2.2 The *String* Class

- A string
  - » is a sequence of characters
  - » no[redacted]type for string in Java
  - » String Class
- The String class
  - » used to store strings
  - » The String class has [redacted] to operate on strings

String constant

- String constant
  - » one or more characters in [redacted] quotes
- Examples:

```
char charVariable = `a`;//single quotes
String stringVariable = "a";//double quotes
String sentence = "Hello, world";
```

.

.

String Variables

- Declare a String variable:

```
String greeting;
```

- Assign a value to the variable

```
greeting = "Hello!";
```

- Use the variable as a String argument in a method:

```
System.out.println(greeting);
```

causes the string **`Hello!`** to be displayed on the screen

[redacted] (Appending) Strings

Stringing together strings - the “+” operator for Strings:

```
String name = "Mondo";
String greeting = "Hi, there!";
System.out.println(greeting + name + "Welcome");
```

causes the following to display on the screen:

```
>Hi, there!MondoWelcome
>
```

Note that you have to remember to include spaces if you want it to look right:

```
System.out.println(greeting + " " + name
                             + " Welcome");
```

causes the following to display on the screen:

```
>Hi, there! Mondo Welcome
>
```

Classes

- A Class
  - » a [redacted] whose values are objects
- Objects
  - » entities that store [redacted] and can take [redacted]
- Methods
  - » the actions that an object can take

`charAt`(*Index*)	Returns the character at *Index* in this string. Index numbers begin at 0.
`compareTo`(*A_String*)	Compares this string with *A_String* to see which string comes first in the lexicographic ordering. (Lexicographic ordering is the same as alphabetical ordering when both strings are either all uppercase letters or all lowercase letters.) Returns a negative integer if this string is first, returns zero if the two strings are equal, and returns a positive integer if *A_String* is first.
`concat`(*A_String*)	Returns a new string having the same characters as this string concatenated with the characters in *A_String*. You can use the ⇓ operator instead of `concat`.
`equals`(*Other_String*)	Returns true if this string and *Other_String* are equal. Otherwise, returns false.
`equalsIgnoreCase`(*Other_String*)	Behaves like the method `equals`, but considers uppercase and lowercase versions of a letter to be the same.
`indexOf`(*A_String*)	Returns the index of the first occurrence of the substring *A_String* within this string. Returns -1 if *A_String* is not found. Index numbers begin at 0.
`lastIndexOf`(*A_String*)	Returns the index of the last occurrence of the substring *A_String* within this string. Returns -1 if *A_String* is not found. Index numbers begin at 0.

`length()`

Returns the length of this string.

`toLowerCase()`

Returns a new string having the same characters as this string, but with any uppercas
letters converted to lowercase.

`toUpperCase()`

Returns a new string having the same characters as this string, but with any lowercas
letters converted to uppercase.

`replace(`*OldChar, NewChar*`)`

Returns a new string having the same characters as this string, but with each occurrenc
of *OldChar* replaced by *NewChar*.

`substring(`*Start*`)`

Returns a new string having the same characters as the substring that begins at inde
Start of this string through to the end of the string. Index numbers begin at 0.

`substring(`*Start, End*`)`

Returns a new string having the same characters as the substring that begins at inde
Start of this string through, but not including, index *End* of the string. Index number
begin at 0.

`trim()`

Returns a new string having the same characters as this string, but with leading an
trailing whitespace removed.

