NSScanner Class Reference

Cocoa > Data Management

2008-10-15

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

NSScanner Class Reference 5

Overview 5
Adopted Protocols 6
Tasks 6
- Creating an Scanner 6
- Getting a Scanner's String 6
- Configuring a Scanner 6
- Scanning a String 7

Class Methods 7
- localizedScannerWithString: 7
- scannerWithString: 8

Instance Methods 8
- caseSensitive 8
- charactersToBeSkipped 9
- initWithString: 9
- isAtEnd 10
- locale 10
- scanCharactersFromSet:intoString: 11
- scanDecimal: 11
- scanDouble: 12
- scanFloat: 12
- scanHexDouble: 13
- scanHexFloat: 14
- scanHexInt: 14
- scanHexLongLong: 14
- scanInt: 15
- scanInteger: 15
- scanLocation 16
- scanLongLong: 16
- scanString:intoString: 17
- scanUpToCharactersFromSet:intoString: 17
- scanUpToString:intoString: 18
- setCaseSensitive: 19
- setCharactersToBeSkipped: 19
- setLocale: 20
- setScanLocation: 20
- string 21

Document Revision History 23

Index 25

NSScanner Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	String Programming Guide for Cocoa
Declared in	NSDecimalNumber.h NSScanner.h
Related sample code	iSpend NumberInput_IMKit_Sample QTAudioExtractionPanel Quartz Composer QCTV Sproing

Overview

The `NSScanner` class is an abstract superclass of a class cluster that declares the programmatic interface for an object that scans values from an `NSString` object.

An `NSScanner` object interprets and converts the characters of an `NSString` object into number and string values. You assign the scanner's string on creating it, and the scanner progresses through the characters of that string from beginning to end as you request items.

Because of the nature of class clusters, scanner objects aren't actual instances of the `NSScanner` class but one of its private subclasses. Although a scanner object's class is private, its interface is public, as declared by this abstract superclass, `NSScanner`. The primitive methods of `NSScanner` are `string` (page 21) and all of the methods listed under "Configuring a Scanner" (page 6) in the "Methods by Task" section. The objects you create using this class are referred to as scanner objects (and when no confusion will result, merely as scanners).

You can set an `NSScanner` object to ignore a set of characters as it scans the string using the `setCharactersToBeSkipped:` (page 19) method. The default set of characters to skip is the whitespace and newline character set.

To retrieve the unscanned remainder of the string, use `[[scanner string]substringFromIndex:[scanner scanLocation]]`.

Adopted Protocols

NSCopying
- `copyWithZone:`

Tasks

Creating an Scanner

`+ scannerWithString:` (page 8)
Returns an `NSScanner` object that scans a given string.

`+ localizedScannerWithString:` (page 7)
Returns an `NSScanner` object that scans a given string according to the user's default locale.

`- initWithString:` (page 9)
Returns an `NSScanner` object initialized to scan a given string.

Getting a Scanner's String

`- string` (page 21)
Returns the string with which the receiver was created or initialized.

Configuring a Scanner

`- setScanLocation:` (page 20)
Sets the location at which the next scan operation will begin to a given index.

`- scanLocation` (page 16)
Returns the character position at which the receiver will begin its next scanning operation.

`- setCaseSensitive:` (page 19)
Sets whether the receiver is case sensitive when scanning characters.

`- caseSensitive` (page 8)
Returns a Boolean value that indicates whether the receiver distinguishes case in the characters it scans.

`- setCharactersToBeSkipped:` (page 19)
Sets the set of characters to ignore when scanning for a value representation.

`- charactersToBeSkipped` (page 9)
Returns a character set containing the characters the receiver ignores when looking for a scannable element.

`- setLocale:` (page 20)
Sets the receiver's locale to a given locale.

`- locale` (page 10)
Returns the receiver's locale.

