Base Language - Bug #7299

resolution of the target OO method in chained calls with poly arguments

04/24/2023 01:48 PM - Constantin Asofiei

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		

Description

History

#1 - 04/24/2023 01:49 PM - Constantin Asofiei

In #7261, this test was found to not work properly:

```
def var l as oo.ctest.
def var h as handle.

def temp-table tt1 field f1 as int field f2 as int field f3 as handle.
create tt1.
h = buffer tt1:handle.
l = new oo.ctest().

l:m1(h::f1):m2(h::f2):m1(h::f3).
```

with oo.Ctest:

```
class oo.CTest:

   method public oo.ctest m1(input i1 as int):
       message "m1 as int".
       return this-object.
   end.

   method public oo.CTest m1(input i1 as handle):
       message "m1 as handle".
       return this-object.
   end.

   method public oo.CTest m2(input i1 as int):
      message "m2 as int".
       return this-object.
   end.

   method public oo.CTest m3(input i1 as int):
      message "m3 as int".
       return this-object.
   end.

   method public oo.CTest m3(input i1 as handle):
      message "m3 as handle".
       return this-object.
   end.
end.
```

From some experimenting, OE computes the list of 'fuzzy method matches' and resolves the 'return type' of the call from the last one in the list. If you

change oo.CTest m1(input i1 as int) to progress.lang.object m1(input i1 as handle), then this no longer compiles.

The patch to solve this would be this:

```
### Eclipse Workspace Patch 1.0
#P 6129c
Index: rules/annotations/oo_references.rules
===================================================================
--- workspace.java.open.client/6129c/rules/annotations/oo_references.rules    (revision 4045)
+++ workspace.java.open.client/6129c/rules/annotations/oo_references.rules    (working copy)
@@ -457,6 +457,12 @@
             </rule>
             <action>ref.setType(prog.func_poly)</action>
             <action>ref.putAnnotation("oldtype", #(long) prog.kw_dyn_invk)</action>
+            <rule>isNote("qualified") and
+                  not (upPath("ASSIGNMENT/EXPRESSION/OBJECT_INVOCATION") and this.indexPos == 1)
+               <action>
+                  ref.putAnnotation("casttype", sprintf("object&lt;%s&gt;", getNoteString("full-java-class"))
)
+               </action>
+            </rule>
             <action>ref.putAnnotation("builtin", true)</action>
             <action>ref.putAnnotation("returnsunknown", true)</action>

Index: src/com/goldencode/p2j/uast/ClassDefinition.java
===================================================================
--- workspace.java.open.client/6129c/src/com/goldencode/p2j/uast/ClassDefinition.java    (revision 4045)
+++ workspace.java.open.client/6129c/src/com/goldencode/p2j/uast/ClassDefinition.java    (working copy)
@@ -1963,7 +1965,7 @@
          MethodSearchResult found = lookupMethodWorker(mname, signature, access, isStatic, internal, node);
          MemberData         mdat    = (found == null) ?  null : found.data;

-         if (mdat != null)
+         if (mdat != null && !(found != null && found.isDynamic()))
          {
             // the type may have been set incorrectly based on guessing, since the parameter
             // signature was not yet available; force it to the correct value here
@@ -2090,6 +2092,33 @@
                {
                   node.putAnnotation("dynamic-classname", this.name);
                }
+
+               if (mdat != null && mdat.type == OO_METH_CLASS)
+               {
+                  if (mdat.qname == null)
+                  {
+                     String err =
+                        String.format("Missing class name for %s!", node.getDescriptiveTokenText());
+                     throw new RuntimeException(err, new NoViableAltException((AnnotatedAst) node));
+                  }
+                  else
+                  {
+                     node.putAnnotation("qualified", mdat.qname.toLowerCase());
+
+                     if (mdat.retType != null && mdat.retType.containsKey("generic-type-parameter"))
+                     {
+                        String gtype = (String) mdat.retType.get("generic-type-parameter");
+
+                        node.putAnnotation("generic-type-parameter", gtype);
+                     }
+                     if (mdat.retType != null && mdat.retType.containsKey("generic-type-is-primitive"))
+                     {
+                        boolean prim = (boolean) mdat.retType.get("generic-type-is-primitive");
+
+                        node.putAnnotation("generic-type-is-primitive", prim);
+                     }
+                  }
+               }
             }
             else
             {
@@ -4080,12 +4109,16 @@
       {
          MethodSearchResult dyn = new MethodSearchResult();
```

```
-          // POLY arguments activates runtime/dynamic mode
+          // POLY arguments activates runtime/dynamic mode, but in OpenEdge the last match is returned always!
           if (polyArgs)
           {
              if (debug)
                 System.out.println("\n\n---------fuzzyMethodLookup DYNAMIC POLY!---------\n");

+             MatchMetrics match = list.get(list.size() - 1);
+
+             dyn = new MethodSearchResult(match.data, match.overrides, true);
+
              return dyn;
           }

@@ -5088,14 +5121,17 @@

       /** Type overrides for fuzzy matches. */
       final private String[] overrides;
+
+      final private boolean dynamic;

       /**
        * Create an dynamic invocation instance.
        */
       MethodSearchResult()
       {
-          data      = null;
-          overrides = null;
+          data         = null;
+          overrides    = null;
+          this.dynamic = false;
       }

       /**
@@ -5108,6 +5144,20 @@
          this.data      = data;
          this.overrides = overrides;
+         this.dynamic   = false;
+      }
+
+      /**
+       * Create an instance using the given result.
+       *
+       * @param    data
+       *           The result of the search.
+       */
+      MethodSearchResult(MemberData data, String[] overrides, boolean dynamic)
+      {
+         this.data      = data;
+         this.overrides = overrides;
+         this.dynamic   = dynamic;
       }

       /**
@@ -5117,7 +5167,7 @@
        */
       boolean isDynamic()
       {
-          return data == null;
+          return data == null || dynamic;
       }

       /**
```

