

## Base Language - Bug #7722

### improve performance of equals/not equals when used with character/longchar

08/21/2023 06:14 AM - Constantin Asofiei

<b>Status:</b>	WIP	<b>Start date:</b>	
<b>Priority:</b>	Normal	<b>Due date:</b>	
<b>Assignee:</b>	Dănuț Filimon	<b>% Done:</b>	80%
<b>Category:</b>		<b>Estimated time:</b>	0.00 hour
<b>Target version:</b>		<b>case_num:</b>	
<b>billable:</b>	No		
<b>vendor_id:</b>	GCD		
<b>Description</b>			

#### History

##### #1 - 08/21/2023 06:20 AM - Constantin Asofiei

In a large app, there are ~350k calls of `Text.compareTo` which originate from equals/not-equals operators (via `CompareOps`). In these cases, we should use the uppercase compare only if their lengths are equal. For different lengths, there is no way for this equals/not-equals operators to return true.

Also, keep in mind that the lengths need to be compared using the `StringHelper.safeTrimTrailingSpaces` result.

`CompareOps.equals` is also called from `BaseDataType.equals`.

##### #2 - 08/21/2023 07:10 AM - Alexandru Lungu

- Assignee set to Dănuț Filimon

##### #3 - 08/21/2023 08:46 AM - Alexandru Lungu

This makes sense. We have some recent implementation in FWD-H2 that was targeting this exact issue. We can even compare strings by reference `s1 == s2` to short-circuit this. Also, we can use `regionMatches` instead of `2 toUpperCase` and `1 equals`.

Danut, please mind that this is not about how `Text.compareTo` is implemented, but the fact that a `Text.equals` should have been used instead.

##### #4 - 08/21/2023 11:10 AM - Constantin Asofiei

Alexandru Lungu wrote:

This makes sense. We have some recent implementation in FWD-H2 that was targeting this exact issue. We can even compare strings by reference `s1 == s2` to short-circuit this.

This only works with intern'ed strings, right?

Also, we can use `regionMatches` instead of `2 toUpperCase` and `1 equals`.

There is this comment in `Text.compareTo`:

```
// DO NOT use String.compareToIgnoreCase() since this lowercases and yields different
// results for >, <, >= and <= forms when [ \ ^ _ ' characters are included in the operands
return s1.toUpperCase().compareTo(s2.toUpperCase());
```

I don't think regionMatches can be used.

Danut, please mind that this is not about how Text.compareTo is implemented, but the fact that a Text.equals should have been used instead.

I think we need overloads as CompareOps.equals(Text, Text), to use Text.equals.

#### #5 - 08/22/2023 03:27 AM - Alexandru Lungu

Constantin Asofiei wrote:

This only works with intern'ed strings, right?

Right. In FWD-H2 this short-circuit had a ~2% hit rate if I remember correctly. This was due to the fact that FWD-H2 uses a light-weight simulated string pool (mostly like an array cache with hash index), which allows the reuse of the same string instances where possible. In FWD however, I expect to see some strings that are interned at start-up, so this short-circuit may have a nice hit rate. This needs to be tracked.

We can leave this at the very end of the implementation.

Also, we can use regionMatches instead of 2 toUpperCase and 1 equals.

There is this comment in Text.compareTo:

[...]

I don't think regionMatches can be used.

I am aware of this comment. We need to investigate this further. Our text comparison methods are not uniform across the application. In FWD-H2 we have collation sensitive comparisons (generating and caching collation keys). I expect that the note you mention is in fact a collation problem in FWD, rather than a Java problem. Unfortunately, including collation aware comparisons may actually affect performance, but guarantee that we have a sound **and** complete solution.

**But**, this issue is only for comparisons, not equal checks. Therefore, most probably we can use regionMatches for equality and keep compareTo for comparisons.

My conclusion for regionMatches: we need to track down the issue that caused the mentioned note. If it proves to be a collation issue, we can open a separate task for further investigation.

#### #6 - 08/22/2023 05:22 AM - Alexandru Lungu

Created 7722a.

#### #7 - 08/22/2023 06:58 AM - Dănuț Filimon

**Committed 7722a/rev.14704.** Added Text.equals() method which uses regionMatches. The method is called in CompareOps.equals when both instances are Text.