Scanning a String

- `scanCharactersFromSet:intoString:` (page 11)
  Scans the string as long as characters from a given character set are encountered, accumulating characters into a string that's returned by reference.
- `scanUpToCharactersFromSet:intoString:` (page 17)
  Scans the string until a character from a given character set is encountered, accumulating characters into a string that's returned by reference.
- `scanDecimal:` (page 11)
  Scans for an `NSDecimal` value, returning a found value by reference.
- `scanDouble:` (page 12)
  Scans for a `double` value, returning a found value by reference.
- `scanFloat:` (page 12)
  Scans for a `float` value, returning a found value by reference.
- `scanHexDouble:` (page 13)
  Scans for a `double` value from a hexadecimal representation, returning a found value by reference.
- `scanHexFloat:` (page 14)
  Scans for a `double` value from a hexadecimal representation, returning a found value by reference.
- `scanHexInt:` (page 14)
  Scans for an `unsigned` value from a hexadecimal representation, returning a found value by reference.
- `scanHexLongLong:` (page 14)
  Scans for a `double` value from a hexadecimal representation, returning a found value by reference.
- `scanInteger:` (page 15)
  Scans for an `NSInteger` value from a decimal representation, returning a found value by reference
- `scanInt:` (page 15)
  Scans for an `int` value from a decimal representation, returning a found value by reference.
- `scanLongLong:` (page 16)
  Scans for a `long long` value from a decimal representation, returning a found value by reference.
- `scanString:intoString:` (page 17)
  Scans a given string, returning an equivalent string object by reference if a match is found.
- `scanUpToString:intoString:` (page 18)
  Scans the string until a given string is encountered, accumulating characters into a string that's returned by reference.
- `isAtEnd` (page 10)
  Returns a Boolean value that indicates whether the receiver has exhausted all significant characters

Class Methods

localizedScannerWithString:

Returns an `NSScanner` object that scans a given string according to the user's default locale.

```
+ (id)localizedScannerWithString:(NSString *)aString
```

Parameters

aString

The string to scan.

Return Value

An `NSScanner` object that scans *aString* according to the user's default locale.

Discussion

Sets the string to scan by invoking `initWithString:` (page 9) with *aString*. The locale is set with `setLocale:` (page 20).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSScanner.h`

scannerWithString:

Returns an `NSScanner` object that scans a given string.

```
+ (id)scannerWithString:(NSString *)aString
```

Parameters

aString

The string to scan.

Return Value

An `NSScanner` object that scans *aString*.

Discussion

Sets the string to scan by invoking `initWithString:` (page 9) with *aString*.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

iSpend

NumberInput_IMKit_Sample

QTAudioExtractionPanel

Quartz Composer QCTV

Sproing

Declared In

`NSScanner.h`

Instance Methods

caseSensitive

Returns a Boolean value that indicates whether the receiver distinguishes case in the characters it scans.

```
- (BOOL)caseSensitive
```

Return Value
`YES` if the receiver distinguishes case in the characters it scans, otherwise `NO`.

Discussion
Scanners are not case sensitive by default. Note that case sensitivity doesn't apply to the characters to be skipped.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setCaseSensitive:` (page 19)
`- setCharactersToBeSkipped:` (page 19)

Declared In
`NSScanner.h`

charactersToBeSkipped

Returns a character set containing the characters the receiver ignores when looking for a scannable element.

```
- (NSCharacterSet *)charactersToBeSkipped
```

Return Value
A character set containing the characters the receiver ignores when looking for a scannable element.

Discussion
For example, if a scanner ignores spaces and you send it a `scanInt:` (page 15) message, it skips spaces until it finds a decimal digit or other character. While an element is being scanned, however, no characters are skipped. If you scan for something made of characters in the set to be skipped (for example, using `scanInt:` (page 15) when the set of characters to be skipped is the decimal digits), the result is undefined.

The default set to skip is the whitespace and newline character set.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setCharactersToBeSkipped:` (page 19)
`whitespaceAndNewlineCharacterSet` (`NSCharacterSet`)

Declared In
`NSScanner.h`

initWithString:

Returns an `NSScanner` object initialized to scan a given string.

```
- (id)initWithString:(NSString *)aString
```

Parameters

aString

The string to scan.

