Collating Sequence

The bit patterns representing a character can be interpreted as an unsigned integer and so the natural order of numbers can be used to order the characters.

Collating sequence. A collating sequence of a character set is the order of the individual characters. The order may be determined by the underlying bit representation of the characters.

Examples of the Unicode Collating Sequence

A<a	U+0041<U+0061	$65 < 97$
Z<a	U+005A<U+0061	$90 < 97$
a<b	U+0061<U+0062	$97 < 98$
e<f	U+0063<U+0064	$101 < 102$
z<ñ	U+007A<U+00F1	$122 < 241$
ö<ü	U+00F6<U+00FC	$246 < 252$
ł<ŵ	U+0142<U+0175	$322 < 373$
$\xi < \phi$	U+03BE<U+03C6	$958 < 966$
$\int < \ncong$	U+222B<U+2247	$8747 < 8820$
$\ncong < \otimes$	U+2247<U+2297	$8820 < 8895$
$\ncong < \odot$	U+2247<U+2299	$8820 < 8897$
ﬁ<ﬂ	U+FB01<U+FB02	$64257 < 64258$

For many natural languages this ordering is *not* the usual ordering of the alphabet.

Collation

A default ordering for all characters in Unicode, `allkeys.txt`, UTS #10.

`allkeys.txt`

```
0061  ; [.0A15.0020.0002.0061] # LATIN SMALL LETTER A
FF41  ; [.0A15.0020.0003.FF41] # FULLWIDTH LATIN SMALL LETTER A; QQK
249C  ; [*027A.0020.0004.249C][.0A15.0020.0004.249C][*027B.0020.001F.249C] # PARENTHESIZED LATIN SMALL LETTER
1D41A ; [.0A15.0020.0005.1D41A] # MATHEMATICAL BOLD SMALL A; QQK
1D44E ; [.0A15.0020.0005.1D44E] # MATHEMATICAL ITALIC SMALL A; QQK
1D4B6 ; [.0A15.0020.0005.1D4B6] # MATHEMATICAL SCRIPT SMALL A; QQK
1D4EA ; [.0A15.0020.0005.1D4EA] # MATHEMATICAL BOLD SCRIPT SMALL A; QQK
1D51E ; [.0A15.0020.0005.1D51E] # MATHEMATICAL FRAKTUR SMALL A; QQK
1D552 ; [.0A15.0020.0005.1D552] # MATHEMATICAL DOUBLE-STRUCK SMALL A; QQK
1D586 ; [.0A15.0020.0005.1D586] # MATHEMATICAL BOLD FRAKTUR SMALL A; QQK
1D5BA ; [.0A15.0020.0005.1D5BA] # MATHEMATICAL SANS-SERIF SMALL A; QQK
1D5EE ; [.0A15.0020.0005.1D5EE] # MATHEMATICAL SANS-SERIF BOLD SMALL A; QQK
1D622 ; [.0A15.0020.0005.1D622] # MATHEMATICAL SANS-SERIF ITALIC SMALL A; QQK
1D68A ; [.0A15.0020.0005.1D68A] # MATHEMATICAL MONOSPACE SMALL A; QQK
24D0  ; [.0A15.0020.0006.24D0] # CIRCLED LATIN SMALL LETTER A; QQK
0041  ; [.0A15.0020.0008.0041] # LATIN CAPITAL LETTER A
FF21  ; [.0A15.0020.0009.FF21] # FULLWIDTH LATIN CAPITAL LETTER A; QQK
1D400 ; [.0A15.0020.000B.1D400] # MATHEMATICAL BOLD CAPITAL A; QQK
1D434 ; [.0A15.0020.000B.1D434] # MATHEMATICAL ITALIC CAPITAL A; QQK
1D49C ; [.0A15.0020.000B.1D49C] # MATHEMATICAL SCRIPT CAPITAL A; QQK
1D504 ; [.0A15.0020.000B.1D504] # MATHEMATICAL FRAKTUR CAPITAL A; QQK
1D538 ; [.0A15.0020.000B.1D538] # MATHEMATICAL DOUBLE-STRUCK CAPITAL A; QQK
1D56C ; [.0A15.0020.000B.1D56C] # MATHEMATICAL BOLD FRAKTUR CAPITAL A; QQK
1D5A0 ; [.0A15.0020.000B.1D5A0] # MATHEMATICAL SANS-SERIF CAPITAL A; QQK
1D5D4 ; [.0A15.0020.000B.1D5D4] # MATHEMATICAL SANS-SERIF BOLD CAPITAL A; QQK
1D608 ; [.0A15.0020.000B.1D608] # MATHEMATICAL SANS-SERIF ITALIC CAPITAL A; QQK
1D670 ; [.0A15.0020.000B.1D670] # MATHEMATICAL MONOSPACE CAPITAL A; QQK
24B6  ; [.0A15.0020.000C.24B6] # CIRCLED LATIN CAPITAL LETTER A; QQK
00AA  ; [.0A15.0020.0014.00AA] # FEMININE ORDINAL INDICATOR; QQK
00E1  ; [.0A15.0020.0002.0061][.0000.0032.0002.0301] # LATIN SMALL LETTER A WITH ACUTE; QQCM
```

