1-Length Method

Returns the length of this character sequence. The length is the number of 16-bit chars in the sequence.

Syntax: int length ()

Returns the length of this character sequence. The length is the number of 16-bit chars in the sequence.

Returns:

The number of chars in this sequence

Syntax

```
public int length()
```

Parameters

None.

Technical Details

Returns:	An `int` value, representing the length of the string

2-charAt

Char charAt(int index)

Returns the char value at the specified index. An index ranges from zero to length() - 1. The first char value of the sequence is at index zero, the next at index one, and so on, as for array indexing.

If the char value specified by the index is a surrogate, the surrogate value is returned.

Parameters:

Index - the index of the char value to be returned

Returns:

The specified char value

Throws:

IndexOutOfBoundsException - if the index argument is negative or not less than length()

Syntax

```
public char charAt(int index)
```

Parameter Values

Parameter	Description
index	An `int` value representing the index of the character to return

Technical Details

Returns:	A `char` value at the specified index of this string. The first char value is at index 0
Throws:	`IndexOutOfBoundsException` - if index is negative or not less than the length of the specified string

3-subSequence

Char Sequence subsequence(int start, int end)

Returns a Char Sequence that is a subsequence of this sequence. The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.

Parameters:

Start - the start index, inclusive

End - the end index, exclusive

Returns:

The specified subsequence

Throws:

IndexOutOfBoundsException - if start or end are negative, if end is greater than length(), or if start is greater than end

Syntax:

```
subSequence(int beginIndex, int endIndex)
```

Parameters:

Name	Description
beginIndex	the beginning index, inclusive.
endIndex	the end index, exclusive.

Return Value: the specified subsequence.

Throws:
IndexOutOfBoundsException - if beginIndex or endIndex is negative, if endIndex is greater than length(), or if beginIndex is greater than endIndex

4-toString

String toString()

Returns a string containing the characters in this sequence in the same order as this sequence. The length of the string will be the length of this sequence.

Overrides:

to String in class Object

Returns:

A string consisting of exactly this sequence of characters

Syntax

```
string.toString()
```

Parameters

NONE

Return Value

Type	Description
A string	The content of the string.

5-default IntStream chars()

Returns a stream of int zero extending the char values from this sequence. Any char, which maps to a surrogate code point, is passed through uninterpreted.

The stream binds to this sequence when the terminal stream operation commences (specifically, for mutable sequences the spliterator for the stream is late binding). If the sequence is modified during that operation then the result is undefined.

Returns:

an IntStream of char values from this sequence

```
default IntStream chars()
```

Returns a stream of int zero-extending the char values from this sequence. Any char which maps to a surrogate code point is passed through uninterpreted.

The stream binds to this sequence when the terminal stream operation commences (specifically, for mutable sequences the spliterator for the stream is *late-binding*). If the sequence is modified during that operation then the result is undefined.

Returns:
an IntStream of char values from this sequence

Since:
1.8

6- default IntStream codePoints()

Returns a stream of code point values from this sequence. Any surrogate pairs encountered in the sequence are combined as if by Character.toCodePoint and the result is passed to the stream. Any other code units, including ordinary BMP characters, unpaired surrogates, and undefined code units, are zero-extended to int values which are then passed to the stream.

The stream binds to this sequence when the terminal stream operation commences (specifically, for mutable sequences the spliterator for the stream is late-binding). If the sequence is modified during that operation then the result is undefined.

Returns:

an IntStream of Unicode code points from this sequence

The **codePoints()** method of **IntStream** class is used to get a stream of code point values from the given sequence. It returns the ASCII values of the characters passed as an argument

Syntax:

```
public IntStream codePoints()
```

Return Value: This method returns the **IntStream code values**

Below examples illustrate the use of codePoints() method:

7-static int compare(CharSequence cs1, CharSequence cs2)

Compares two CharSequence instances lexicographically. Returns a negative value, zero, or a positive value if the first sequence is lexicographically less than, equal to, or greater than the second, respectively.

The lexicographical ordering of CharSequence is defined as follows. Consider a CharSequence cs of length len to be a sequence of char values, cs[0] to cs[len-1]. Suppose k is the lowest index at which the corresponding char values from each sequence differ. The lexicographic ordering of the sequences is determined by a numeric comparison of the char values cs1[k] with cs2[k]. If there is no such index k, the shorter sequence is considered lexicographically less than the other. If the sequences have the same length, the sequences are considered lexicographically equal.

Parameters:

cs1 - the first CharSequence

cs2 - the second CharSequence

Returns:

the value 0 if the two CharSequence are equal; a negative integer if the first CharSequence is lexicographically less than the second; or a positive integer if the first CharSequence is lexicographically greater than the second.

Parameter Values

Parameter	Description
string2	A `String`, representing the other string to be compared
object	An `Object`, representing an object to be compared

Technical Details

Returns:	An `int` value: 0 if the string is equal to the other string. < 0 if the string is lexicographically less than the other string > 0 if the string is lexicographically greater than the other string (more characters)