Base Language - Feature #4629

implement fully compatible 4GL collections (backed by protected temp-tables)

04/27/2020 09:55 AM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Base Language - Feature #4384: Builtin OO Implementation	**Closed**
Related to Base Language - Feature #3751: implement support for OO 4GL and st...	**Closed**
Related to Base Language - Feature #4373: finish core OO 4GL support	**New**

History

#1 - 04/27/2020 10:02 AM - Greg Shah

A simple version of the 4GL collections was implemented instead of the temp-table backed alternative in #4384.

As noted by Marian:

> I have a question about collections/maps, the implementation in 4GL is based on temp-tables but I would say using existing Java collections would make more sense... the problem is often the inner temp-tables are protected hence visible for extending classes. The only option here to be 100% compatible with the 4GL implementation is to stick to their 'limited' implementation using temp-tables. I would say those are more general use objects that are less likely to be extended by anyone but as long as this is possible it's always a possibility:(

My response:

> Nasty.
>
> For now, let's do the simple approach and map things to the Java collections.
>
> As usual, we will eventually hit an application which has such code. It is inevitable. But we will wait to implement the full tmep-table approach until then. For the common case, it is much better to use the Java collections. So even when we do implement the temp-table backed 4GL collections, we will only use them for the subclasses which have this bad behavior (directly inspecting/manipulating data structures in parent classes is considered quite "dirty"). We will have to implement some special conversion rules in that case, detecting any access to that protected table. But we will leave this part until later.

This task (#4629) is for the deferred work. We should not need hints. Instead we would write rules to detect access to the protected temp-table members and in this case (and ONLY in that case) change the target parent class to be the one that is fully compatible. To make things work we probably need to have the fully compatible class as a subclass of our "mostly compatible" class. It is not "foolproof", since any reflective code may break in this approach. But I think the benefit of using the clean collections is greater than the cost of this special case (reflection dependencies on the compatible map parent class). If we ever hit that case, we can always add hints to force conversion of everything using the compatible classes.

#2 - 04/27/2020 10:02 AM - Greg Shah

- Related to Feature #4384: Builtin OO Implementation added

#3 - 04/27/2020 10:04 AM - Greg Shah

- Related to Feature #3751: implement support for OO 4GL and structured error handling added

#4 - 04/27/2020 10:05 AM - Greg Shah

- Related to Feature #4373: finish core OO 4GL support added