The switch away from “traditional” Spanish ordering of the digraphs ch and ll as separate letters is well advanced.

Spanish

Spanish (traditional)

Multilingual ordering of European languages is being standarized.

For example, in Java `c1<c2` is defined to be `((int)c1)<((int)c2)`. In fact, the cast is a no-op in Java; the bits stay the same, only the interpretation changes.

In Java, characters are automatically promoted to integers (no cast is needed). Characters are *not* automatically promoted to **short**. One can cast them to **short**; this is a bad idea even though no bits are lost, because the 16-bit, twos-compliment representation of **short** is incompatible with the intuitive, collating sequence of 16-bit **char**.

```
final char xc = 'A';
final char zc = '\ufb01';  // fi ligature
final short xs = (short)xc, zs =(short)zc;
System.out.println (xc<zc); // true
System.out.println (xs<zs); // false
```

There is no unsigned, 16-bit, integral type in Java. There are no unsigned integral types in Java at all.

Lexicographic Ordering

An ordering of characters gives rise to an order on strings of those characters. Strings of characters of (possibly) different lengths are ordered by the first difference in the strings.

Let $<_C$ be the ordering on characters, e.g., $x <_C y$ for any two characters x and y. Let $x_0 x_1 \cdots x_{k-1}$ and $y_0 y_1 \cdots y_{l-1}$ be two strings of length $k \geq 0$ and $l \geq 0$, respectively. The two strings are equal $x_0 x_1 \cdots x_{k-1} = y_0 y_1 \cdots y_{l-1}$, if $k = l$ and $x_i = y_i$ for all $0 \leq i \leq k$.

Lexicographic Ordering

Now let us define lexicographic ordering $<_L$.

Lexicographic ordering. We define $x_0 x_1 \cdots x_{k-1} <_L y_0 y_1 \cdots y_{l-1}$ if there is an index $0 \leq i$ such that $i < k$, $i < l$, $x_i <_C y_i$ and $x_j = y_j$ for all $0 \leq j < i$, or if $l > k$ and for all $0 \leq j < k$ we have $x_j = y_j$.

```
alligator  <  crocodile
alligator  <  ant
aardvark   <  anteater
ant        <  armadillo
ant        <  anteater
anteater   <  antelope
```

Notice that the empty string (the sequence of 0 characters) is the smallest string in lexicographic order.

Lexicographic Ordering

Now let us define lexicographic ordering $<_L$.

