN CAPITAL LETTER L WITH SMALL LETTER J
* LATIN CAPITAL LETTER N WITH SMALL LETTER J
* LATIN CAPITAL LETTER D WITH SMALL LETTER Z

Many other Unicode characters are titlecase too.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character is titlecase; false otherwise.**Since:** 1.5 **See Also:**[isLowerCase(int)](http://docs.google.com/java/lang/Character.html#isLowerCase(int)), [isUpperCase(int)](http://docs.google.com/java/lang/Character.html#isUpperCase(int)), [toTitleCase(int)](http://docs.google.com/java/lang/Character.html#toTitleCase(int)), [getType(int)](http://docs.google.com/java/lang/Character.html#getType(int))

### isDigit

public static boolean **isDigit**(char ch)

Determines if the specified character is a digit.

A character is a digit if its general category type, provided by Character.getType(ch), is DECIMAL\_DIGIT\_NUMBER.

Some Unicode character ranges that contain digits:

* '\u0030' through '\u0039', ISO-LATIN-1 digits ('0' through '9')
* '\u0660' through '\u0669', Arabic-Indic digits
* '\u06F0' through '\u06F9', Extended Arabic-Indic digits
* '\u0966' through '\u096F', Devanagari digits
* '\uFF10' through '\uFF19', Fullwidth digits

Many other character ranges contain digits as well.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isDigit(int)](http://docs.google.com/java/lang/Character.html#isDigit(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character is a digit; false otherwise.**See Also:**[digit(char, int)](http://docs.google.com/java/lang/Character.html#digit(char,%20int)), [forDigit(int, int)](http://docs.google.com/java/lang/Character.html#forDigit(int,%20int)), [getType(char)](http://docs.google.com/java/lang/Character.html#getType(char))

### isDigit

public static boolean **isDigit**(int codePoint)

Determines if the specified character (Unicode code point) is a digit.

A character is a digit if its general category type, provided by [getType(codePoint)](http://docs.google.com/java/lang/Character.html#getType(int)), is DECIMAL\_DIGIT\_NUMBER.

Some Unicode character ranges that contain digits:

* '\u0030' through '\u0039', ISO-LATIN-1 digits ('0' through '9')
* '\u0660' through '\u0669', Arabic-Indic digits
* '\u06F0' through '\u06F9', Extended Arabic-Indic digits
* '\u0966' through '\u096F', Devanagari digits
* '\uFF10' through '\uFF19', Fullwidth digits

Many other character ranges contain digits as well.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character is a digit; false otherwise.**Since:** 1.5 **See Also:**[forDigit(int, int)](http://docs.google.com/java/lang/Character.html#forDigit(int,%20int)), [getType(int)](http://docs.google.com/java/lang/Character.html#getType(int))

### isDefined

public static boolean **isDefined**(char ch)

Determines if a character is defined in Unicode.

A character is defined if at least one of the following is true:

* It has an entry in the UnicodeData file.
* It has a value in a range defined by the UnicodeData file.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isDefined(int)](http://docs.google.com/java/lang/Character.html#isDefined(int)) method.

**Parameters:**ch - the character to be tested **Returns:**true if the character has a defined meaning in Unicode; false otherwise.**Since:** 1.0.2 **See Also:**[isDigit(char)](http://docs.google.com/java/lang/Character.html#isDigit(char)), [isLetter(char)](http://docs.google.com/java/lang/Character.html#isLetter(char)), [isLetterOrDigit(char)](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(char)), [isLowerCase(char)](http://docs.google.com/java/lang/Character.html#isLowerCase(char)), [isTitleCase(char)](http://docs.google.com/java/lang/Character.html#isTitleCase(char)), [isUpperCase(char)](http://docs.google.com/java/lang/Character.html#isUpperCase(char))

### isDefined

public static boolean **isDefined**(int codePoint)

Determines if a character (Unicode code point) is defined in Unicode.

A character is defined if at least one of the following is true:

* It has an entry in the UnicodeData file.
* It has a value in a range defined by the UnicodeData file.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character has a defined meaning in Unicode; false otherwise.**Since:** 1.5 **See Also:**[isDigit(int)](http://docs.google.com/java/lang/Character.html#isDigit(int)), [isLetter(int)](http://docs.google.com/java/lang/Character.html#isLetter(int)), [isLetterOrDigit(int)](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(int)), [isLowerCase(int)](http://docs.google.com/java/lang/Character.html#isLowerCase(int)), [isTitleCase(int)](http://docs.google.com/java/lang/Character.html#isTitleCase(int)), [isUpperCase(int)](http://docs.google.com/java/lang/Character.html#isUpperCase(int))

### isLetter

public static boolean **isLetter**(char ch)

Determines if the specified character is a letter.

A character is considered to be a letter if its general category type, provided by Character.getType(ch), is any of the following:

* UPPERCASE\_LETTER
* LOWERCASE\_LETTER
* TITLECASE\_LETTER
* MODIFIER\_LETTER
* OTHER\_LETTER

Not all letters have case. Many characters are letters but are neither uppercase nor lowercase nor titlecase.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isLetter(int)](http://docs.google.com/java/lang/Character.html#isLetter(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character is a letter; false otherwise.**See Also:**[isDigit(char)](http://docs.google.com/java/lang/Character.html#isDigit(char)), [isJavaIdentifierStart(char)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierStart(char)), [isJavaLetter(char)](http://docs.google.com/java/lang/Character.html#isJavaLetter(char)), [isJavaLetterOrDigit(char)](http://docs.google.com/java/lang/Character.html#isJavaLetterOrDigit(char)), [isLetterOrDigit(char)](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(char)), [isLowerCase(char)](http://docs.google.com/java/lang/Character.html#isLowerCase(char)), [isTitleCase(char)](http://docs.google.com/java/lang/Character.html#isTitleCase(char)), [isUnicodeIdentifierStart(char)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierStart(char)), [isUpperCase(char)](http://docs.google.com/java/lang/Character.html#isUpperCase(char))

### isLetter

public static boolean **isLetter**(int codePoint)

Determines if the specified character (Unicode code point) is a letter.

A character is considered to be a letter if its general category type, provided by [getType(codePoint)](http://docs.google.com/java/lang/Character.html#getType(int)), is any of the following:

* UPPERCASE\_LETTER
* LOWERCASE\_LETTER
* TITLECASE\_LETTER
* MODIFIER\_LETTER
* OTHER\_LETTER

Not all letters have case. Many characters are letters but are neither uppercase nor lowercase nor titlecase.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character is a letter; false otherwise.**Since:** 1.5 **See Also:**[isDigit(int)](http://docs.google.com/java/lang/Character.html#isDigit(int)), [isJavaIdentifierStart(int)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierStart(int)), [isLetterOrDigit(int)](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(int)), [isLowerCase(int)](http://docs.google.com/java/lang/Character.html#isLowerCase(int)), [isTitleCase(int)](http://docs.google.com/java/lang/Character.html#isTitleCase(int)), [isUnicodeIdentifierStart(int)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierStart(int)), [isUpperCase(int)](http://docs.google.com/java/lang/Character.html#isUpperCase(int))

### isLetterOrDigit

public static boolean **isLetterOrDigit**(char ch)

Determines if the specified character is a letter or digit.