Return Value

An `NSScanner` object initialized to scan *aString* from the beginning. The returned object might be different than the original receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

`+ localizedScannerWithString:` (page 7)

`+ scannerWithString:` (page 8)

Declared In

`NSScanner.h`

isAtEnd

Returns a Boolean value that indicates whether the receiver has exhausted all significant characters

```
- (BOOL)isAtEnd
```

Return Value

`YES` if the receiver has exhausted all significant characters in its string, otherwise `NO`.

If only characters from the set to be skipped remain, returns `YES`.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- charactersToBeSkipped` (page 9)

Related Sample Code

QTAudioExtractionPanel

Declared In

`NSScanner.h`

locale

Returns the receiver's locale.

```
- (id)locale
```

Return Value

The receiver's locale, or `nil` if it has none.

Discussion
A scanner's locale affects the way it interprets numeric values from the string. In particular, a scanner uses the locale's decimal separator to distinguish the integer and fractional parts of floating-point representations. A scanner with no locale set uses non-localized values.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setLocale:` (page 20)

Declared In
`NSScanner.h`

scanCharactersFromSet:intoString:

Scans the string as long as characters from a given character set are encountered, accumulating characters into a string that's returned by reference.

```
- (BOOL)scanCharactersFromSet:(NSCharacterSet *)scanSet intoString:(NSString
    **)stringValue
```

Parameters

scanSet
The set of characters to scan.

stringValue
Upon return, contains the characters scanned.

Return Value
`YES` if the receiver scanned any characters, otherwise `NO`.

Discussion
Invoke this method with `NULL` as *stringValue* to simply scan past a given set of characters.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- scanUpToCharactersFromSet:intoString:` (page 17)

Declared In
`NSScanner.h`

scanDecimal:

Scans for an `NSDecimal` value, returning a found value by reference.

```
- (BOOL)scanDecimal:(NSDecimal *)decimalValue
```

Parameters

decimalValue
Upon return, contains the scanned value. See the `NSDecimalNumber` class specification for more information about `NSDecimal` values.

Return Value
`YES` if the receiver finds a valid `NSDecimal` representation, otherwise `NO`.

Discussion
Invoke this method with `NULL` as *`decimalValue`* to simply scan past an `NSDecimal` representation.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
NumberInput_IMKit_Sample

Declared In
`NSDecimalNumber.h`

scanDouble:

Scans for a `double` value, returning a found value by reference.

```
- (BOOL)scanDouble:(double *)doubleValue
```

Parameters

`doubleValue`

Upon return, contains the scanned value. Contains `HUGE_VAL` or `-HUGE_VAL` on overflow, or `0.0` on underflow.

Return Value
`YES` if the receiver finds a valid floating-point representation, otherwise `NO`.

Discussion
Skips past excess digits in the case of overflow, so the scanner's position is past the entire floating-point representation.

Invoke this method with `NULL` as *`doubleValue`* to simply scan past a `double` value representation. Floating-point representations are assumed to be IEEE compliant.

Availability
Available in Mac OS X v10.0 and later.

See Also
`doubleValue` (`NSString`)

Declared In
`NSScanner.h`

scanFloat:

Scans for a `float` value, returning a found value by reference.

```
- (BOOL)scanFloat:(float *)floatValue
```

Parameters

floatValue

Upon return, contains the scanned value. Contains HUGE_VAL or -HUGE_VAL on overflow, or 0.0 on underflow.

Return Value

YES if the receiver finds a valid floating-point representation, otherwise NO.

Discussion

Skips past excess digits in the case of overflow, so the scanner's position is past the entire floating-point representation.

Invoke this method with NULL as *floatValue* to simply scan past a float value representation. Floating-point representations are assumed to be IEEE compliant.

Availability

Available in Mac OS X v10.0 and later.