Lexicographic ordering. We define $x_0 x_1 \cdots x_{k-1} <_L y_0 y_1 \cdots y_{l-1}$ if one of the following hold

$$k = 0$$
$$k > 0 \text{and} l > 0 \text{and} x_0 <_C y_0$$
$$k > 0 \text{and} l > 0 \text{and} x_0 = y_0 \text{and} x_1 \cdots x_{k-1} <_L y_1 \cdots y_{l-1}$$

Notice:

`aaaargh`	$<$	`aaargh`
`aaargh`	$<$	`aardvark`
`aardvark`	$<$	`arc`
`arc`	$<$	`rack`

More formally we note that the set $\{ a^n b \mid n \geq 0 \}$ has no least element

$$\ldots < aaab < aab < ab < b$$

This means lexicograph order is not a total ordering.
Since this complicated reasoning by induction, words over an alphabet can be considered ordered another way: first by length, then lexicographically for strings of the same length. This ordering is total.

Discrete Math

See, for example, Section 4.3.3 in *Discrete Structures, Logic, and Computability*, 2nd edition, by James L. Hein.

Java Method

A recursive Java method to implement lexicographic ordering on strings based on the Unicode collating sequence:

```
static boolean lexicographic (String x, String y) {
  if (y.length()==0) return false;
  else if (x.length()==0) return true;
  else if (x.charAt(0) < y.charAt(0)) return true;
  else if (x.charAt(0) > y.charAt(0)) return false;
  else return lexicographic (
      x.substring(1), y.substring(1));
}
```

Dictionary Ordering

Natural languages often have numerous special rules about: ignorable characters, capitalization, diacritics, digraphs, etc.

Ignorable characters:

```
dictionary:    coal < concentrate < co-operate < corporation
lexicographic: co-operate < coal < concentrate < corporation
```

Capitalization

```
dictionary:     abduct < Abelian < Aberdeen <  abet
lexicographic:  Abelian < Aberdeen <  abduct < abet
```

Diacritics

`dictionary: cote < côte < coté < côté`
`lexicographic: cote < coté < côte < côté`

Digraphs

`dictionary: casa < como < chalupa < dónde`
`lexicographic: casa < chalupa < como < dónde`

Other, more complex rules require semantic analysis. Mc=Mac, Mrs.=Mistress, St.=Saint, 1812=Eighteen twelve. Ignoring an initial article, etc. See Knuth, volume 3, pages 8–9.

Time Stamp

Recording the time of an event using a time stamp is a very common task in database and other programs. A good choice for the format of a time stamp is one for which the timestamps when sorted in lexicographic order are also in temporal order.

```
Aug 8, 2010, 10:56:32.876 am   2010-08-08T10:56:32.876Z
Dec 3, 2010, 9:04:01.327 am    2010-12-03T90:04:01.327Z
Sep 21, 2010, 2:03:11.002 pm   2010-09-21T14:03:11.002Z
```

Month names when sorted in lexicographic order (even when abbreviated to three characters) are not in chronological order. Also the string of length one `"8"` is not less than the string of length two `"10"`.

This trick obviates the need for a special timestamp function to compare two timestamps in chronological order. Such a function can be difficult to write correctly due to the irregular nature our society uses in keeping time.

Dictionary Ordering

Java can compare strings in dictionary order using the class `java.text.Collator`.

```
Collator co = Collator.getInstance (Locale.US);
co.setStrength (Collator.PRIMARY);
if (co.compare ("abc", "ABC")==0) {
   System.out.println ("Equivalent.");
}
```

The difference between "a" and "b" is considered primary, while the difference between "e" and "é" is secondary, and the difference between "e" and "E" is tertiary.

An Aside

Where do objects come from?

Ultimately they are constructed, but we have just seen an example of an important idiom or pattern.

Singleton Pattern

Late creation or frugal management of large objects is often controlled by a static method that creates an instance of a class on behalf of the client.
In this way the correct subclass or implementation can be created.
Also the number of these objects can be controlled so that repeated requests for the object will be fulfilled by returning the same instance.

```
java.lang.Runtime.getRuntime();
java.util.Calendar.getInstance();
java.text.Collator.getInstance (Locale.US);
java.text.NumberFormat.getInstance (Locale.US);
java.security.KeyFactory.getInstance ("DSA");
java.security.MessageDigest.getInstance ("MD5");
java.awt.AlphaComposite.getInstance (
    AlphaComposite.SR_OUT)
```