A character is considered to be a letter or digit if either Character.isLetter(char ch) or Character.isDigit(char ch) returns true for the character.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isLetterOrDigit(int)](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character is a letter or digit; false otherwise.**Since:** 1.0.2 **See Also:**[isDigit(char)](http://docs.google.com/java/lang/Character.html#isDigit(char)), [isJavaIdentifierPart(char)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(char)), [isJavaLetter(char)](http://docs.google.com/java/lang/Character.html#isJavaLetter(char)), [isJavaLetterOrDigit(char)](http://docs.google.com/java/lang/Character.html#isJavaLetterOrDigit(char)), [isLetter(char)](http://docs.google.com/java/lang/Character.html#isLetter(char)), [isUnicodeIdentifierPart(char)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierPart(char))

### isLetterOrDigit

public static boolean **isLetterOrDigit**(int codePoint)

Determines if the specified character (Unicode code point) is a letter or digit.

A character is considered to be a letter or digit if either [isLetter(codePoint)](http://docs.google.com/java/lang/Character.html#isLetter(int)) or [isDigit(codePoint)](http://docs.google.com/java/lang/Character.html#isDigit(int)) returns true for the character.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character is a letter or digit; false otherwise.**Since:** 1.5 **See Also:**[isDigit(int)](http://docs.google.com/java/lang/Character.html#isDigit(int)), [isJavaIdentifierPart(int)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(int)), [isLetter(int)](http://docs.google.com/java/lang/Character.html#isLetter(int)), [isUnicodeIdentifierPart(int)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierPart(int))

### isJavaLetter

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public static boolean **isJavaLetter**(char ch)

**Deprecated.** *Replaced by isJavaIdentifierStart(char).*

Determines if the specified character is permissible as the first character in a Java identifier.

A character may start a Java identifier if and only if one of the following is true:

* [isLetter(ch)](http://docs.google.com/java/lang/Character.html#isLetter(char)) returns true
* [getType(ch)](http://docs.google.com/java/lang/Character.html#getType(char)) returns LETTER\_NUMBER
* ch is a currency symbol (such as "$")
* ch is a connecting punctuation character (such as "\_").

**Parameters:**ch - the character to be tested. **Returns:**true if the character may start a Java identifier; false otherwise.**Since:** 1.02 **See Also:**[isJavaLetterOrDigit(char)](http://docs.google.com/java/lang/Character.html#isJavaLetterOrDigit(char)), [isJavaIdentifierStart(char)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierStart(char)), [isJavaIdentifierPart(char)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(char)), [isLetter(char)](http://docs.google.com/java/lang/Character.html#isLetter(char)), [isLetterOrDigit(char)](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(char)), [isUnicodeIdentifierStart(char)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierStart(char))

### isJavaLetterOrDigit

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public static boolean **isJavaLetterOrDigit**(char ch)

**Deprecated.** *Replaced by isJavaIdentifierPart(char).*

Determines if the specified character may be part of a Java identifier as other than the first character.

A character may be part of a Java identifier if and only if any of the following are true:

* it is a letter
* it is a currency symbol (such as '$')
* it is a connecting punctuation character (such as '\_')
* it is a digit
* it is a numeric letter (such as a Roman numeral character)
* it is a combining mark
* it is a non-spacing mark
* isIdentifierIgnorable returns true for the character.

**Parameters:**ch - the character to be tested. **Returns:**true if the character may be part of a Java identifier; false otherwise.**Since:** 1.02 **See Also:**[isJavaLetter(char)](http://docs.google.com/java/lang/Character.html#isJavaLetter(char)), [isJavaIdentifierStart(char)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierStart(char)), [isJavaIdentifierPart(char)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(char)), [isLetter(char)](http://docs.google.com/java/lang/Character.html#isLetter(char)), [isLetterOrDigit(char)](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(char)), [isUnicodeIdentifierPart(char)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierPart(char)), [isIdentifierIgnorable(char)](http://docs.google.com/java/lang/Character.html#isIdentifierIgnorable(char))

### isJavaIdentifierStart

public static boolean **isJavaIdentifierStart**(char ch)

Determines if the specified character is permissible as the first character in a Java identifier.

A character may start a Java identifier if and only if one of the following conditions is true:

* [isLetter(ch)](http://docs.google.com/java/lang/Character.html#isLetter(char)) returns true
* [getType(ch)](http://docs.google.com/java/lang/Character.html#getType(char)) returns LETTER\_NUMBER
* ch is a currency symbol (such as "$")
* ch is a connecting punctuation character (such as "\_").

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isJavaIdentifierStart(int)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierStart(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character may start a Java identifier; false otherwise.**Since:** 1.1 **See Also:**[isJavaIdentifierPart(char)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(char)), [isLetter(char)](http://docs.google.com/java/lang/Character.html#isLetter(char)), [isUnicodeIdentifierStart(char)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierStart(char)), [SourceVersion.isIdentifier(CharSequence)](http://docs.google.com/javax/lang/model/SourceVersion.html#isIdentifier(java.lang.CharSequence))

### isJavaIdentifierStart

public static boolean **isJavaIdentifierStart**(int codePoint)

Determines if the character (Unicode code point) is permissible as the first character in a Java identifier.

A character may start a Java identifier if and only if one of the following conditions is true:

* [isLetter(codePoint)](http://docs.google.com/java/lang/Character.html#isLetter(int)) returns true
* [getType(codePoint)](http://docs.google.com/java/lang/Character.html#getType(int)) returns LETTER\_NUMBER
* the referenced character is a currency symbol (such as "$")
* the referenced character is a connecting punctuation character (such as "\_").

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character may start a Java identifier; false otherwise.**Since:** 1.5 **See Also:**[isJavaIdentifierPart(int)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(int)), [isLetter(int)](http://docs.google.com/java/lang/Character.html#isLetter(int)), [isUnicodeIdentifierStart(int)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierStart(int)), [SourceVersion.isIdentifier(CharSequence)](http://docs.google.com/javax/lang/model/SourceVersion.html#isIdentifier(java.lang.CharSequence))

### isJavaIdentifierPart

public static boolean **isJavaIdentifierPart**(char ch)

Determines if the specified character may be part of a Java identifier as other than the first character.