where:

- if more than one match is found and there are POLY arguments, assume as 'found' the last one in the list
- this will still mark the call as 'dynamic'
- for these kind of calls, force a cast to the resolved return type, to allow chained calls.

The conversion will look like:

```
ObjectOps.invokeStandalone(((object<com.goldencode.testcases.src.oo.Ctest>) ObjectOps.invoke(l, "m1", "I", h.u
nwrapDereferenceable().dereference("f1")))  // l:m1(h::f1) - dynamic call
                           .ref().m2(new integer(h.unwrapDereferenceable().dereference("f2"))),   // :m2(h::f2
) - direct call, a single match
                           "m1", "I", h.unwrapDereferenceable().dereference("f3"));               // :m1(h::f3
) - dynamic call
```

#3 - 04/24/2023 02:11 PM - Constantin Asofiei

Marian, we need more tests to determine how OE behaves in this case. The problem here is when there are multiple overload methods which match a chained call: OE seems to determine the return type of the call from the last method in the overload list (so it matters the actual position of the method in the method definition list), while the actual target of the call will be determined in a dynamic way (as the argument's real type is not known until runtime).

I don't think it matters the argument list (it can be a single argument), what matters is the call is POLY (like using when using ::) and multiple target methods can be resolved.

These tests need to explore:

- more than two overload methods (like m1) - what happens if there is a 'middle' overload which has a different return type than the last one?
- what happens if there is i.e. progress.lang.object and oo.Ctest return type for the 2 m1(...) definitions, and the chained call targets a method common to both types (like toString() in l:m1(h::f1):toString())?
- what happens if the resolved methods are from both the object's declared type and from super-classes, only from super-classes, interfaces, etc? Here I mean how would OE order the methods from the super-classes/interfaces, to find the 'last one in the possible targets', assuming this is the rule to be applied?
- what happens in a case like l:m1(h::f1):m2(h::f1), where there are defined in l's declared type:
  - oo.Iface1 m1(...)
  - oo.Iface2 m1(...)
  - both Iface1 and Iface2 have a m2(...) definition (maybe with different single parameter type) - will OE target Iface1, Iface2, let the entire chain be a dynamic call, or just not compile?
- previous case can be expanded to l:m1(h::f1):m2(h::f1):m3(), where m3() is only defined either in Iface1 or Iface2

If you have questions or other cases which can be added/discussed, please let me know.