Length of String

- Int n = “Hello” [redacted]
  - » returns 5 integer
- String greeting = “Hello”;
- Int n = [redacted]

- String Methods
  - » Figure 2.5 . Methods in the Class String

Indexing Characters within a String

- The index of a character within a string is an integer starting at 0 for the first character and gives the position of the character
- The [redacted]*`(Position)`* method returns the char at the specified position
- [redacted]*`(Start, End)`* method returns the string from position ***`Start`*** to position [redacted]
- For example:

`String greeting = "Hi, there!";`

`greeting.charAt(0)` returns **`H`**

`greeting.charAt(2)` returns **`,`**

`greeting.substring(4,6)` returns

H	i	,		t	h	e	r	e	!
0	1	2	3	4	5	6	7	8	9

[redacted]

- [redacted]
  - » return the index of the substring given as its one argument.
  - » if the substring occurs more than once, indexOf returns the [redacted] of the first occurrence of its substring argument.
- Ex) String phrase = “Time flies like an arrow.”;

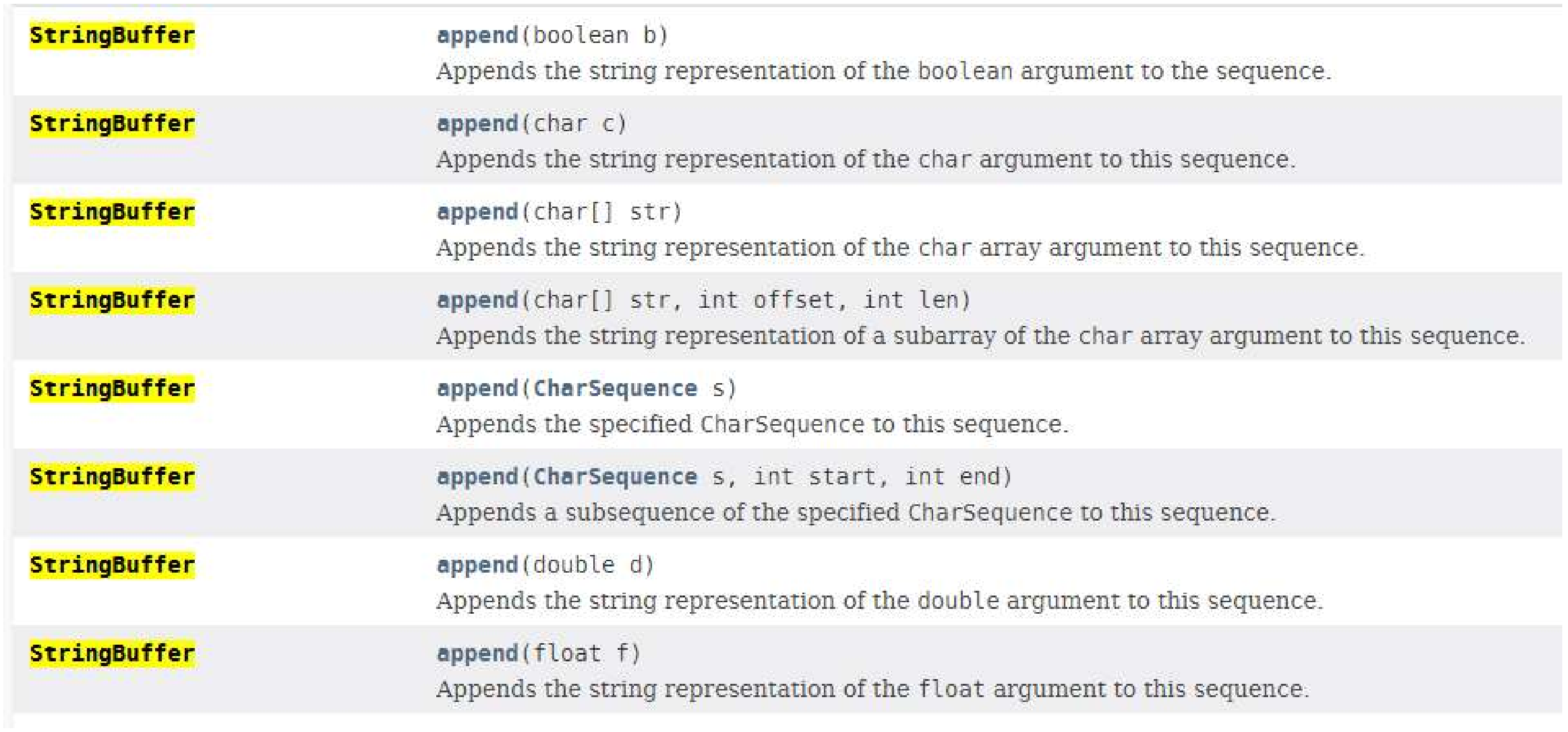
  phrase.indexOf(“flies”) will return

String Processing

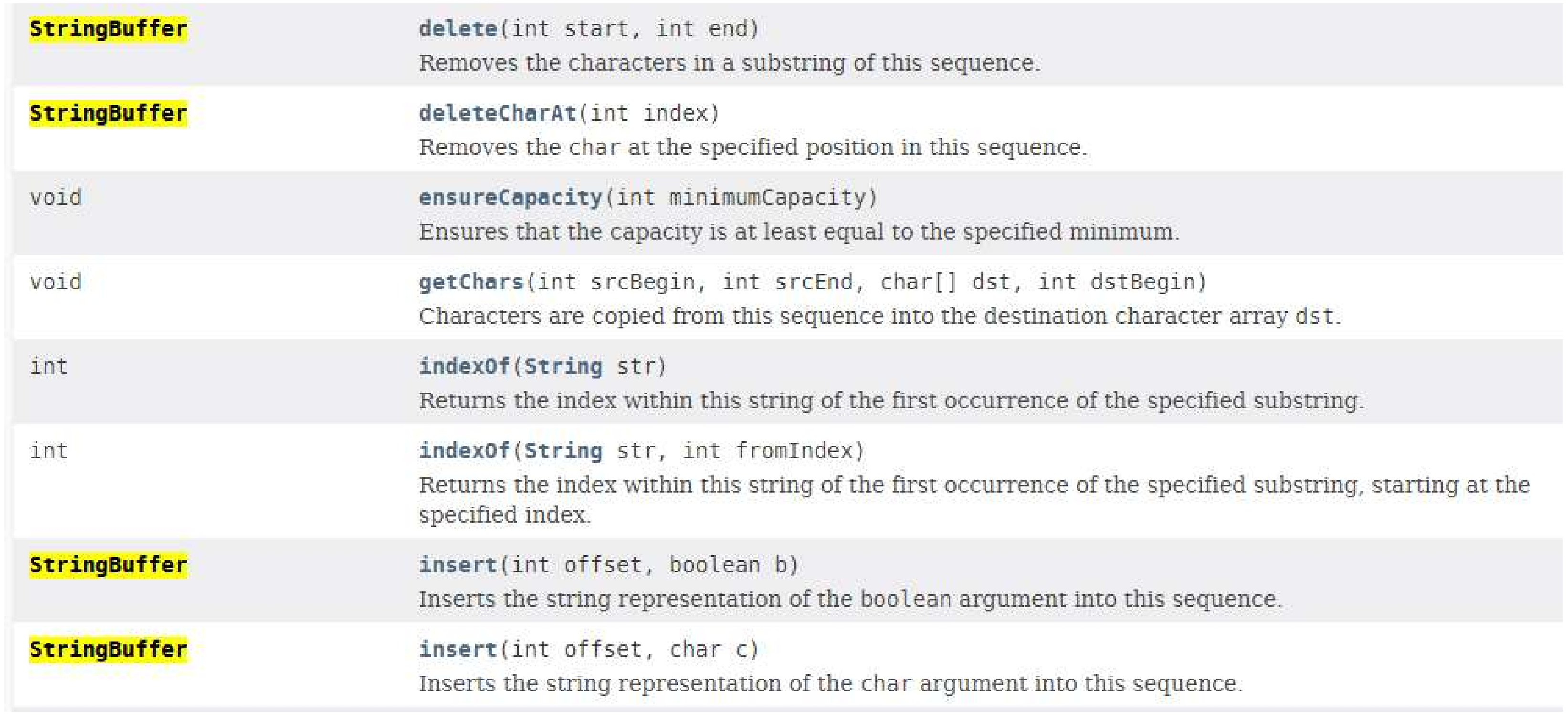
- String Class
  - » objects of type String [redacted] changed.
- The StringBuffer Class
  - » has methods for altering the string object.