A character may be part of a Java identifier if any of the following are true:

* it is a letter
* it is a currency symbol (such as '$')
* it is a connecting punctuation character (such as '\_')
* it is a digit
* it is a numeric letter (such as a Roman numeral character)
* it is a combining mark
* it is a non-spacing mark
* isIdentifierIgnorable returns true for the character

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isJavaIdentifierPart(int)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character may be part of a Java identifier; false otherwise.**Since:** 1.1 **See Also:**[isIdentifierIgnorable(char)](http://docs.google.com/java/lang/Character.html#isIdentifierIgnorable(char)), [isJavaIdentifierStart(char)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierStart(char)), [isLetterOrDigit(char)](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(char)), [isUnicodeIdentifierPart(char)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierPart(char)), [SourceVersion.isIdentifier(CharSequence)](http://docs.google.com/javax/lang/model/SourceVersion.html#isIdentifier(java.lang.CharSequence))

### isJavaIdentifierPart

public static boolean **isJavaIdentifierPart**(int codePoint)

Determines if the character (Unicode code point) may be part of a Java identifier as other than the first character.

A character may be part of a Java identifier if any of the following are true:

* it is a letter
* it is a currency symbol (such as '$')
* it is a connecting punctuation character (such as '\_')
* it is a digit
* it is a numeric letter (such as a Roman numeral character)
* it is a combining mark
* it is a non-spacing mark
* [isIdentifierIgnorable(codePoint)](http://docs.google.com/java/lang/Character.html#isIdentifierIgnorable(int)) returns true for the character

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character may be part of a Java identifier; false otherwise.**Since:** 1.5 **See Also:**[isIdentifierIgnorable(int)](http://docs.google.com/java/lang/Character.html#isIdentifierIgnorable(int)), [isJavaIdentifierStart(int)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierStart(int)), [isLetterOrDigit(int)](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(int)), [isUnicodeIdentifierPart(int)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierPart(int)), [SourceVersion.isIdentifier(CharSequence)](http://docs.google.com/javax/lang/model/SourceVersion.html#isIdentifier(java.lang.CharSequence))

### isUnicodeIdentifierStart

public static boolean **isUnicodeIdentifierStart**(char ch)

Determines if the specified character is permissible as the first character in a Unicode identifier.

A character may start a Unicode identifier if and only if one of the following conditions is true:

* [isLetter(ch)](http://docs.google.com/java/lang/Character.html#isLetter(char)) returns true
* [getType(ch)](http://docs.google.com/java/lang/Character.html#getType(char)) returns LETTER\_NUMBER.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isUnicodeIdentifierStart(int)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierStart(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character may start a Unicode identifier; false otherwise.**Since:** 1.1 **See Also:**[isJavaIdentifierStart(char)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierStart(char)), [isLetter(char)](http://docs.google.com/java/lang/Character.html#isLetter(char)), [isUnicodeIdentifierPart(char)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierPart(char))

### isUnicodeIdentifierStart

public static boolean **isUnicodeIdentifierStart**(int codePoint)

Determines if the specified character (Unicode code point) is permissible as the first character in a Unicode identifier.

A character may start a Unicode identifier if and only if one of the following conditions is true:

* [isLetter(codePoint)](http://docs.google.com/java/lang/Character.html#isLetter(int)) returns true
* [getType(codePoint)](http://docs.google.com/java/lang/Character.html#getType(int)) returns LETTER\_NUMBER.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character may start a Unicode identifier; false otherwise.**Since:** 1.5 **See Also:**[isJavaIdentifierStart(int)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierStart(int)), [isLetter(int)](http://docs.google.com/java/lang/Character.html#isLetter(int)), [isUnicodeIdentifierPart(int)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierPart(int))

### isUnicodeIdentifierPart

public static boolean **isUnicodeIdentifierPart**(char ch)

Determines if the specified character may be part of a Unicode identifier as other than the first character.

A character may be part of a Unicode identifier if and only if one of the following statements is true:

* it is a letter
* it is a connecting punctuation character (such as '\_')
* it is a digit
* it is a numeric letter (such as a Roman numeral character)
* it is a combining mark
* it is a non-spacing mark
* isIdentifierIgnorable returns true for this character.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isUnicodeIdentifierPart(int)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierPart(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character may be part of a Unicode identifier; false otherwise.**Since:** 1.1 **See Also:**[isIdentifierIgnorable(char)](http://docs.google.com/java/lang/Character.html#isIdentifierIgnorable(char)), [isJavaIdentifierPart(char)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(char)), [isLetterOrDigit(char)](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(char)), [isUnicodeIdentifierStart(char)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierStart(char))

### isUnicodeIdentifierPart

public static boolean **isUnicodeIdentifierPart**(int codePoint)

Determines if the specified character (Unicode code point) may be part of a Unicode identifier as other than the first character.

A character may be part of a Unicode identifier if and only if one of the following statements is true:

* it is a letter
* it is a connecting punctuation character (such as '\_')
* it is a digit
* it is a numeric letter (such as a Roman numeral character)
* it is a combining mark
* it is a non-spacing mark
* isIdentifierIgnorable returns true for this character.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character may be part of a Unicode identifier; false otherwise.**Since:** 1.5 **See Also:**[isIdentifierIgnorable(int)](http://docs.google.com/java/lang/Character.html#isIdentifierIgnorable(int)), [isJavaIdentifierPart(int)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(int)), [isLetterOrDigit(int)](http://docs.google.com/java/lang/Character.html#isLetterOrDigit(int)), [isUnicodeIdentifierStart(int)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierStart(int))

### isIdentifierIgnorable

public static boolean **isIdentifierIgnorable**(char ch)

Determines if the specified character should be regarded as an ignorable character in a Java identifier or a Unicode identifier.

The following Unicode characters are ignorable in a Java identifier or a Unicode identifier:

* ISO control characters that are not whitespace
  + '\u0000' through '\u0008'
  + '\u000E' through '\u001B'
  + '\u007F' through '\u009F'
* all characters that have the FORMAT general category value

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isIdentifierIgnorable(int)](http://docs.google.com/java/lang/Character.html#isIdentifierIgnorable(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character is an ignorable control character that may be part of a Java or Unicode identifier; false otherwise.**Since:** 1.1 **See Also:**[isJavaIdentifierPart(char)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(char)), [isUnicodeIdentifierPart(char)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierPart(char))

### isIdentifierIgnorable

public static boolean **isIdentifierIgnorable**(int codePoint)

Determines if the specified character (Unicode code point) should be regarded as an ignorable character in a Java identifier or a Unicode identifier.