#4 - 04/25/2023 05:39 AM - Marian Edu

Constantin, will do... as a side note, imho this should not have been allowed in 4GL, it completely break any strong typing from OO 4GL :(

#5 - 04/25/2023 07:19 AM - Constantin Asofiei

Created task branch 7299a from trunk rev 14550. The patch is in rev 14551. Greg, please review.

#6 - 04/25/2023 12:22 PM - Greg Shah

Code Review Task Branch 7299a Revision 14551

No objections.

#7 - 04/25/2023 12:24 PM - Greg Shah

I think this fix is needed urgently by one project. Should we merge this to trunk and 7156a now and then check the tests later?

#8 - 04/25/2023 12:27 PM - Constantin Asofiei

Greg Shah wrote:

> I think this fix is needed urgently by one project. Should we merge this to trunk and 7156a now and then check the tests later?

I'm running a full parse of that application (including #7301 changes), if that passes I'll push both to 7156a. For trunk merge, I need more testing.

#9 - 04/25/2023 12:36 PM - Marian Edu

Constantin Asofiei wrote:

> Marian, we need more tests to determine how OE behaves in this case. The problem here is when there are multiple overload methods which match a chained call: OE seems to determine the return type of the call from the last method in the overload list (so it matters the actual position of the method in the method definition list), while the actual target of the call will be determined in a dynamic way (as the argument's real type is not known until runtime).
>
> I don't think it matters the argument list (it can be a single argument), what matters is the call is POLY (like using when using ::) and multiple target methods can be resolved.

So far I can certainly confirm this, the return type of the last method wins as far as the compiler is concerned but the correct method is resolved at runtime based on the input parameter data type.

But, I would say this is probably not an error you need to be concerned with since - presumably - you only convert valid 4GL code, isn't it?

#10 - 04/25/2023 12:48 PM - Greg Shah

We convert compilable code, which this is. hIt is used heavily in a customer project so we need to get it right.

#11 - 04/26/2023 12:04 PM - Greg Shah

Marian: Please have your team review the documentation on method overloading in #3751-492 and look at the code in oo/params/*.p for the testcases I used to determine these rules.

#12 - 04/28/2023 11:31 AM - Greg Shah

> I think this fix is needed urgently by one project. Should we merge this to trunk and 7156a now and then check the tests later?

I'm running a full parse of that application (including #7301 changes), if that passes I'll push both to 7156a. For trunk merge, I need more testing.

Do we have a result of this testing that I can report?

#13 - 05/02/2023 10:45 AM - Greg Shah

What is the status of this?

#14 - 05/02/2023 10:47 AM - Constantin Asofiei

Greg Shah wrote:

> What is the status of this?

This is in 7156a for the customer. What's left is Marian's tests, validate current impl and implement any other found behavior.

#15 - 06/05/2023 11:58 AM - Constantin Asofiei

Rebased 7299a/14551 from trunk rev 14612 - new rev 14613.

7299a rev 14614 replaces upPath with the token list version.

#16 - 06/05/2023 01:12 PM - Greg Shah

> Marian: Please have your team review the documentation on method overloading in #3751-492 and look at the code in oo/params/*.p for the testcases I used to determine these rules.

What is the status of writing these tests?

#17 - 06/06/2023 07:57 AM - Marian Edu

Greg Shah wrote:

> Marian: Please have your team review the documentation on method overloading in #3751-492 and look at the code in oo/params/*.p for the testcases I used to determine these rules.

> What is the status of writing these tests?

Daniel did some work on this one, will review those and merge them back in testcases tomorrow but I think he didn't cover all possible combinations so will have to see what is there.

#18 - 06/09/2023 02:58 PM - Constantin Asofiei

Rebased 7299a/14613 from trunk rev 14623 - new rev 14625 .