See Also

floatValue (NSString)

Related Sample Code

iSpend

Quartz Composer QCTV

Declared In

NSScanner.h

scanHexDouble:

Scans for a double value from a hexadecimal representation, returning a found value by reference.

```
- (BOOL)scanHexDouble:(double *)result
```

Parameters

result

Upon return, contains the scanned value.

Return Value

YES if the receiver finds a valid double-point representation, otherwise NO.

Discussion

This corresponds to %a or %A formatting. The hexadecimal double representation must be preceded by 0x or 0X.

Invoke this method with NULL as *result* to simply scan past a hexadecimal double representation.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSScanner.h

scanHexFloat:

Scans for a `double` value from a hexadecimal representation, returning a found value by reference.

```
- (BOOL)scanHexFloat:(float *)result
```

Parameters

result

Upon return, contains the scanned value.

Return Value

`YES` if the receiver finds a valid float-point representation, otherwise `NO`.

Discussion

This corresponds to `%a` or `%A` formatting. The hexadecimal float representation must be preceded by `0x` or `0X`.

Invoke this method with `NULL` as *result* to simply scan past a hexadecimal float representation.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSScanner.h`

scanHexInt:

Scans for an `unsigned` value from a hexadecimal representation, returning a found value by reference.

```
- (BOOL)scanHexInt:(unsigned *)intValue
```

Parameters

intValue

Upon return, contains the scanned value. Contains `INT_MAX` or `INT_MIN` on overflow.

Return Value

Returns `YES` if the receiver finds a valid hexadecimal integer representation, otherwise `NO`.

Discussion

The hexadecimal integer representation may optionally be preceded by `0x` or `0X`. Skips past excess digits in the case of overflow, so the receiver's position is past the entire hexadecimal representation.

Invoke this method with `NULL` as *intValue* to simply scan past a hexadecimal integer representation.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSScanner.h`

scanHexLongLong:

Scans for a `double` value from a hexadecimal representation, returning a found value by reference.

```
- (BOOL)scanHexLongLong:(unsigned long long *)result
```

Parameters

result

Upon return, contains the scanned value.

Return Value

YES if the receiver finds a valid double-point representation, otherwise NO.

Discussion

Invoke this method with NULL as *result* to simply scan past a hexadecimal long long representation.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSScanner.h

scanInt:

Scans for an int value from a decimal representation, returning a found value by reference.

```
- (BOOL)scanInt:(int *)intValue
```

Parameters

intValue

Upon return, contains the scanned value. Contains INT_MAX or INT_MIN on overflow.

Return Value

YES if the receiver finds a valid decimal integer representation, otherwise NO.

Discussion

Skips past excess digits in the case of overflow, so the receiver's position is past the entire decimal representation.

Invoke this method with NULL as *intValue* to simply scan past a decimal integer representation.

Availability

Available in Mac OS X v10.0 and later.

See Also

intValue (NSString)

- scanInteger: (page 15)

Declared In

NSScanner.h

scanInteger:

Scans for an NSInteger value from a decimal representation, returning a found value by reference

```
- (BOOL)scanInteger:(NSInteger *)value
```

Parameters

value

Upon return, contains the scanned value.

Return Value
`YES` if the receiver finds a valid integer representation, otherwise `NO`.

Discussion
Skips past excess digits in the case of overflow, so the receiver's position is past the entire integer representation.

Invoke this method with `NULL` as *`value`* to simply scan past a decimal integer representation.

Availability
Available in Mac OS X v10.5 and later.

See Also
`integerValue` (`NSString`)
`- scanInt:` (page 15)

Declared In
`NSScanner.h`

scanLocation

Returns the character position at which the receiver will begin its next scanning operation.

```
- (NSUInteger)scanLocation
```

Return Value
The character position at which the receiver will begin its next scanning operation.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setScanLocation:` (page 20)

Declared In
`NSScanner.h`

scanLongLong:

Scans for a `long long` value from a decimal representation, returning a found value by reference.

```
- (BOOL)scanLongLong:(long long *)longLongValue
```

Parameters
`longLongValue`
Upon return, contains the scanned value. Contains `LLONG_MAX` or `LLONG_MIN` on overflow.

Return Value
`YES` if the receiver finds a valid decimal integer representation, otherwise `NO`.

Discussion
All overflow digits are skipped. Skips past excess digits in the case of overflow, so the receiver's position is past the entire decimal representation.

Invoke this method with `NULL` as *`longLongValue`* to simply scan past a long decimal integer representation.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSScanner.h`

scanString:intoString:

Scans a given string, returning an equivalent string object by reference if a match is found.

```
- (BOOL)scanString:(NSString *)string intoString:(NSString **)stringValue
```

Parameters

`string`

The string for which to scan at the current scan location.

`stringValue`

Upon return, if the receiver contains a string equivalent to *`string`* at the current scan location, contains a string equivalent to *`string`*.

Return Value
`YES` if *`stringValue`* matches the characters at the scan location, otherwise `NO`.

Discussion
If *`string`* is present at the current scan location, then the current scan location is advanced to after the string; otherwise the scan location does not change.

Invoke this method with `NULL` as *`stringValue`* to simply scan past a given string.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- scanUpToString:intoString:` (page 18)

Declared In
`NSScanner.h`

scanUpToCharactersFromSet:intoString:

Scans the string until a character from a given character set is encountered, accumulating characters into a string that's returned by reference.

```
- (BOOL)scanUpToCharactersFromSet:(NSCharacterSet *)stopSet intoString:(NSString
    **)stringValue
```

Parameters

`stopSet`

The set of characters up to which to scan.

`stringValue`

Upon return, contains the characters scanned.

Return Value
`YES` if the receiver scanned any characters, otherwise `NO`.

If the only scanned characters are in the `charactersToBeSkipped` (page 9) character set (which is the whitespace and newline character set by default), then returns `NO`.

Discussion
Invoke this method with `NULL` as *`stringValue`* to simply scan up to a given set of characters.

If no characters in *`stopSet`* are present in the scanner's source string, the remainder of the source string is put into *`stringValue`*, the receiver's `scanLocation` is advanced to the end of the source string, and the method returns `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- scanCharactersFromSet:intoString:` (page 11)

Declared In
`NSScanner.h`

scanUpToString:intoString:

Scans the string until a given string is encountered, accumulating characters into a string that's returned by reference.

```
- (BOOL)scanUpToString:(NSString *)stopString intoString:(NSString **)stringValue
```

Parameters
`stopString`
The string to scan up to.

`stringValue`
Upon return, contains any characters that were scanned.

Return Value
`YES` if the receiver scans any characters, otherwise `NO`.

If the only scanned characters are in the `charactersToBeSkipped` (page 9) character set (which by default is the whitespace and newline character set), then this method returns `NO`.

Discussion
If `stopString` is present in the receiver, then on return the scan location is set to the beginning of that string.

If `stopString` is the first string in the receiver, then the method returns `NO` and *`stringValue`* is not changed.

If the search string (`stopString`) isn't present in the scanner's source string, the remainder of the source string is put into *`stringValue`*, the receiver's `scanLocation` is advanced to the end of the source string, and the method returns `YES`.

Invoke this method with `NULL` as *`stringValue`* to simply scan up to a given string.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `scanString:intoString:` (page 17)

Declared In
`NSScanner.h`

setCaseSensitive:

Sets whether the receiver is case sensitive when scanning characters.

```
- (void)setCaseSensitive:(BOOL)flag
```

Parameters

flag

If `YES`, the receiver will distinguish case when scanning characters, otherwise it will ignore case distinctions.

Discussion
Scanners are not case sensitive by default. Note that case sensitivity doesn't apply to the characters to be skipped.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `caseSensitive` (page 8)
- `setCharactersToBeSkipped:` (page 19)

Declared In
`NSScanner.h`

setCharactersToBeSkipped:

Sets the set of characters to ignore when scanning for a value representation.

```
- (void)setCharactersToBeSkipped:(NSCharacterSet *)skipSet
```

Parameters

skipSet

The characters to ignore when scanning for a value representation.