    [redacted] – insert, delete, append etc.
- You can still write programs that change the value of a String variable.
  - » ex) String name = “D’Aargo”;

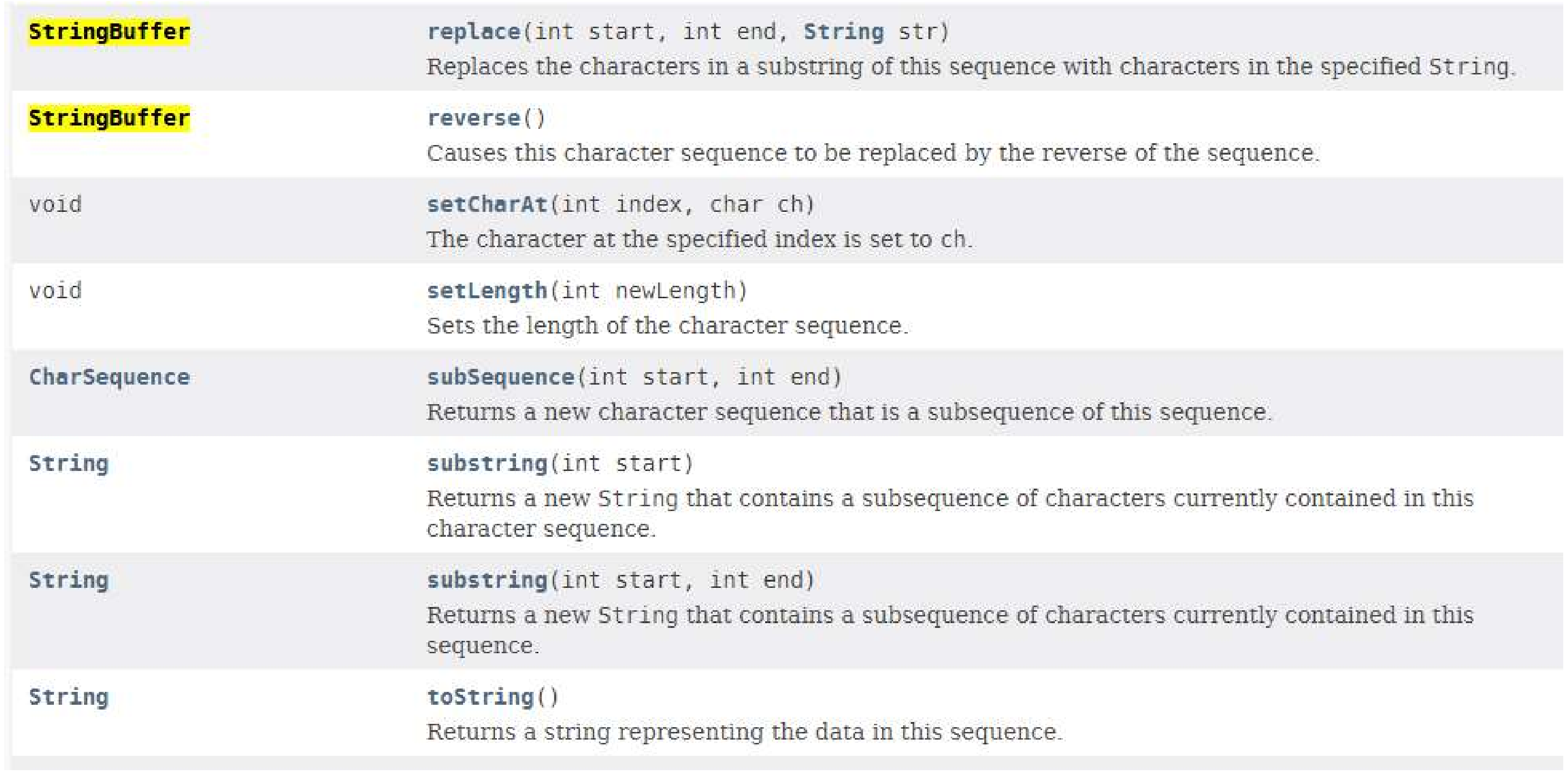
    name = “Ka “ + name ;