#20 - 07/03/2023 07:10 AM - Constantin Asofiei

The same rules used at the conversion to determine the target for a direct call needs to be implemented in ControlFlowOps.resolveLegacyEntry, when the target is invoked via reflection.

#21 - 07/07/2023 11:41 AM - Greg Shah

> Marian: Please have your team review the documentation on method overloading in #3751-492 and look at the code in oo/params/*.p for the testcases I used to determine these rules.

> What is the status of writing these tests?

> Daniel did some work on this one, will review those and merge them back in testcases tomorrow but I think he didn't cover all possible combinations so will have to see what is there.

Marian: Do you have an update on this? It is blocking some critical customer testing and we need to get this fix moved along but we are dependent upon more tests from your side.

#22 - 07/10/2023 07:00 AM - Marian Edu

Greg Shah wrote:

> Marian: Please have your team review the documentation on method overloading in #3751-492 and look at the code in oo/params/*.p for the testcases I used to determine these rules.

Greg, missed somehow your comments on oo/params, those tests were not refactored as ABLUnit as far as I can see as it looks those were added after we've started the 'migration'. Do you want us to do the conversion for those as well? There might be other tests that we've missed, I need to go back through history to see what was added outside of our team if we need to convert those too.

> Marian: Do you have an update on this? It is blocking some critical customer testing and we need to get this fix moved along but we are dependent upon more tests from your side.

The tests for method resolution for in/out/in-out parameters for both primitive data types and OO are in tests/oo/method_resolve and already available on testcases project.

#23 - 07/10/2023 07:14 AM - Greg Shah

> those tests were not refactored as ABLUnit as far as I can see as it looks those were added after we've started the 'migration'. Do you want us to do the conversion for those as well? There might be other tests that we've missed, I need to go back through history to see what was added outside of our team if we need to convert those too.

Yes, please do that. These are quite important as they were used to determine the method overloading rules I documented in #3751-492 and upon which we based our implementation.

#24 - 07/11/2023 07:45 AM - Marian Edu

Greg Shah wrote:

> those tests were not refactored as ABLUnit as far as I can see as it looks those were added after we've started the 'migration'. Do you want us to do the conversion for those as well? There might be other tests that we've missed, I need to go back through history to see what was

added outside of our team if we need to convert those too.

> Yes, please do that. These are quite important as they were used to determine the method overloading rules I documented in #3751-492 and upon which we based our implementation.

Greg, the tests in oo/params were converted to ABLUnit and you can find them in tests/oo/params - some of the supporting code were moved in tests/oo/support, hope I've managed to keep the original intent intact... committed in testcases rev #1451 on xref.

#25 - 07/17/2023 02:40 PM - Greg Shah

- Assignee set to Constantin Asofiei

#26 - 07/18/2023 05:18 AM - Constantin Asofiei

- Status changed from New to WIP

~~Created task branch 7229b from trunk rev 14661.~~

#27 - 07/18/2023 07:51 AM - Constantin Asofiei

Constantin Asofiei wrote:

> ~~Created task branch 7229b from trunk rev 14661.~~

This branch was physically removed, 7299a already exists. Rebased 7299a from trunk rev 14661 - new rev 14663.

#28 - 07/18/2023 09:29 AM - Constantin Asofiei

Marian, there is something wrong with the TestLastMethodReturnTypeChainedCall.testLastOverloadHasDifferentReturnType test:

```
        define variable auc as String no-undo.

        // the return type of all m3 methods is wrongly assumed to be the
        // return type of the last overload of m3 (OpenEdge.Core.String) and,
        // thus, this compiles and runs with no errors
        auc = ctest:m3(tt1Handle::finteger) no-error.
        support.test.AssertExt:NotErrorNotWarning().
        Assert:Equals("tests.oo.support.CTest", auc:GetClass():TypeName).
```

I've tested (via a standalone example, not unit tests) in Openedge, and the runtime resolves tests.oo.support.CTest m3(input i3 as integer):, and not OpenEdge.Core.String m3(input i3 as character): (the last method definition).

