Java String comparison FAQ: How to compare Java Strings

Java String comparison FAQ: Can you share some examples of how to compare Strings in Java?

If you're like me, when I first started using Java, I wanted to use the "==" operator to test whether two String instances were equal, but for better or worse, that's not the correct way to do it in Java.

In this tutorial I'll demonstrate several different ways to correctly *compare* Java strings, starting with the approach I use most of the time. At the end of this Java String comparison tutorial I'll also discuss why the "==" operator doesn't work when comparing Java strings.

Option 1: Java String comparison with the equals method

Most of the time (maybe 95% of the time) I compare strings with the *equals* method of the Java Stringclass, like this:

```
if (string1.equals(string2))
```

This String equals method looks at the two Java strings, and if they contain the exact same string of characters, they are considered equal.

Taking a look at a quick String comparison example with the equals method, if the following test were run, the two strings would not be considered equal because the characters are not the exactly the same (the case of the characters is different):

```
String string1 = "foo";
String string2 = "FOO";

if (string1.equals(string2))
{
```

```
  // this line will not print because the
  // java string equals method returns false:
  System.out.println("The two strings are the same.")
}
```

But, when the two strings contain the *exact same* string of characters, the *equals* method will return true, as in this example:

```
String string1 = "foo";
String string2 = "foo";

// test for equality with the java string equals method
if (string1.equals(string2))
{
  // this line WILL print
  System.out.println("The two strings are the same.")
}
```

Option 2: String comparison with the equalsIgnoreCase method

In some string comparison tests you'll want to ignore whether the strings are *uppercase* or *lowercase*. When you want to test your strings for equality in this case-insensitive manner, use the*equalsIgnoreCase* method of the String class, like this:

```
String string1 = "foo";
String string2 = "FOO";

// java string compare while ignoring case
if (string1.equalsIgnoreCase(string2))
{
  // this line WILL print
  System.out.println("Ignoring case, the two strings are the same.")
}
```

Option 3: Java String comparison with the compareTo method

There is also a third, less common way to compare Java strings, and that's with the String class*compareTo* method. If the two strings are exactly the same, the compareTo method will return a value of 0 (zero). Here's a quick example of what this String comparison approach looks like:

```
String string1 = "foo bar";
String string2 = "foo bar";

// java string compare example
if (string1.compareTo(string2) == 0)
{
  // this line WILL print
  System.out.println("The two strings are the same.")
}
```

(As a quick note, I haven't used this approach in many years. If anyone would like to add a comment on when this approach is recommended, please leave a note below.)

Java String compare: why "==" doesn't work

As I mentioned above, when I first began working with Java I wanted to compare strings with the "==" operator, like this:

```
if (string1 == string2)
{
  // do something ...
}
```

However, unlike C++, Java does not let us overload operators like the "==" operator. The designers of the Java language intentionally made this design decision to simplify the language. While it seems counter-intuitive at first, once you realize that there is an equals method on the String class, it's not a big deal.

(For more information on "==" in regards to Java String comparison, please see a detailed explanation in the comments area below, specifically the comment entry title "Example of how/why '==' doesn't work".)

A closing note - the Java Object equals method

While I'm writing about this concept of equality in Java, it's important to note that the Java language includes an *equals* method in the base Java Object class. Whenever you're creating your own objects and you want to provide a means to see if two instances of your object are "equal", you should override (and implement) this equals method in your class (in the same way the Java language provides this equality/comparison behavior in the String equals method).