StringBuffer	append(boolean b) Appends the string representation of the boolean argument to the sequence.
StringBuffer	append(char c) Appends the string representation of the char argument to this sequence.
StringBuffer	append(char[] str) Appends the string representation of the char array argument to this sequence.
StringBuffer	append(char[] str, int offset, int len) Appends the string representation of a subarray of the char array argument to this sequence.
StringBuffer	append(CharSequence s) Appends the specified CharSequence to this sequence.
StringBuffer	append(CharSequence s, int start, int end) Appends a subsequence of the specified CharSequence to this sequence.
StringBuffer	append(double d) Appends the string representation of the double argument to this sequence.
StringBuffer	append(float f) Appends the string representation of the float argument to this sequence.



`StringBuffer`	`delete(int start, int end)` Removes the characters in a substring of this sequence.
`StringBuffer`	`deleteCharAt(int index)` Removes the `char` at the specified position in this sequence.
`void`	`ensureCapacity(int minimumCapacity)` Ensures that the capacity is at least equal to the specified minimum.
`void`	`getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)` Characters are copied from this sequence into the destination character array `dst`.
`int`	`indexOf(String str)` Returns the index within this string of the first occurrence of the specified substring.
`int`	`indexOf(String str, int fromIndex)` Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
`StringBuffer`	`insert(int offset, boolean b)` Inserts the string representation of the `boolean` argument into this sequence.
`StringBuffer`	`insert(int offset, char c)` Inserts the string representation of the `char` argument into this sequence.



`StringBuffer`	`replace(int start, int end, String str)` Replaces the characters in a substring of this sequence with characters in the specified `String`.
`StringBuffer`	`reverse()` Causes this character sequence to be replaced by the reverse of the sequence.
`void`	`setCharAt(int index, char ch)` The character at the specified index is set to `ch`.
`void`	`setLength(int newLength)` Sets the length of the character sequence.
`CharSequence`	`subSequence(int start, int end)` Returns a new character sequence that is a subsequence of this sequence.
`String`	`substring(int start)` Returns a new `String` that contains a subsequence of characters currently contained in this character sequence.
`String`	`substring(int start, int end)` Returns a new `String` that contains a subsequence of characters currently contained in this sequence.
`String`	`toString()` Returns a string representing the data in this sequence.

(Not) Changing `String` Objects

- No [redacted] allow you to change the value of a `String` object.
  - » But you can change the value of a String variable.
  - » C.f. Methods of `String` Object, `StringBuffer` Object

	<u>value of pause</u>
`String pause = "    Hmm    ";`	**Hmm**
`pause = pause.trim()`	**Hmm**
`pause = pause + "mmm!";`	**Hmmmmm!**
`pause = "Ahhh";`	**Ahhh**

Escape Characters

- How do you print characters that have special meaning? For example, how do you print the following string?

  `The word "hard"`

  Would this do it?

  ```
  System.out.println("The word "hard"");
  ```

  No, it would give a compiler error - it sees the string **`The word`** between the first set of double quotes and is confused by what comes after

- Use the backslash character, “\”, to escape the special meaning of the internal double quotes:

  ```
  System.out.println("The word \"hard\""); //this works
  ```

More Escape Characters

Use the following escape characters to include the character listed in a quoted string:

`\"` Double quote.

`\'` Single quote.

`\\` Backslash.

`\n` New line. Go to the beginning of the next line.

`\r` carriage return. Go to the beginning of the current line.

`\t` Tab. White space up to the next tab stop.