Does the test pass in OpenEdge?

#29 - 07/18/2023 11:26 AM - Marian Edu

Constantin Asofiei wrote:

> Marian, there is something wrong with the TestLastMethodReturnTypeChainedCall.testLastOverloadHasDifferentReturnType test:
> [...]
> I've tested (via a standalone example, not unit tests) in Openedge, and the runtime resolves tests.oo.support.CTest m3(input i3 as integer):, and not OpenEdge.Core.String m3(input i3 as character): (the last method definition).

Yes, this actual the quirks... if you change the type of the variable to be CTest instead of String the code won't compile complaining about the wrong data type.

```
using Progress.Lang.*.
using OpenEdge.Core.Assert from propath.
using tests.oo.support.CTest.
using OpenEdge.Core.String from propath.

block-level on error undo, throw.

{tests/oo/support/ttCTest.i}

define variable ctest     as CTest.
define variable tt1Handle as handle.
define variable obj       as Progress.Lang.Object no-undo.

create tt1.
tt1Handle = buffer tt1:handle.
ctest = new CTest().

// if you define it as CTest this doesn't compile in 4GL
define variable auc as CTest no-undo.

//define variable auc as String no-undo.

        // the return type of all m3 methods is wrongly assumed to be the
        // return type of the last overload of m3 (OpenEdge.Core.String) and,
        // thus, this compiles and runs with no errors
auc = ctest:m3(tt1Handle::finteger) no-error.

if valid-object(auc) then
    message auc:GetClass():TypeName view-as alert-box.
```

> Does the test pass in OpenEdge?

Yes it does, also the standalone example works but we see this as a quirk - the method resolution at compile time is different from what happens at runtime, even more a different kind of object is assigned to that variable, it definitively isn't a String but a CTest... this is a plain bug imho.

#30 - 07/18/2023 11:38 AM - Constantin Asofiei

Please run this on your side:

```
using Progress.Lang.*.
using OpenEdge.Core.Assert from propath.
using oo.CTest.
using OpenEdge.Core.String from propath.

block-level on error undo, throw.

def temp-table tt1
    field fhandle as handle
    field flogical as logical
    field finteger as integer
    field fint64 as int64
    field fdecimal as decimal
    field fdate as date
    field fdatetime as datetime
    field fdatetime-tz as datetime-tz
    field fcharacter as character
    field frecid as recid
    field frowid as rowid
    field fcomponent-handle as component-handle.

define variable ctest     as CTest.
define variable tt1Handle as handle.
define variable obj       as Progress.Lang.Object no-undo.

create tt1.
tt1Handle = buffer tt1:handle.
ctest = new CTest().

define variable auc as String no-undo.

auc = ctest:m3(tt1Handle::finteger) no-error.

message error-status:error error-status:get-message(1).
```

I get this:

```
yes A variable of class 'oo.CTest' cannot be assigned to a variable of class 'OpenEdge.Core.String'. (13448)
```

#31 - 07/18/2023 11:46 AM - Marian Edu

Constantin Asofiei wrote:

> Please run this on your side:

Not sure, we have the CTest in tests.oo.support so had to adjust the using statement but when I ran it it gives me no and the actual instance assigned to that variable is a CTest... this is on 12.2 btw.

But even your result is still wrong, because that should have been a compile time error not runtime - as said, try to change the data type of auc variable to CTest instead of String and see if it compiles or you get a compiler error.

#32 - 07/18/2023 11:59 AM - Constantin Asofiei

Marian Edu wrote:

> Constantin Asofiei wrote:
>
> > Please run this on your side:
>
> Not sure, we have the CTest in tests.oo.support so had to adjust the using statement but when I ran it it gives me no and the actual instance assigned to that variable is a CTest... this is on 12.2 btw.

I'm running with 11.6.3.

> But even your result is still wrong, because that should have been a compile time error not runtime - as said, try to change the data type of auc variable to CTest instead of String and see if it compiles or you get a compiler error.

This runs fine in 11.6.3, with auc being a String. If I change auc to CTest, then yes, is a compile error.

