Base Language - Bug #5029

OO constructor with array input parameter

12/04/2020 08:43 AM - Marian Edu

Status:	Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 12/04/2020 08:43 AM - Marian Edu

The way newInstance works when the ctor has only one input parameter array we can't simply pass that array as an argument since it will be seen as multiple arguments (args), the converted code for this case shows an InputOutputExtentParameter being used. However the initialization code does not handle that, at least in our branch, and it fails to find the appropriate ctor to call. Did tried a quick fix to handle that case and then the right ctor is being selected but then it fails later on argument validations with DYN_ARRAY_FOR_NON_ARRAY.

Before going further into this thought to ask if that was already fixed maybe?

#2 - 12/04/2020 01:04 PM - Constantin Asofiei

For a ctor call like:

```
def var c1 as char extent 5.
o = new oo.OExt(input-output c1).
```

the hand-written java code needs to be like:

```
         o.assign(ObjectOps.newInstance(com.goldencode.testcases.oo.Oext.class, "U", new InputOutputExtentPara
meter<character>()
         {
            public character[] getVariable()
            {
               return c1;
            }

            public void setVariable(final character[] newRef)
            {
               c1 = newRef;
            }
         }));
```

FWD has a problem validating the parameters against AbstractParameter or AbstractExtentParameter argument instances. There's a TODO there in ControlFlowOps line 5192, I'm working on fixing it.

#3 - 12/07/2020 02:31 AM - Marian Edu

Constantin Asofiei wrote:

> the hand-written java code needs to be like:
> [...]

Yeah, this is what we use although input-output seems to be the only option even if only input is what we need :)

> FWD has a problem validating the parameters against AbstractParameter or AbstractExtentParameter argument instances. There's a TODO there in ControlFlowOps line 5192, I'm working on fixing it.

OK, then we're waiting for you ;)

Thanks

#4 - 12/17/2020 01:15 PM - Constantin Asofiei

Marian, I have the conversion rules and runtime for this and #4749, but I need to test them with the customer apps. It will take a few days to reconvert.

In the mean time, some notes. As I understand, you need to use an INPUT parameter. For this, pass the Java array reference directly, and cast it to Object, like ObjectOps.newInstance("oo.OExt", "I", (Object) extVar);. This will let the compiler know that extVar will be treated as a distinct reference, and not the entire vararg array.

#5 - 12/18/2020 02:04 AM - Marian Edu

Constantin Asofiei wrote:

> Marian, I have the conversion rules and runtime for this and #4749, but I need to test them with the customer apps. It will take a few days to reconvert.

Great, let us know when we can have that patch applied or we will probably sync and have a new branch before anyway.

> In the mean time, some notes. As I understand, you need to use an INPUT parameter. For this, pass the Java array reference directly, and cast it to Object, like ObjectOps.newInstance("oo.OExt", "I", (Object) extVar);. This will let the compiler know that extVar will be treated as a distinct reference, and not the entire vararg array.

Right, that is indeed one option when we do it from java side, as a work-around I've called the default ctor and then the method that takes the input

array parameter to get the same result. However the 4GL converted code seems to use an InputOutput parameter for any input array, that's why we've went down that route.

#6 - 12/18/2020 03:42 AM - Constantin Asofiei

Marian Edu wrote:

> However the 4GL converted code seems to use an InputOutput parameter for any input array.

That was a bug.

#7 - 01/20/2021 08:15 AM - Marian Edu

Constantin Asofiei wrote:

> Marian Edu wrote:
>
> > However the 4GL converted code seems to use an InputOutput parameter for any input array.
>
> That was a bug.

Constantin, if that one happens to be fixed it would be great if we can have the right code in our branch cause we're testing now JsonArray and none of the ctor's that have an extent input parameter are considered a match by the initialization routine :(

If InputOutputExtentParameter used by conversion is the right choice then this probably should be handled correctly in resolveLegacyEntry method... only there are about 500 lines of code in that method and is not clear where exactly the getValue should be used - it does look like the data type is resolved/matched a number of times and in the end the original arguments array is used again so I gave-up :(

#8 - 01/20/2021 08:26 AM - Marian Edu

Hmm, finding the right ctor is not the only problem, now the exception say "The extent parameter reference was not set, before the function/procedure end !" so probably the InputOutputExtentParameter is not to be used for plain extent input parameters at all so more a conversion issue than runtime :(

#9 - 02/04/2021 05:16 AM - Constantin Asofiei

Marian, many issues related to extent parameters and others are fixed in 3821c 11984. This will be in the 4384h.

Once that is available, please reconvert and check your tests.

I've also fixed the extent usage from external programs (if you recall, you had to use an internal procedure/function in some cases).