#### #8 - 08/22/2023 08:27 AM - Alexandru Lungu

##### Review of 7722a/rev.14704

- Use Override annotation for equals in Text. Make it return false if the parameter is not a Text. This will make the casting safe
- Use a single return with regionMatches. The first parameter can be directly caseSens || o.caseSens instead of the if conditional.
- I think Text.value and Text.unknown can be set in the same time (?). In this case, the code fails for op1 |-> {value: "a" unknown: true} and op2 |-> {value: "a" unknown: false}. In this case, even if they have the same value, one is known and the other is unknown. Please add the value o.value short-circuit **after** you do the (unknown | null)-check.
- unknown || o.unknown is more concise than value == null || o.value == null.

#### #9 - 08/24/2023 03:46 AM - Alexandru Lungu

Danut, please get this done **asap**. I am planning to do a set of regression tests + profiling. I am keen on getting this merged today.

#### #10 - 08/24/2023 03:46 AM - Alexandru Lungu

- Status changed from New to WIP

- % Done changed from 0 to 80

#### #11 - 08/24/2023 05:10 AM - Dănuț Filimon

Alexandru Lungu wrote:

Danut, please get this done **asap**. I am planning to do a set of regression tests + profiling. I am keen on getting this merged today.

**Committed 7722a/rev.14705.** Made changes based on the [#7722-8](#) review.

#### #12 - 08/24/2023 06:00 AM - Constantin Asofiei

Alexandru, there is a concern; if we compare case-insensitive, there is no guarantee that a lowercase and an uppercase letter is always of the same length. See this for a starting point: <https://stackoverflow.com/questions/2357315/does-javas-tolowercase-preserve-original-string-length>, especially the STRASSE vs straÙe case.

#### #13 - 08/24/2023 07:45 AM - Alexandru Lungu

Constantin Asofiei wrote:

Alexandru, there is a concern; if we compare case-insensitive, there is no guarantee that a lowercase and an uppercase letter is always of the same length. See this for a starting point: <https://stackoverflow.com/questions/2357315/does-javas-tolowercase-preserve-original-string-length>, especially the STRASSE vs straÙe case.

This is part of my recent observation regarding the collation specific comparisons. They are done in FWD-H2, but not in FWD (note my [#7722-5](#), second part). Take the following example:

```
Collator coll = Collator.getInstance(new Locale("tr"));

coll.setStrength(Collator.PRIMARY);
System.out.println(coll.compare("straße", "STRASSE")); // 1
coll.setStrength(Collator.SECONDARY);
System.out.println(coll.compare("straße", "STRASSE")); // 1
coll.setStrength(Collator.TERTIARY);
System.out.println(coll.compare("straße", "STRASSE")); // 1
coll.setStrength(Collator.IDENTICAL);
System.out.println(coll.compare("straße", "STRASSE")); // 1

coll = Collator.getInstance(new Locale("fr"));

coll.setStrength(Collator.PRIMARY);
System.out.println(coll.compare("straße", "STRASSE")); // 0
coll.setStrength(Collator.SECONDARY);
System.out.println(coll.compare("straße", "STRASSE")); // 0
coll.setStrength(Collator.TERTIARY);
System.out.println(coll.compare("straße", "STRASSE")); // -1
coll.setStrength(Collator.IDENTICAL);
System.out.println(coll.compare("straße", "STRASSE")); // -1
```

The point here is that the comparison is highly dependent upon the used locale / collator. When it comes down to equality, the collator simply uses `compare(source, target) == Collator.EQUAL`. Therefore, there is no "faster" way of doing equals instead of `compareTo`.

I guess we can disregard the length comparison out of obvious reasons (for case-insensitive). In FWD-H2, we generate collation keys that we cache to do the proper comparison. In fact, we generate a collation key for case-sensitive and one for ignore-case (that is the upper-case). We don't use TERTIARY comparison, but we just compare the upper-case versions - I guess this can be improved.

For the moment, we either:

- keep the `toUpperCase` solution and short-circuit the comparison if the upper-case versions have a different number of characters. This won't avoid the upper-case, but will short-circuit the `compareTo`.
- move to the collation approach with `SECONDARY` level for case-insensitive and `IDENTICAL` for case-sensitive.

In 4GL, message `"straße" = "STRASSE"`. prints yes.