The following Unicode characters are ignorable in a Java identifier or a Unicode identifier:

* ISO control characters that are not whitespace
  + '\u0000' through '\u0008'
  + '\u000E' through '\u001B'
  + '\u007F' through '\u009F'
* all characters that have the FORMAT general category value

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character is an ignorable control character that may be part of a Java or Unicode identifier; false otherwise.**Since:** 1.5 **See Also:**[isJavaIdentifierPart(int)](http://docs.google.com/java/lang/Character.html#isJavaIdentifierPart(int)), [isUnicodeIdentifierPart(int)](http://docs.google.com/java/lang/Character.html#isUnicodeIdentifierPart(int))

### toLowerCase

public static char **toLowerCase**(char ch)

Converts the character argument to lowercase using case mapping information from the UnicodeData file.

Note that Character.isLowerCase(Character.toLowerCase(ch)) does not always return true for some ranges of characters, particularly those that are symbols or ideographs.

In general, [String.toLowerCase()](http://docs.google.com/java/lang/String.html#toLowerCase()) should be used to map characters to lowercase. String case mapping methods have several benefits over Character case mapping methods. String case mapping methods can perform locale-sensitive mappings, context-sensitive mappings, and 1:M character mappings, whereas the Character case mapping methods cannot.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [toLowerCase(int)](http://docs.google.com/java/lang/Character.html#toLowerCase(int)) method.

**Parameters:**ch - the character to be converted. **Returns:**the lowercase equivalent of the character, if any; otherwise, the character itself.**See Also:**[isLowerCase(char)](http://docs.google.com/java/lang/Character.html#isLowerCase(char)), [String.toLowerCase()](http://docs.google.com/java/lang/String.html#toLowerCase())

### toLowerCase

public static int **toLowerCase**(int codePoint)

Converts the character (Unicode code point) argument to lowercase using case mapping information from the UnicodeData file.

Note that Character.isLowerCase(Character.toLowerCase(codePoint)) does not always return true for some ranges of characters, particularly those that are symbols or ideographs.

In general, [String.toLowerCase()](http://docs.google.com/java/lang/String.html#toLowerCase()) should be used to map characters to lowercase. String case mapping methods have several benefits over Character case mapping methods. String case mapping methods can perform locale-sensitive mappings, context-sensitive mappings, and 1:M character mappings, whereas the Character case mapping methods cannot.

**Parameters:**codePoint - the character (Unicode code point) to be converted. **Returns:**the lowercase equivalent of the character (Unicode code point), if any; otherwise, the character itself.**Since:** 1.5 **See Also:**[isLowerCase(int)](http://docs.google.com/java/lang/Character.html#isLowerCase(int)), [String.toLowerCase()](http://docs.google.com/java/lang/String.html#toLowerCase())

### toUpperCase

public static char **toUpperCase**(char ch)

Converts the character argument to uppercase using case mapping information from the UnicodeData file.

Note that Character.isUpperCase(Character.toUpperCase(ch)) does not always return true for some ranges of characters, particularly those that are symbols or ideographs.

In general, [String.toUpperCase()](http://docs.google.com/java/lang/String.html#toUpperCase()) should be used to map characters to uppercase. String case mapping methods have several benefits over Character case mapping methods. String case mapping methods can perform locale-sensitive mappings, context-sensitive mappings, and 1:M character mappings, whereas the Character case mapping methods cannot.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [toUpperCase(int)](http://docs.google.com/java/lang/Character.html#toUpperCase(int)) method.

**Parameters:**ch - the character to be converted. **Returns:**the uppercase equivalent of the character, if any; otherwise, the character itself.**See Also:**[isUpperCase(char)](http://docs.google.com/java/lang/Character.html#isUpperCase(char)), [String.toUpperCase()](http://docs.google.com/java/lang/String.html#toUpperCase())

### toUpperCase

public static int **toUpperCase**(int codePoint)

Converts the character (Unicode code point) argument to uppercase using case mapping information from the UnicodeData file.

Note that Character.isUpperCase(Character.toUpperCase(codePoint)) does not always return true for some ranges of characters, particularly those that are symbols or ideographs.

In general, [String.toUpperCase()](http://docs.google.com/java/lang/String.html#toUpperCase()) should be used to map characters to uppercase. String case mapping methods have several benefits over Character case mapping methods. String case mapping methods can perform locale-sensitive mappings, context-sensitive mappings, and 1:M character mappings, whereas the Character case mapping methods cannot.

**Parameters:**codePoint - the character (Unicode code point) to be converted. **Returns:**the uppercase equivalent of the character, if any; otherwise, the character itself.**Since:** 1.5 **See Also:**[isUpperCase(int)](http://docs.google.com/java/lang/Character.html#isUpperCase(int)), [String.toUpperCase()](http://docs.google.com/java/lang/String.html#toUpperCase())

### toTitleCase

public static char **toTitleCase**(char ch)

Converts the character argument to titlecase using case mapping information from the UnicodeData file. If a character has no explicit titlecase mapping and is not itself a titlecase char according to UnicodeData, then the uppercase mapping is returned as an equivalent titlecase mapping. If the char argument is already a titlecase char, the same char value will be returned.

Note that Character.isTitleCase(Character.toTitleCase(ch)) does not always return true for some ranges of characters.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [toTitleCase(int)](http://docs.google.com/java/lang/Character.html#toTitleCase(int)) method.

**Parameters:**ch - the character to be converted. **Returns:**the titlecase equivalent of the character, if any; otherwise, the character itself.**Since:** 1.0.2 **See Also:**[isTitleCase(char)](http://docs.google.com/java/lang/Character.html#isTitleCase(char)), [toLowerCase(char)](http://docs.google.com/java/lang/Character.html#toLowerCase(char)), [toUpperCase(char)](http://docs.google.com/java/lang/Character.html#toUpperCase(char))

### toTitleCase

public static int **toTitleCase**(int codePoint)

Converts the character (Unicode code point) argument to titlecase using case mapping information from the UnicodeData file. If a character has no explicit titlecase mapping and is not itself a titlecase char according to UnicodeData, then the uppercase mapping is returned as an equivalent titlecase mapping. If the character argument is already a titlecase character, the same character value will be returned.

Note that Character.isTitleCase(Character.toTitleCase(codePoint)) does not always return true for some ranges of characters.

**Parameters:**codePoint - the character (Unicode code point) to be converted. **Returns:**the titlecase equivalent of the character, if any; otherwise, the character itself.**Since:** 1.5 **See Also:**[isTitleCase(int)](http://docs.google.com/java/lang/Character.html#isTitleCase(int)), [toLowerCase(int)](http://docs.google.com/java/lang/Character.html#toLowerCase(int)), [toUpperCase(int)](http://docs.google.com/java/lang/Character.html#toUpperCase(int))

### digit

public static int **digit**(char ch,  
 int radix)

Returns the numeric value of the character ch in the specified radix.