#33 - 07/18/2023 12:08 PM - Marian Edu

Constantin Asofiei wrote:

> I'm running with 11.6.3.

Sadly the oldest we have available is 11.7 and that will probably e removed next year from our subscription as we only have the last version and the previous LTE one available into the package :(

> This runs fine in 11.6.3, with auc being a String. If I change auc to CTest, then yes, is a compile error.

It might run fine but it's still wrong, this should not happen - the copile time resolution should see the variable needs to be a Ctest not a String, it doesn't and then it complains at runtime that the data type doesn't match. If you change it so it match the data type then you can't compile the code :)

#34 - 07/18/2023 12:11 PM - Constantin Asofiei

Marian Edu wrote:

> It might run fine but it's still wrong, this should not happen - the copile time resolution should see the variable needs to be a Ctest not a String, it doesn't and then it complains at runtime that the data type doesn't match. If you change it so it match the data type then you can't compile the code :)

Well, we need to make it work as it does in OE, even if it doesn't make sense in a programing language point of view. Please run the test in #7299-30 in your OpenEdge and let me know if you see the same error message I see.

#35 - 07/18/2023 01:15 PM - Marian Edu

Constantin Asofiei wrote:

> Please run the test in #7299-30 in your OpenEdge and let me know if you see the same error message I see.

Thought I did that, if I run the code (with the using stuff updated to point to tests.oo.support instead of oo) it runs with no error - meaning it display no as a message, it doesn't give any runtime error.

#36 - 07/18/2023 01:17 PM - Constantin Asofiei

Marian Edu wrote:

> Constantin Asofiei wrote:
>
> > Please run the test in #7299-30 in your OpenEdge and let me know if you see the same error message I see.
>
> Thought I did that, if I run the code (with the using stuff updated to point to tests.oo.support instead of oo) it runs with no error - meaning it display no as a message, it doesn't give any runtime error.

That is really weird. Can you test also on 11.7?

#37 - 07/18/2023 01:17 PM - Constantin Asofiei

Constantin Asofiei wrote:

> Marian Edu wrote:
>
> > Constantin Asofiei wrote:
> >
> > > Please run the test in #7299-30 in your OpenEdge and let me know if you see the same error message I see.
> >
> > Thought I did that, if I run the code (with the using stuff updated to point to tests.oo.support instead of oo) it runs with no error - meaning it display no as a message, it doesn't give any runtime error.
>
> That is really weird. Can you test also on 11.7?

Also, are you running with .r, via the procedure editor or command line?

#38 - 07/18/2023 01:19 PM - Constantin Asofiei

The most important part missing in FWD I think is this: at compile time, OpenEdge hard-links the call to invoke the exact resolved method. So a call like this:

```
ctest:mdecimal(tt1Handle::fdatetime).
```

will produce an ERROR condition like this:

```
Routine testDecimalDatetime tests.oo.method_resolve.type_conversion.TestTtFieldConversion sent called routine
mdecimal tests.oo.support.CTest mismatched parameters. (2570)
```

and a call like this:

```
dynamic-invoke(ctest, "mdecimal", tt1handle::fdatetime).
```

will produce an ERROR condition like this:

```
Could not locate method 'mdecimal' with matching signature in class 'oo.CTest'. (14457)
```

So we can't emulate this POLY arguments call at runtime, via the DYNAMIC-INVOKE. We need a 'invokePoly' API which (somehow) knows exactly which method to execute, using some artifact emitted from the conversion.

#39 - 07/18/2023 02:23 PM - Constantin Asofiei

I was thinking that instead of wrapping the POLY expr in a constructor with the expected type, emit a Parameter.forType(integer.class, <expr>) which checks that the expression's type exactly matches. But this doesn't have any knowledge about the method being called, to show the proper error.

