I'll put the material you see in these tables to use in the next few topics, in which we'll create and use String objects.

Table 2.2 String class field summary.

Field	Means
static Comparator CASE_INSENSITIVE_ORDER	Yields a comparator (which you'll see more about later) that orders **String** objects, as in **compareToIgnoreCase**.

Table 2.3 String class constructor summary.

Constructor	Means
String()	Initialises a new **String** object so that it holds an empty character sequence.
String(byte[] bytes)	Constructs a new **String** object by converting the array of bytes using the platform's default character encoding.
String(byte[] ascii, int hibyte)	Deprecated. This method does not properly convert bytes into characters.
String(byte[] bytes, int offset, int length)	Constructs a new **String** object by converting the subarray of bytes using the default character encoding.
String(byte[] ascii, int hibyte, int offset, int count)	Deprecated. This method does not properly convert bytes into characters.
String(byte[] bytes, int offset, int length, String enc)	Constructs a new **String** object by converting the subarray of bytes using the specified character encoding.
String(byte[] bytes, String enc)	Constructs a new **String** object by converting the array of bytes using the specified character encoding.

Table 2.3 String class constructor summary.

Constructor	Means
String(char[] value)	Allocates a new **String** object so that it represents the sequence of characters contained in the character array argument.
String(char[] value, int offset, int count)	Allocates a new **String** object that contains characters from a subarray of the character array argument.
String(String value)	Initialises a new **String** object so that it represents the same sequence of characters as the argument string.
String(StringBuffer buffer)	Allocates a new **String** object that contains the sequence of characters contained in the string buffer argument.

Table 2.4 String class methods.

Method	Means
char charAt(int index)	Yields the character at the given index.
int compareTo(Object o)	Compares this **String** object to another object
int compareTo(String anotherString)	Compares two strings lexicographically.
int compareToIgnoreCase(String str)	Compares two strings lexicographically, ignoring case.
String concat(String str)	Concatenates the given string to the end of this string.
Static String copyValueOf(char[] data)	Yields a **String** object that's equivalent to the given character array.
static String copyValueOf(char[] data, int offset, int count)	Yields a **String** object that's equivalent to the given character array, using offsets.
boolean endsWith(String suffix)	True if this string ends with the given suffix.
boolean equals(Object anObject)	Compares this string to an object.
boolean equalsIgnoreCase(String anotherString)	Compares this **String** object to another String object, ignoring case.
byte[] getBytes()	Converts this **String** object into bytes according to the default character encoding, storing the result in a new byte array.
void getBytes(int srcBegin, int srcEnd, byte[] dst, int dstBegin)	Deprecated. This method does not properly convert characters into bytes.
byte[] getBytes(String enc)	Converts this **String** object into bytes according to the given character encoding, storing the result in a new byte array.
void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)	Copies characters from this string into the destination array.
int hashCode()	Yields a hashcode for this string.
int indexOf(int ch)	Yields the index within this string of the first occurrence of the given character.
int indexOf(int ch, int fromIndex)	Yields the index within this string of the first occurrence of the given character, starting at the given index.

Table 2.4 String class methods.

Method	Means
int indexOf(String str)	Yields the index within this string of the first occurrence of the given substring.
int indexOf(String str, int fromIndex)	Yields the index within this string of the first occurrence of the given substring, starting at the given index.
String intern()	Yields a representation for the **String** object.
int lastIndexOf(int ch)	Yields the index within this string of the last occurrence of the given character.
int lastIndexOf(int ch, int fromIndex)	Yields the index within this string of the last occurrence of the given character, searching backward from the given index.
int lastIndexOf(String str)	Yields the index within this string of the rightmost occurrence of the given substring.
int lastIndexOf(String str, int fromIndex)	Yields the index within this string of the last occurrence of the given substring.
int length()	Yields the length of this string.
boolean regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len)	Tests whether two string regions are equal, allowing you to ignore case.
boolean regionMatches(int toffset, String other, int ooffset, int len)	Tests whether two string regions are equal.
String replace(char oldChar, char newChar)	Yields a new string by replacing all occurrences of **oldChar** in this string with **newChar**.
boolean startsWith(String prefix)	Tests whether this string starts with the given prefix.
boolean startsWith(String prefix, int toffset)	Tests whether this string starts with the given prefix, beginning at the given index.
String substring(int beginIndex)	Yields a new string that's a substring of this string.
String substring(int beginIndex, int endIndex)	Yields a new string that's a substring of this string, allowing you to specify the end index.
char[] toCharArray()	Converts this string to a new character array.
String toLowerCase()	Converts all the characters in this String object to lowercase using the rules of the default locale, which is returned by **Locale.getDefault**.
String toLowerCase(Locale locale)	Converts all the characters in this **String** object to lowercase using the rules of the given locale.
String toString()	This object (which is already a string) is returned.
String toUpperCase()	Converts all the characters in this **String** object to uppercase using the rules of the default locale, which is returned by **Locale.getDefault**.
String toUpperCase(Locale locale)	Converts all the characters in this String object to uppercase using the rules of the given locale.
String trim()	Removes white space from both ends of this string.

Table 2.4 String class methods.

Method	Means
static String valueOf(boolean b)	Yields the string representation of the **boolean** argument.
static String valueOf(char c)	Yields the string representation of the char argument.
static String valueOf(char[] data)	Yields the string representation of the **char** array argument.
static String valueOf(char[ ] data, int offset, int count)	Yields the string representation of a specific subarray of the **char** array argument.
static String valueOf(double d)	Yields the string representation of a **double**.
static String valueOf(float f)	Yields the string representation of a **float**.
static String valueOf(int i)	Yields the string representation of an **int**.
static String valueOf(long l)	Yields the string representation of a **long**.
static String valueOf(Object obj)	Yields the string representation of an object.