If the radix is not in the range MIN\_RADIX <= radix <= MAX\_RADIX or if the value of ch is not a valid digit in the specified radix, -1 is returned. A character is a valid digit if at least one of the following is true:

* The method isDigit is true of the character and the Unicode decimal digit value of the character (or its single-character decomposition) is less than the specified radix. In this case the decimal digit value is returned.
* The character is one of the uppercase Latin letters 'A' through 'Z' and its code is less than radix + 'A' - 10. In this case, ch - 'A' + 10 is returned.
* The character is one of the lowercase Latin letters 'a' through 'z' and its code is less than radix + 'a' - 10. In this case, ch - 'a' + 10 is returned.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [digit(int, int)](http://docs.google.com/java/lang/Character.html#digit(int,%20int)) method.

**Parameters:**ch - the character to be converted.radix - the radix. **Returns:**the numeric value represented by the character in the specified radix.**See Also:**[forDigit(int, int)](http://docs.google.com/java/lang/Character.html#forDigit(int,%20int)), [isDigit(char)](http://docs.google.com/java/lang/Character.html#isDigit(char))

### digit

public static int **digit**(int codePoint,  
 int radix)

Returns the numeric value of the specified character (Unicode code point) in the specified radix.

If the radix is not in the range MIN\_RADIX <= radix <= MAX\_RADIX or if the character is not a valid digit in the specified radix, -1 is returned. A character is a valid digit if at least one of the following is true:

* The method [isDigit(codePoint)](http://docs.google.com/java/lang/Character.html#isDigit(int)) is true of the character and the Unicode decimal digit value of the character (or its single-character decomposition) is less than the specified radix. In this case the decimal digit value is returned.
* The character is one of the uppercase Latin letters 'A' through 'Z' and its code is less than radix + 'A' - 10. In this case, ch - 'A' + 10 is returned.
* The character is one of the lowercase Latin letters 'a' through 'z' and its code is less than radix + 'a' - 10. In this case, ch - 'a' + 10 is returned.

**Parameters:**codePoint - the character (Unicode code point) to be converted.radix - the radix. **Returns:**the numeric value represented by the character in the specified radix.**Since:** 1.5 **See Also:**[forDigit(int, int)](http://docs.google.com/java/lang/Character.html#forDigit(int,%20int)), [isDigit(int)](http://docs.google.com/java/lang/Character.html#isDigit(int))

### getNumericValue

public static int **getNumericValue**(char ch)

Returns the int value that the specified Unicode character represents. For example, the character '\u216C' (the roman numeral fifty) will return an int with a value of 50.

The letters A-Z in their uppercase ('\u0041' through '\u005A'), lowercase ('\u0061' through '\u007A'), and full width variant ('\uFF21' through '\uFF3A' and '\uFF41' through '\uFF5A') forms have numeric values from 10 through 35. This is independent of the Unicode specification, which does not assign numeric values to these char values.

If the character does not have a numeric value, then -1 is returned. If the character has a numeric value that cannot be represented as a nonnegative integer (for example, a fractional value), then -2 is returned.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [getNumericValue(int)](http://docs.google.com/java/lang/Character.html#getNumericValue(int)) method.

**Parameters:**ch - the character to be converted. **Returns:**the numeric value of the character, as a nonnegative int value; -2 if the character has a numeric value that is not a nonnegative integer; -1 if the character has no numeric value.**Since:** 1.1 **See Also:**[forDigit(int, int)](http://docs.google.com/java/lang/Character.html#forDigit(int,%20int)), [isDigit(char)](http://docs.google.com/java/lang/Character.html#isDigit(char))

### getNumericValue

public static int **getNumericValue**(int codePoint)

Returns the int value that the specified character (Unicode code point) represents. For example, the character '\u216C' (the Roman numeral fifty) will return an int with a value of 50.

The letters A-Z in their uppercase ('\u0041' through '\u005A'), lowercase ('\u0061' through '\u007A'), and full width variant ('\uFF21' through '\uFF3A' and '\uFF41' through '\uFF5A') forms have numeric values from 10 through 35. This is independent of the Unicode specification, which does not assign numeric values to these char values.

If the character does not have a numeric value, then -1 is returned. If the character has a numeric value that cannot be represented as a nonnegative integer (for example, a fractional value), then -2 is returned.

**Parameters:**codePoint - the character (Unicode code point) to be converted. **Returns:**the numeric value of the character, as a nonnegative int value; -2 if the character has a numeric value that is not a nonnegative integer; -1 if the character has no numeric value.**Since:** 1.5 **See Also:**[forDigit(int, int)](http://docs.google.com/java/lang/Character.html#forDigit(int,%20int)), [isDigit(int)](http://docs.google.com/java/lang/Character.html#isDigit(int))

### isSpace

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public static boolean **isSpace**(char ch)

**Deprecated.** *Replaced by isWhitespace(char).*

Determines if the specified character is ISO-LATIN-1 white space. This method returns true for the following five characters only:

| '\t' | '\u0009' | HORIZONTAL TABULATION |
| --- | --- | --- |
| '\n' | '\u000A' | NEW LINE |
| '\f' | '\u000C' | FORM FEED |
| '\r' | '\u000D' | CARRIAGE RETURN |
| ' ' | '\u0020' | SPACE |

**Parameters:**ch - the character to be tested. **Returns:**true if the character is ISO-LATIN-1 white space; false otherwise.**See Also:**[isSpaceChar(char)](http://docs.google.com/java/lang/Character.html#isSpaceChar(char)), [isWhitespace(char)](http://docs.google.com/java/lang/Character.html#isWhitespace(char))

### isSpaceChar

public static boolean **isSpaceChar**(char ch)

Determines if the specified character is a Unicode space character. A character is considered to be a space character if and only if it is specified to be a space character by the Unicode standard. This method returns true if the character's general category type is any of the following:

* SPACE\_SEPARATOR
* LINE\_SEPARATOR
* PARAGRAPH\_SEPARATOR

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isSpaceChar(int)](http://docs.google.com/java/lang/Character.html#isSpaceChar(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character is a space character; false otherwise.**Since:** 1.1 **See Also:**[isWhitespace(char)](http://docs.google.com/java/lang/Character.html#isWhitespace(char))

### isSpaceChar

public static boolean **isSpaceChar**(int codePoint)

Determines if the specified character (Unicode code point) is a Unicode space character. A character is considered to be a space character if and only if it is specified to be a space character by the Unicode standard. This method returns true if the character's general category type is any of the following:

* [SPACE\_SEPARATOR](http://docs.google.com/java/lang/Character.html#SPACE_SEPARATOR)
* [LINE\_SEPARATOR](http://docs.google.com/java/lang/Character.html#LINE_SEPARATOR)
* [PARAGRAPH\_SEPARATOR](http://docs.google.com/java/lang/Character.html#PARAGRAPH_SEPARATOR)