Discussion
For example, if a scanner ignores spaces and you send it a `scanInt:` (page 15) message, it skips spaces until it finds a decimal digit or other character. While an element is being scanned, however, no characters are skipped. If you scan for something made of characters in the set to be skipped (for example, using `scanInt:` (page 15) when the set of characters to be skipped is the decimal digits), the result is undefined.

The characters to be skipped are treated literally as single values. A scanner doesn't apply its case sensitivity setting to these characters and doesn't attempt to match composed character sequences with anything in the set of characters to be skipped (though it does match pre-composed characters individually). If you want to skip all vowels while scanning a string, for example, you can set the characters to be skipped to those in the string "AEIOUaeiou" (plus any accented variants with pre-composed characters).

The default set of characters to skip is the whitespace and newline character set.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- charactersToBeSkipped` (page 9)
`whitespaceAndNewlineCharacterSet` (`NSCharacterSet`)

Related Sample Code
ImageMapExample
QTAudioExtractionPanel
Quartz Composer QCTV

Declared In
`NSScanner.h`

setLocale:

Sets the receiver's locale to a given locale.

```
- (void)setLocale:(id)aLocale
```

Parameters
aLocale
The locale for the receiver.

Discussion
A scanner's locale affects the way it interprets values from the string. In particular, a scanner uses the locale's decimal separator to distinguish the integer and fractional parts of floating-point representations. A new scanner's locale is by default `nil`, which causes it to use non-localized values.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- locale` (page 10)

Declared In
`NSScanner.h`

setScanLocation:

Sets the location at which the next scan operation will begin to a given index.

```
- (void)setScanLocation:(NSUInteger)index
```

Parameters
index
The location at which the next scan operation will begin. Raises an `NSRangeException` if *index* is beyond the end of the string being scanned.

Discussion
This method is useful for backing up to rescan after an error.

Rather than setting the scan location directly to skip known sequences of characters, use `scanString:intoString:` (page 17) or `scanCharactersFromSet:intoString:` (page 11), which allow you to verify that the expected substring (or set of characters) is in fact present.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- scanLocation` (page 16)

Declared In
`NSScanner.h`

string

Returns the string with which the receiver was created or initialized.

```
- (NSString *)string
```

Return Value
The string with which the receiver was created or initialized.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- locale` (page 10)

Declared In
`NSScanner.h`

Document Revision History

This table describes the changes to *NSScanner Class Reference*.

Date	Notes
2008-10-15	Documented hexadecimal scanning methods introduced in Mac OS X v10.5
2007-02-22	Included API introduced in Mac OS X v10.5.
	Added a note that `[[scanner string]substringFromIndex:[scanner scanLocation]]` is the best way to retrieve the unscanned remainder of the string.
2006-06-28	Clarified the behavior of scanUpToString:intoString:, scanUpToCharactersFromSet:intoString:, and scanString:intoString:.
2006-05-23	Corrected typographical errors.
	First publication of this content as a separate document.

Document Revision History

Index

C

caseSensitive instance method 8
charactersToBeSkipped instance method 9

I

initWithString: instance method 9
isAtEnd instance method 10

L

locale instance method 10
localizedScannerWithString: class method 7

S

scanCharactersFromSet:intoString: instance method 11
scanDecimal: instance method 11
scanDouble: instance method 12
scanFloat: instance method 12
scanHexDouble: instance method 13
scanHexFloat: instance method 14
scanHexInt: instance method 14
scanHexLongLong: instance method 14
scanInt: instance method 15
scanInteger: instance method 15
scanLocation instance method 16
scanLongLong: instance method 16
scannerWithString: class method 8
scanString:intoString: instance method 17
scanUpToCharactersFromSet:intoString: instance method 17
scanUpToString:intoString: instance method 18
setCaseSensitive: instance method 19
setCharactersToBeSkipped: instance method 19
setLocale: instance method 20
setScanLocation: instance method 20
string instance method 21