Comparing Strings

The *relational operators* used with the primitive data types (i.e., less than, greater than, equal to, etc.) should not be used to compare strings. Recall that a `String` variable does not actually contain the string, but only the location (address) of the string object in memory. Thus any comparison between strings using relational operators would actually be comparing two addresses, rather than the `String` values.

Java provides a number of *methods* for comparing strings, but for now we will introduce only the `equals` and the `compareTo` methods. These are *String methods*, which means it can be called from any *String variable*.

The `equals` method – testing for identical strings

The `equals` method tests whether or not two strings are *identical*, returning true if they are identical, and false otherwise.

Example 1 – given the declaration

```
String s = "Same";
```

then the `equals` method would yield the following results:

a) `s.equals("Same")` will return true

b) `s.equals("same")` will return false because 'S' is not equal to 's'

c) `s.equals("Same  ")` will return false because of the spaces at the end of the word

The `compareTo` method – determining the alphabetical order of strings

The `compareTo` method provides information about the *lexicographic order* of two strings, which is, essentially, their dictionary order (except upper-case letters precede lower-case letters). The strings are compared character by character until their order is determined, or they prove to be identical.

The compareTo method returns an *integer* value that indicates the ordering of the two strings, and uses the following syntax.

```
string1.compareTo(string2)
```

The result will be zero if the strings are identical, negative if `string1` precedes `string2`, and positive if `string1` follows `string2`.

Example 2

a) `"cab".compareTo("car")` < 0 because 'b' < 'r'

b) `"Car".compareTo("car")` < 0 because 'C' < 'c'

c) `"27".compareTo("186")` > 0 because '2' > '1' (the overall numeric values are ignored)

d) `"car".compareTo("cart")` < 0 because "car" is only the first part of "cart"

Exercises

1. For each of the following pairs of strings, state, with reasons, which string precedes the other.

   (a) "cat" and "dog"

   (b) "cat" and "Cathy" 

   (c) "X " and " X"

   (d) "cab" and "CAR"

   (e) "XX" and "X X"

   (f) "XY" and "XY "

   (g) "375" and "84" 

   (h) "" and " "

2. State the value of each expression

   (a) "one".equals("one ")

   (b) "two".equals("2")

   (c) "Three".equals("Three")

   (d) "four".equals("for")

3. State whether the value of the expression is negative, zero, or positive.

   (a) "for".compareTo("fore")

   (b) "fore".compareTo("force")

   (c) "force".compareTo("Force")

   (d) "Force".compareTo("Farce")

   (e) "Farce".compareTo("Fare")

   (f) "Fare".compareTo("far")

   (g) "far".compareTo("far")

   (h) "far".compareTo("far ")

4. Write a program that asks the user for two strings, `username` and `password`. Inside your program, you should hard code the `correctUser` and `correctPassword`. Compare the user's input to the values stored in the program and output appropriate messages if the user gets both correct, only one correct, or neither correct. You may have more than one user and password if you like.

   Note: To hard code the username or password, you will need to set your String variable in the program. For example,

```
String correctPassword = "p@ssw0rd";
```