Examples

```
System.out.println("abc\\def");
```

```
System.out.println("new\nline");
```

```
char singleQuote = '\'';
System.out.println(singleQuote);
```

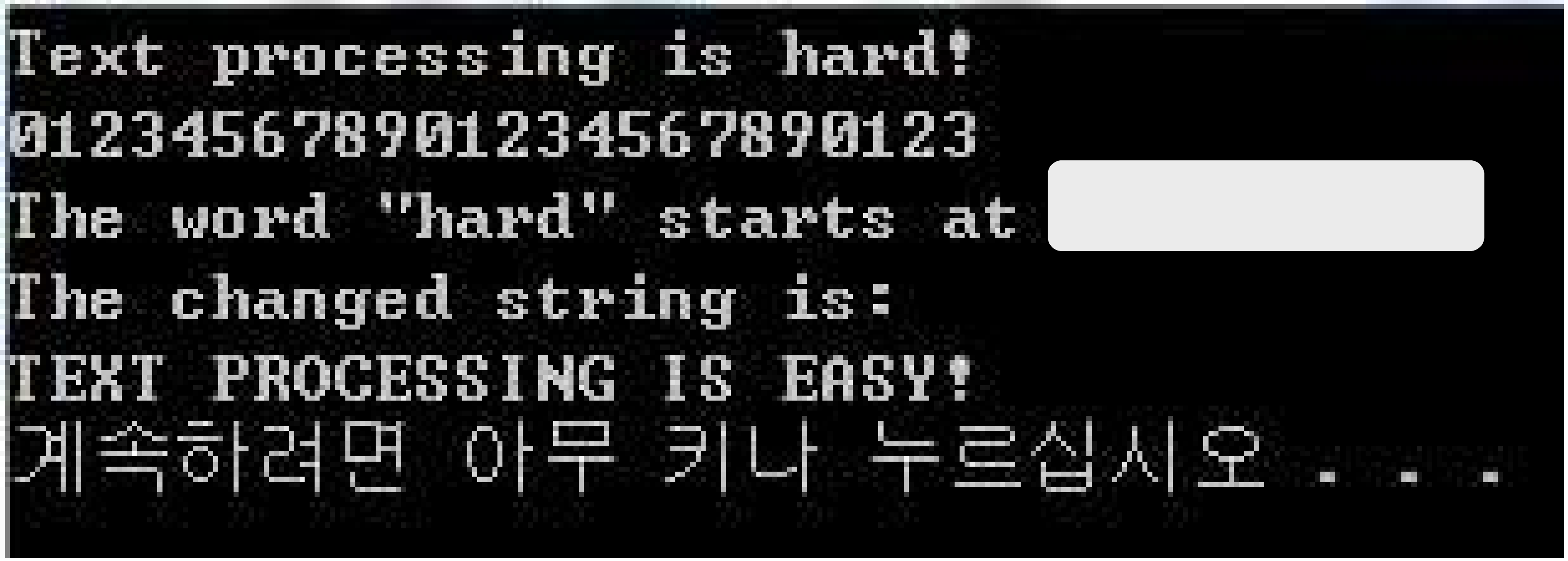
```java
// Listing 2.4 Using the String Class
//
public class StringDemo
{
    public static void main(String[] args)
    {
        String sentence = "Text processing is hard!";
        int position = sentence.indexOf("hard");

        System.out.println(sentence);
        System.out.println("012345678901234567890123");
        System.out.println("The word \"hard\" starts at index "
                        + position);

        sentence = sentence.substring(0, position) + "easy!";
        sentence = sentence.toUpperCase();

        System.out.println("The changed string is:");
        System.out.println(sentence);
    }
}
```

Listing 2.4
Using the String Class



```
Text processing is hard!
012345678901234567890123
The word "hard" starts at
The changed string is:
TEXT PROCESSING IS EASY!
계속하려면 아무 키나 누르십시오 . . .
```

The Unicode Character Set

- Most programming languages use the [redacted] character set.
- Java uses the [redacted] character set which includes the ASCII character set.
- The Unicode character set includes characters from many different alphabets (but you probably won’t use them).