Confusing Comparable- & Comparator-Contract

Recently I came accross the following exception, thrown by the Java Runtime Environment:

"Comparison method violates its general contract!"

The Exception occurred in a Comparable-implementation looking similar as the following code, when Collections.sort() was invoked on an ArrayList of the Foo type:

I was a bit confused, because as I knew a comparator doesn't necessarily have to be consistent with equals (see here). The same situation we have when implementing the Comparable interface (see here).

Until Java 1.6, sort functionality was fine with equal-inconsistent Comparator- / Comparable-implementations. Even if your Comparable / Comparator implementations returned the same non-zero integer value for equal objects, Collections.sort(..) would have been working fine. Not now, after *timsort* - a better-performing sorting algorithm by Tim Peter in Python - was introduced by Oracle with Java 7 to be the default sorting algorithm, whose implementation obviously needs the Comparable- and Comparator-interfaces to be consistent with equals. After not being able to reproduce the Exception mentioned above for some time, I finally managed to write a sample Project, which is able to demonstrate the issue. You need three classes:

File 1: CollectionsAutomatedTest.java

File 2: Foo.java

File 2: FooComparator.java

Findings

When running the main() under a Java 7 based runtime environment - I tested it with JDK 1.7.0_21 - you'll get the following output:

When running the main() under a Java 1.6 based runtime environment - I tested it with JDK 1.6.0_37 - you'll get the following output:

This only proves empirically, that the Sort-Functionality requires the Comparator- and

Comparable-interfaces to be consistent with equals to work as expected under JDK 1.7.0_21, but not under JDK 1.6.0_37. To demonstrate, that this issue is related to timsort, let us run the application under JDK 1.7.0_21 and disable timsort using the useLegacyMergeSort-system property. Just add the following line to be the first line within the main-method:

and run the application again. This would produce the following output even under JDK 1.7.0_21:

Consequences

In words: The introduction of timsort changed the behaviour of Collections.sort() / Arrays.sort(). Starting from Java 7 you have the following options:

- either make sure, your Comparator- / Comparable-interfaces are consistent with equals (see example beneath) or
- use the ‘-Djava.util.Arrays.useLegacyMergeSort=true' switch to disable timsort

To make the compareTo()-implementation consistent with equals, just make sure, that the method not only provides negative / positive integers for smaller / bigger object but also returns 0 for objects where a.equals(b) or b.equals(a) would return true. To continue with the example from the beginning (where no date may ever be equal to null), this would turn out to be: