Figure 9.0 Comparing *Strings*.

Strings are objects, so the value of a **String** variable is a *reference*.
What does this print?

```
String s = new String("hello");
String t = new String("hello");
if (s == t)
    System.out.println("the same");
else
    System.out.println("not the same");
```

Use the **equals()** method of the **String** class to see if two strings are the same!

```
if (s.equals(t))   ...
```

Similarly, to see which string comes earlier in the alphabet:

```
int s_relativeTo_t = s.compareTo(t);
```

s.compareTo(t) returns

0 if **s** and **t** are the same
< 0 if **s** comes before **t**
> 0 if **s** comes after **t**

For example,

```
"zoo".compareTo("aardvark"));    // returns [ ]
"cat".compareTo("dog"));    // returns [ ]
"horse".compareTo("horse"));    // returns [ ]
```

What exactly does it mean to *compare* two strings? Java uses "dictionary" or *lexicographic* order. Just remember that:

- blank comes before all other printable characters, and
- A … Z, a … z, 0 … 9 are each contiguous and in order.

Exercise: Try the first example on this page, but this time use the short form of **String** instantiation for **s** and **t** (i.e. **String s = "hello"; String t = "hello";**). What's the output? Maybe there's more to this "short form" than we've discussed!