#10 - 02/04/2021 05:27 AM - Marian Edu

Constantin Asofiei wrote:

> Marian, many issues related to extent parameters and others are fixed in 3821c 11984. This will be in the 4384h.
>
> Once that is available, please reconvert and check your tests.
>
> I've also fixed the extent usage from external programs (if you recall, you had to use an internal procedure/function in some cases).

Sure thing Constantin, we have a number of classes on hold because conversion issues so will make sure we review those once we get the new branch... meanwhile I'm just staging all changes in our git repo and will merge those back once we get the new branch.

Thanks.

#11 - 02/11/2021 08:07 AM - Marian Edu

Constantin, conversion is not using InputOutputExtentParameter anymore but it's missing the cast to Object so it still doesn't find the right ctor, could you please add that cast during conversion?

Thanks

#12 - 02/11/2021 08:11 AM - Constantin Asofiei

What are you using exactly as argument? A OO property? A class variable? Something else?

#13 - 02/11/2021 08:14 AM - Marian Edu

Constantin Asofiei wrote:

> What are you using exactly as argument? A OO property? A class variable? Something else?

In this particular case a variable...

```
define variable jsonArr as JsonArray no-undo.
define variable cInExt  as character no-undo extent.

jsonArr = new JsonArray(cInExt) no-error.
```

This converts to...

```
silent(() -> jsonArr.assign(ObjectOps.newInstance(JsonArray.class, "I", cInExt[0])));
```

Adding a cast to (Object) for cInExt[0] solve the issue.

#14 - 02/11/2021 08:20 AM - Constantin Asofiei

OK, that's because my tests always had the INPUT modifier. I'll fix this.

For now, change it like INPUT cInExt (if is not something difficult, I mean you don't have to change a lot of code).

#15 - 02/11/2021 08:24 AM - Marian Edu

There is another thing with indeterminate extent output parameters, in 4GL it doesn't matter if the extent was already set before, calling the method will simply reset-it to the new value. In 4GL assignMulti fails if the variable was already used (initialized) and the new extent does not match the previous value.

In 4GI this works, after first call extent is 2 and changes to 3 after second call... no error there.

```
define variable vRspExt as datetime  no-undo extent.

vRspExt = jsonArr:GetDatetime(1, 2) no-error.
vRspExt = jsonArr:GetDatetime(1, 3) no-error.
```

In FWD the second call fails in assignMulti, we have to put `extent(var) = ?.` in between in our tests to make it work.

#16 - 02/11/2021 08:32 AM - Constantin Asofiei

My current work touches how 4GL moves the data from the arguments to the parameters, via the stack, for the integer/int64 combinations, when the argument is an expression.

Part of this, I found that if a function returns int has a return <expr> where the expression's compile time type is integer, but its runtime computed value is int64, the value will be kept on the stack as int64, and the range check is done when the returned value is actually assigned.

Your extent case may be something related, some kind of 'direct assignment' being done in such cases. I'll think about it, but I may have to postpone it a little.

#17 - 02/16/2021 01:12 PM - Constantin Asofiei

Constantin Asofiei wrote:

> OK, that's because my tests always had the INPUT modifier. I'll fix this.
>
> For now, change it like INPUT clnExt (if is not something difficult, I mean you don't have to change a lot of code).

Fixed in 3821c rev 12012

#18 - 02/16/2021 01:58 PM - Greg Shah

- Status changed from New to Test

- % Done changed from 0 to 100

#19 - 02/19/2021 11:14 AM - Constantin Asofiei

Marian, regarding #5029-15 - I've tried duplicating using this test:

```
define variable j as int  no-undo extent.

function func0 returns int extent (input n as int).
   def var k as int extent.
   extent(k) = n.
   k = n.
   message "k:" k[1].
   return k.
end.

j = func0(5).
message extent(j) j[1].
j = func0(3).
message extent(j) j[1].
```

But 4GL doesn't allow the j = func0(3) assignment (raises an ERROR condition that the extent is already set).

Please specify a 4GL test where you see this issue.

#20 - 02/22/2021 03:24 AM - Marian Edu

Constantin Asofiei wrote:

> Marian, regarding #5029-15 - I've tried duplicating using this test:
> [...]
>
> But 4GL doesn't allow the j = func0(3) assignment (raises an ERROR condition that the extent is already set).
>
> Please specify a 4GL test where you see this issue.

Constantin, you should probably ignore that... I do not remember the place where it was but I might have been read wrong the code in that test cause the 4GL does throw the same error is it was already set. I did check if there is some way to reset an already set extent by calling a method that does not return any value, returns an indeterminate extent or throw error but it looks like there is no way to make it work. The only option that does work without reseting the variable first is when the method returns an array with the same size. I will write a test to make sure this works the same in FWD but most probably it does so I've just read the code wrong :(