Table 2.5 StringBuffer class constructors.

Constructors	Means
StringBuffer()	Constructs a string buffer with no characters in it and a capacity of 16 characters.
StringBuffer(int length)	Constructs a string buffer with no characters in it and a capacity as given by the length argument.
StringBuffer(String str)	Constructs a string buffer so that it represents the same sequence of characters as the argument string.

Table 2.6 StringBuffer class methods.

Method	Means
StringBuffer append(boolean b)	Appends the string representation of the **boolean** argument to the string buffer.
StringBuffer append(char c)	Appends the string representation of the **char** argument to the string buffer.
StringBuffer append(char[] str)	Appends the string representation of the **char** array argument to the string buffer.
StringBuffer append(char[] str, int offset, int len)	Appends the string representation of a subarray of the char array argument to the string buffer.

Table 2.6 StringBuffer class methods.

Method	Means
StringBuffer append(double d)	Appends the string representation of the **double** argument to the string buffer.
StringBuffer append(float f)	Appends the string representation of the **float** argument to the string buffer.
StringBuffer append(int i)	Appends the string representation of the **int** argument to the string buffer.
StringBuffer append(long l)	Appends the string representation of the **long** argument to the string buffer.
StringBuffer append(Object obj)	Appends the string representation of the **Object** argument to the string buffer.
StringBuffer append(String str)	Appends the string to the string buffer.
int capacity()	Yields the capacity of the string buffer.
char charAt(int index)	Yields the given character of the sequence represented by the string buffer, as indicated by the **index** argument.
StringBuffer delete(int start, int end)	Removes the characters in a substring of this string buffer.
StringBuffer deleteCharAt(int index)	Removes the character at the given position in this stringbuffer, shortening the string buffer by one character.
void ensureCapacity(int minimumCapacity)	Ensures that the capacity of the buffer is at least equal to the given minimum.
void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)	Characters are copied from this string buffer into the destination character array.
StringBuffer insert(int offset, boolean b)	Inserts the string representation of the **boolean** argument into the string buffer.
StringBuffer insert(int offset, char c)	Inserts the string representation of the **char** argument into the string buffer.
StringBuffer insert(int offset, char[ ] str)	Inserts the string representation of the **char** array argument into the string buffer.
StringBuffer insert(int index, char[ ] str, int offset, int len)	Inserts the string representation of a subarray of the **str** array argument into the string buffer.
StringBuffer insert(int offset, double d)	Inserts the string representation of the **double** argument into the string buffer.
StringBuffer insert(int offset, float f)	Inserts the string representation of the **float** argument into the string buffer.
StringBuffer insert(int offset, int i)	Inserts the string representation of the second **int** argument into the string buffer.
StringBuffer insert(int offset, long l)	Inserts the string representation of the **long** argument into the string buffer.
StringBuffer insert(int offset, Object obj)	Inserts the string representation of the **Object** argument into the string buffer.
StringBuffer insert(int offset, String str)	Inserts the string into the string buffer.

Table 2.6 StringBuffer class methods.

Method	Means
int length()	Yields the length (in characters) of this string buffer.
StringBuffer replace(int start, int end, String str)	Replaces the characters in a substring of the string buffer with the characters in the given string.
StringBuffer reverse()	The character sequence contained in this string buffer is replaced by the reverse of the sequence.
void setCharAt(int index, char ch)	The character at the given index of the string buffer is set to ch.
void setLength(int newLength)	Sets the length of the string buffer.
String substring(int start)	Yields a new string that contains a subsequence of characters currently contained in this string buffer. The substring begins at the given index.
String substring(int start, int end)	Yields a new string that contains a subsequence of characters currently contained in this string buffer.
String toString()	Converts to a string representing the data in this string buffer.

Here are the constants and methods of the **Math** class:

- **double E**—The constant *e* (2.7182818284590452354)
- **double PI**—The constant *pi* (3.14159265358979323846)
- **double sin(double a)**—Trigonometric sine
- **double cos(double a)**—Trigonometric cosine
- **double tan(double a)**—Trigonometric tangent
- **double asin(double a)**—Trigonometric arcsine
- **double acos(double a)**—Trigonometric arccosine
- **double atan(double a)**—Trigonometric arctangent
- **double atan2(double a, double b)**—Trigonometric arctangent, two-operand version
- **double exp(double a)**—Raise *e* to a power
- **double log(double a)**—Log of a value
- **double sqrt(double a)**—Square root of a value
- **double pow(double a, double b)**—Raise to a power
- **double IEEEremainder(double f1, double f2)**—IEEE remainder method
- **double ceil(double a)**—Ceiling method

- **double floor(double a)** — Floor method
- **double rint(double a)** — Random integer
- **int round(float a)** — Rounds a float
- **long round(double a)** — Rounds a double
- **double random()** — Random number
- **int abs(int a)** — Absolute value of an **int**
- **long abs(long a)** — Absolute value of a **long**
- **float abs(float a)** — Absolute value of a **float**
- **double abs(double a)** — Absolute value of a **double**
- **int min(int a, int b)** — Minimum of two **int** types
- **long min(long a, long b)** — Minimum of two **long** types
- **float min(float a, float b)** — Minimum of two **float** types
- **double min(double a, double b)** — Minimum of two **double** types
- **int max(int a, int b)** — Maximum of two **int** types
- **long max(long a, long b)** — Maximum of two **long** types
- **float max(float a, float b)** — Maximum of two **float** types
- **double max(double a, double b)** — Maximum of two **double** types