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character is a space character; false otherwise.**Since:** 1.5 **See Also:**[isWhitespace(int)](http://docs.google.com/java/lang/Character.html#isWhitespace(int))

### isWhitespace

public static boolean **isWhitespace**(char ch)

Determines if the specified character is white space according to Java. A character is a Java whitespace character if and only if it satisfies one of the following criteria:

* It is a Unicode space character (SPACE\_SEPARATOR, LINE\_SEPARATOR, or PARAGRAPH\_SEPARATOR) but is not also a non-breaking space ('\u00A0', '\u2007', '\u202F').
* It is '\u0009', HORIZONTAL TABULATION.
* It is '\u000A', LINE FEED.
* It is '\u000B', VERTICAL TABULATION.
* It is '\u000C', FORM FEED.
* It is '\u000D', CARRIAGE RETURN.
* It is '\u001C', FILE SEPARATOR.
* It is '\u001D', GROUP SEPARATOR.
* It is '\u001E', RECORD SEPARATOR.
* It is '\u001F', UNIT SEPARATOR.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isWhitespace(int)](http://docs.google.com/java/lang/Character.html#isWhitespace(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character is a Java whitespace character; false otherwise.**Since:** 1.1 **See Also:**[isSpaceChar(char)](http://docs.google.com/java/lang/Character.html#isSpaceChar(char))

### isWhitespace

public static boolean **isWhitespace**(int codePoint)

Determines if the specified character (Unicode code point) is white space according to Java. A character is a Java whitespace character if and only if it satisfies one of the following criteria:

* It is a Unicode space character ([SPACE\_SEPARATOR](http://docs.google.com/java/lang/Character.html#SPACE_SEPARATOR), [LINE\_SEPARATOR](http://docs.google.com/java/lang/Character.html#LINE_SEPARATOR), or [PARAGRAPH\_SEPARATOR](http://docs.google.com/java/lang/Character.html#PARAGRAPH_SEPARATOR)) but is not also a non-breaking space ('\u00A0', '\u2007', '\u202F').
* It is '\u0009', HORIZONTAL TABULATION.
* It is '\u000A', LINE FEED.
* It is '\u000B', VERTICAL TABULATION.
* It is '\u000C', FORM FEED.
* It is '\u000D', CARRIAGE RETURN.
* It is '\u001C', FILE SEPARATOR.
* It is '\u001D', GROUP SEPARATOR.
* It is '\u001E', RECORD SEPARATOR.
* It is '\u001F', UNIT SEPARATOR.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character is a Java whitespace character; false otherwise.**Since:** 1.5 **See Also:**[isSpaceChar(int)](http://docs.google.com/java/lang/Character.html#isSpaceChar(int))

### isISOControl

public static boolean **isISOControl**(char ch)

Determines if the specified character is an ISO control character. A character is considered to be an ISO control character if its code is in the range '\u0000' through '\u001F' or in the range '\u007F' through '\u009F'.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isISOControl(int)](http://docs.google.com/java/lang/Character.html#isISOControl(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**true if the character is an ISO control character; false otherwise.**Since:** 1.1 **See Also:**[isSpaceChar(char)](http://docs.google.com/java/lang/Character.html#isSpaceChar(char)), [isWhitespace(char)](http://docs.google.com/java/lang/Character.html#isWhitespace(char))

### isISOControl

public static boolean **isISOControl**(int codePoint)

Determines if the referenced character (Unicode code point) is an ISO control character. A character is considered to be an ISO control character if its code is in the range '\u0000' through '\u001F' or in the range '\u007F' through '\u009F'.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character is an ISO control character; false otherwise.**Since:** 1.5 **See Also:**[isSpaceChar(int)](http://docs.google.com/java/lang/Character.html#isSpaceChar(int)), [isWhitespace(int)](http://docs.google.com/java/lang/Character.html#isWhitespace(int))

### getType

public static int **getType**(char ch)

Returns a value indicating a character's general category.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [getType(int)](http://docs.google.com/java/lang/Character.html#getType(int)) method.

**Parameters:**ch - the character to be tested. **Returns:**a value of type int representing the character's general category.**Since:** 1.1 **See Also:**[COMBINING\_SPACING\_MARK](http://docs.google.com/java/lang/Character.html#COMBINING_SPACING_MARK), [CONNECTOR\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#CONNECTOR_PUNCTUATION), [CONTROL](http://docs.google.com/java/lang/Character.html#CONTROL), [CURRENCY\_SYMBOL](http://docs.google.com/java/lang/Character.html#CURRENCY_SYMBOL), [DASH\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#DASH_PUNCTUATION), [DECIMAL\_DIGIT\_NUMBER](http://docs.google.com/java/lang/Character.html#DECIMAL_DIGIT_NUMBER), [ENCLOSING\_MARK](http://docs.google.com/java/lang/Character.html#ENCLOSING_MARK), [END\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#END_PUNCTUATION), [FINAL\_QUOTE\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#FINAL_QUOTE_PUNCTUATION), [FORMAT](http://docs.google.com/java/lang/Character.html#FORMAT), [INITIAL\_QUOTE\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#INITIAL_QUOTE_PUNCTUATION), [LETTER\_NUMBER](http://docs.google.com/java/lang/Character.html#LETTER_NUMBER), [LINE\_SEPARATOR](http://docs.google.com/java/lang/Character.html#LINE_SEPARATOR), [LOWERCASE\_LETTER](http://docs.google.com/java/lang/Character.html#LOWERCASE_LETTER), [MATH\_SYMBOL](http://docs.google.com/java/lang/Character.html#MATH_SYMBOL), [MODIFIER\_LETTER](http://docs.google.com/java/lang/Character.html#MODIFIER_LETTER), [MODIFIER\_SYMBOL](http://docs.google.com/java/lang/Character.html#MODIFIER_SYMBOL), [NON\_SPACING\_MARK](http://docs.google.com/java/lang/Character.html#NON_SPACING_MARK), [OTHER\_LETTER](http://docs.google.com/java/lang/Character.html#OTHER_LETTER), [OTHER\_NUMBER](http://docs.google.com/java/lang/Character.html#OTHER_NUMBER), [OTHER\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#OTHER_PUNCTUATION), [OTHER\_SYMBOL](http://docs.google.com/java/lang/Character.html#OTHER_SYMBOL), [PARAGRAPH\_SEPARATOR](http://docs.google.com/java/lang/Character.html#PARAGRAPH_SEPARATOR), [PRIVATE\_USE](http://docs.google.com/java/lang/Character.html#PRIVATE_USE), [SPACE\_SEPARATOR](http://docs.google.com/java/lang/Character.html#SPACE_SEPARATOR), [START\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#START_PUNCTUATION), [SURROGATE](http://docs.google.com/java/lang/Character.html#SURROGATE), [TITLECASE\_LETTER](http://docs.google.com/java/lang/Character.html#TITLECASE_LETTER), [UNASSIGNED](http://docs.google.com/java/lang/Character.html#UNASSIGNED), [UPPERCASE\_LETTER](http://docs.google.com/java/lang/Character.html#UPPERCASE_LETTER)

### getType

public static int **getType**(int codePoint)

Returns a value indicating a character's general category.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**a value of type int representing the character's general category.**Since:** 1.5 **See Also:**[COMBINING\_SPACING\_MARK](http://docs.google.com/java/lang/Character.html#COMBINING_SPACING_MARK), [CONNECTOR\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#CONNECTOR_PUNCTUATION), [CONTROL](http://docs.google.com/java/lang/Character.html#CONTROL), [CURRENCY\_SYMBOL](http://docs.google.com/java/lang/Character.html#CURRENCY_SYMBOL), [DASH\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#DASH_PUNCTUATION), [DECIMAL\_DIGIT\_NUMBER](http://docs.google.com/java/lang/Character.html#DECIMAL_DIGIT_NUMBER), [ENCLOSING\_MARK](http://docs.google.com/java/lang/Character.html#ENCLOSING_MARK), [END\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#END_PUNCTUATION), [FINAL\_QUOTE\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#FINAL_QUOTE_PUNCTUATION), [FORMAT](http://docs.google.com/java/lang/Character.html#FORMAT), [INITIAL\_QUOTE\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#INITIAL_QUOTE_PUNCTUATION), [LETTER\_NUMBER](http://docs.google.com/java/lang/Character.html#LETTER_NUMBER), [LINE\_SEPARATOR](http://docs.google.com/java/lang/Character.html#LINE_SEPARATOR), [LOWERCASE\_LETTER](http://docs.google.com/java/lang/Character.html#LOWERCASE_LETTER), [MATH\_SYMBOL](http://docs.google.com/java/lang/Character.html#MATH_SYMBOL), [MODIFIER\_LETTER](http://docs.google.com/java/lang/Character.html#MODIFIER_LETTER), [MODIFIER\_SYMBOL](http://docs.google.com/java/lang/Character.html#MODIFIER_SYMBOL), [NON\_SPACING\_MARK](http://docs.google.com/java/lang/Character.html#NON_SPACING_MARK), [OTHER\_LETTER](http://docs.google.com/java/lang/Character.html#OTHER_LETTER), [OTHER\_NUMBER](http://docs.google.com/java/lang/Character.html#OTHER_NUMBER), [OTHER\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#OTHER_PUNCTUATION), [OTHER\_SYMBOL](http://docs.google.com/java/lang/Character.html#OTHER_SYMBOL), [PARAGRAPH\_SEPARATOR](http://docs.google.com/java/lang/Character.html#PARAGRAPH_SEPARATOR), [PRIVATE\_USE](http://docs.google.com/java/lang/Character.html#PRIVATE_USE), [SPACE\_SEPARATOR](http://docs.google.com/java/lang/Character.html#SPACE_SEPARATOR), [START\_PUNCTUATION](http://docs.google.com/java/lang/Character.html#START_PUNCTUATION), [SURROGATE](http://docs.google.com/java/lang/Character.html#SURROGATE), [TITLECASE\_LETTER](http://docs.google.com/java/lang/Character.html#TITLECASE_LETTER), [UNASSIGNED](http://docs.google.com/java/lang/Character.html#UNASSIGNED), [UPPERCASE\_LETTER](http://docs.google.com/java/lang/Character.html#UPPERCASE_LETTER)

### forDigit

public static char **forDigit**(int digit,  
 int radix)

Determines the character representation for a specific digit in the specified radix. If the value of radix is not a valid radix, or the value of digit is not a valid digit in the specified radix, the null character ('\u0000') is returned.

The radix argument is valid if it is greater than or equal to MIN\_RADIX and less than or equal to MAX\_RADIX. The digit argument is valid if 0 <=digit < radix.

If the digit is less than 10, then '0' + digit is returned. Otherwise, the value 'a' + digit - 10 is returned.

**Parameters:**digit - the number to convert to a character.radix - the radix. **Returns:**the char representation of the specified digit in the specified radix.**See Also:**[MIN\_RADIX](http://docs.google.com/java/lang/Character.html#MIN_RADIX), [MAX\_RADIX](http://docs.google.com/java/lang/Character.html#MAX_RADIX), [digit(char, int)](http://docs.google.com/java/lang/Character.html#digit(char,%20int))

### getDirectionality

public static byte **getDirectionality**(char ch)

Returns the Unicode directionality property for the given character. Character directionality is used to calculate the visual ordering of text. The directionality value of undefined char values is DIRECTIONALITY\_UNDEFINED.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [getDirectionality(int)](http://docs.google.com/java/lang/Character.html#getDirectionality(int)) method.

**Parameters:**ch - char for which the directionality property is requested. **Returns:**the directionality property of the char value.**Since:** 1.4 **See Also:**[DIRECTIONALITY\_UNDEFINED](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_UNDEFINED), [DIRECTIONALITY\_LEFT\_TO\_RIGHT](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_LEFT_TO_RIGHT), [DIRECTIONALITY\_RIGHT\_TO\_LEFT](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_RIGHT_TO_LEFT), [DIRECTIONALITY\_RIGHT\_TO\_LEFT\_ARABIC](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_RIGHT_TO_LEFT_ARABIC), [DIRECTIONALITY\_EUROPEAN\_NUMBER](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_EUROPEAN_NUMBER), [DIRECTIONALITY\_EUROPEAN\_NUMBER\_SEPARATOR](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_EUROPEAN_NUMBER_SEPARATOR), [DIRECTIONALITY\_EUROPEAN\_NUMBER\_TERMINATOR](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_EUROPEAN_NUMBER_TERMINATOR), [DIRECTIONALITY\_ARABIC\_NUMBER](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_ARABIC_NUMBER), [DIRECTIONALITY\_COMMON\_NUMBER\_SEPARATOR](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_COMMON_NUMBER_SEPARATOR), [DIRECTIONALITY\_NONSPACING\_MARK](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_NONSPACING_MARK), [DIRECTIONALITY\_BOUNDARY\_NEUTRAL](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_BOUNDARY_NEUTRAL), [DIRECTIONALITY\_PARAGRAPH\_SEPARATOR](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_PARAGRAPH_SEPARATOR), [DIRECTIONALITY\_SEGMENT\_SEPARATOR](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_SEGMENT_SEPARATOR), [DIRECTIONALITY\_WHITESPACE](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_WHITESPACE), [DIRECTIONALITY\_OTHER\_NEUTRALS](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_OTHER_NEUTRALS), [DIRECTIONALITY\_LEFT\_TO\_RIGHT\_EMBEDDING](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_LEFT_TO_RIGHT_EMBEDDING), [DIRECTIONALITY\_LEFT\_TO\_RIGHT\_OVERRIDE](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_LEFT_TO_RIGHT_OVERRIDE), [DIRECTIONALITY\_RIGHT\_TO\_LEFT\_EMBEDDING](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_RIGHT_TO_LEFT_EMBEDDING), [DIRECTIONALITY\_RIGHT\_TO\_LEFT\_OVERRIDE](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_RIGHT_TO_LEFT_OVERRIDE), [DIRECTIONALITY\_POP\_DIRECTIONAL\_FORMAT](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_POP_DIRECTIONAL_FORMAT)

### getDirectionality

public static byte **getDirectionality**(int codePoint)

Returns the Unicode directionality property for the given character (Unicode code point). Character directionality is used to calculate the visual ordering of text. The directionality value of undefined character is [DIRECTIONALITY\_UNDEFINED](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_UNDEFINED).

**Parameters:**codePoint - the character (Unicode code point) for which the directionality property is requested. **Returns:**the directionality property of the character.**Since:** 1.5 **See Also:**[DIRECTIONALITY\_UNDEFINED](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_UNDEFINED), [DIRECTIONALITY\_LEFT\_TO\_RIGHT](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_LEFT_TO_RIGHT), [DIRECTIONALITY\_RIGHT\_TO\_LEFT](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_RIGHT_TO_LEFT), [DIRECTIONALITY\_RIGHT\_TO\_LEFT\_ARABIC](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_RIGHT_TO_LEFT_ARABIC), [DIRECTIONALITY\_EUROPEAN\_NUMBER](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_EUROPEAN_NUMBER), [DIRECTIONALITY\_EUROPEAN\_NUMBER\_SEPARATOR](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_EUROPEAN_NUMBER_SEPARATOR), [DIRECTIONALITY\_EUROPEAN\_NUMBER\_TERMINATOR](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_EUROPEAN_NUMBER_TERMINATOR), [DIRECTIONALITY\_ARABIC\_NUMBER](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_ARABIC_NUMBER), [DIRECTIONALITY\_COMMON\_NUMBER\_SEPARATOR](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_COMMON_NUMBER_SEPARATOR), [DIRECTIONALITY\_NONSPACING\_MARK](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_NONSPACING_MARK), [DIRECTIONALITY\_BOUNDARY\_NEUTRAL](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_BOUNDARY_NEUTRAL), [DIRECTIONALITY\_PARAGRAPH\_SEPARATOR](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_PARAGRAPH_SEPARATOR), [DIRECTIONALITY\_SEGMENT\_SEPARATOR](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_SEGMENT_SEPARATOR), [DIRECTIONALITY\_WHITESPACE](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_WHITESPACE), [DIRECTIONALITY\_OTHER\_NEUTRALS](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_OTHER_NEUTRALS), [DIRECTIONALITY\_LEFT\_TO\_RIGHT\_EMBEDDING](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_LEFT_TO_RIGHT_EMBEDDING), [DIRECTIONALITY\_LEFT\_TO\_RIGHT\_OVERRIDE](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_LEFT_TO_RIGHT_OVERRIDE), [DIRECTIONALITY\_RIGHT\_TO\_LEFT\_EMBEDDING](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_RIGHT_TO_LEFT_EMBEDDING), [DIRECTIONALITY\_RIGHT\_TO\_LEFT\_OVERRIDE](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_RIGHT_TO_LEFT_OVERRIDE), [DIRECTIONALITY\_POP\_DIRECTIONAL\_FORMAT](http://docs.google.com/java/lang/Character.html#DIRECTIONALITY_POP_DIRECTIONAL_FORMAT)

### isMirrored

public static boolean **isMirrored**(char ch)

Determines whether the character is mirrored according to the Unicode specification. Mirrored characters should have their glyphs horizontally mirrored when displayed in text that is right-to-left. For example, '\u0028' LEFT PARENTHESIS is semantically defined to be an *opening parenthesis*. This will appear as a "(" in text that is left-to-right but as a ")" in text that is right-to-left.

**Note:** This method cannot handle  [supplementary characters](#2et92p0). To support all Unicode characters, including supplementary characters, use the [isMirrored(int)](http://docs.google.com/java/lang/Character.html#isMirrored(int)) method.

**Parameters:**ch - char for which the mirrored property is requested **Returns:**true if the char is mirrored, false if the char is not mirrored or is not defined.**Since:** 1.4

### isMirrored

public static boolean **isMirrored**(int codePoint)

Determines whether the specified character (Unicode code point) is mirrored according to the Unicode specification. Mirrored characters should have their glyphs horizontally mirrored when displayed in text that is right-to-left. For example, '\u0028' LEFT PARENTHESIS is semantically defined to be an *opening parenthesis*. This will appear as a "(" in text that is left-to-right but as a ")" in text that is right-to-left.

**Parameters:**codePoint - the character (Unicode code point) to be tested. **Returns:**true if the character is mirrored, false if the character is not mirrored or is not defined.**Since:** 1.5

### compareTo

public int **compareTo**([Character](http://docs.google.com/java/lang/Character.html) anotherCharacter)

Compares two Character objects numerically.

**Specified by:**[compareTo](http://docs.google.com/java/lang/Comparable.html#compareTo(T)) in interface [Comparable](http://docs.google.com/java/lang/Comparable.html)<[Character](http://docs.google.com/java/lang/Character.html)> **Parameters:**anotherCharacter - the Character to be compared. **Returns:**the value 0 if the argument Character is equal to this Character; a value less than 0 if this Character is numerically less than the Character argument; and a value greater than 0 if this Character is numerically greater than the Character argument (unsigned comparison). Note that this is strictly a numerical comparison; it is not locale-dependent.**Since:** 1.2

### reverseBytes

public static char **reverseBytes**(char ch)

Returns the value obtained by reversing the order of the bytes in the specified char value.

**Returns:**the value obtained by reversing (or, equivalently, swapping) the bytes in the specified char value.**Since:** 1.5

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Character.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/lang/Byte.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/Character.Subset.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/Character.html)    [**NO FRAMES**](http://docs.google.com/Character.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#tyjcwt) | [FIELD](#3dy6vkm) | [CONSTR](#1t3h5sf) | [METHOD](#4d34og8) | DETAIL: [FIELD](#17dp8vu) | [CONSTR](#4h042r0) | [METHOD](#1baon6m) |

[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).