#40 - 07/19/2023 10:27 AM - Constantin Asofiei

I've ran tests.oo.method_resolve.TestSuiteMethodResolve in OpenEdge 11.6.3 and the only test that fails is this:

```
    @Test.
```

```
    method public void testLastOverloadHasDifferentReturnType():
        define variable auc as String no-undo.

        // the return type of all m3 methods is wrongly assumed to be the
        // return type of the last overload of m3 (OpenEdge.Core.String) and,
        // thus, this compiles and runs with no errors
        auc = ctest:m3(tt1Handle::finteger) no-error.
        support.test.AssertExt:NotErrorNotWarning().
        Assert:Equals("tests.oo.support.CTest", auc:GetClass():TypeName).
    end method.
```

Marian: please double-check that you are using the same source code from xfer testcases; or maybe some .r program was left behind? I'm really hopping that this is **not** a OE configuration difference.

I'll run the other OO tests and I'll let you know.

#41 - 07/19/2023 11:23 AM - Constantin Asofiei

Marian, from the tests/oo @TestSuite files I could run in OpenEdge, this one I had to manually change:

```
@TestSuite(procedures="
        tests/oo/constructor_destructor_study/TestConstructor.p,
        tests/oo/constructor_destructor_study/TestSetUnknownProcedure.p").
class tests.oo.constructor_destructor_study.TestSuiteClass:
```

on xfer being this:

```
@TestSuite(procedures="
        tests.oo.constructor_destructor_study.TestConstructor,
        tests.oo.constructor_destructor_study.TestUnknownProcedure").
```

#43 - 09/28/2023 11:25 AM - Constantin Asofiei

There is something which looks like a bug in the OpenEdge method resolution. Assume there is a method like this:

```
   method public void m3(input i3 as oo.Foo).
      message "m3" i3.
   end.
```

and a call like this:

```
def temp-table tt1 field f3 as progress.lang.object.
create tt1.
def var i3 as progress.lang.object.

// these calls will not be performed (failure in method lookup)
tt1.f3 = new progress.lang.object().
dynamic-invoke(l, "m3", h::f3) no-error.
dynamic-invoke(l, "m3", tt1.f3) no-error.
l:m3(h::f3) no-error.
i3 = new progress.lang.object().
dynamic-invoke(l, "m3", i3) no-error.
i3 = ?.
dynamic-invoke(l, "m3", i3) no-error.

// these calls will be performed
tt1.f3 = ?.
dynamic-invoke(l, "m3", h::f3) no-error.
l:m3(h::f3) no-error.
```

For some reason, if the buffer field is unknown and referenced via :: attribute, the method call is allowed. If the field is read directly as tt1.f3 and passed to dynamic-invoke, then the call is not allowed.

My assumption is that h::f3 doesn't set the type of the object returned (which would be progress.lang.object, even if unknown), only direct field access (tt1.f3) does it.

#44 - 09/28/2023 11:27 AM - Constantin Asofiei

Constantin Asofiei wrote:

> My assumption is that h::f3 doesn't set the type of the object returned (which would be progress.lang.object, even if unknown), only direct field

> access (tt1.f3) does it.

The same happens with BUFFER-FIELD:BUFFER-VALUE.

#45 - 09/28/2023 11:57 AM - Constantin Asofiei

The current changes are in 7299a rev 14758. I'll reconvert some projects and check real scenarios.

This adds ObjectOps.invokePoly, which receives both the target type on which to perform the method lookup, and the compatible expected return type, in cases when this is on the left-side of a chained OO call. Also, it refactors the fuzzy method matching from conversion to be used by runtime, too.

What is missing, for dynamic invoke cases:

- runtime signature for DATASET, TEMP-TABLE and BUFFER parameters
- I think the runtime matching must use the 'exact method matching' algorithm from conversion, too, beside the fuzzy matching. At least, 'exact method matching' must be done in the entire hierarchy, before going into the 'fuzzy matching' - I think this can be solved by getting the overloads from the entire hierarchy and not just the target type, I'll look into this.

#46 - 09/28/2023 12:24 PM - Constantin Asofiei

Constantin Asofiei wrote:

> ... At least, 'exact method matching' must be done in the entire hierarchy, before going into the 'fuzzy matching' - I think this can be solved by getting the overloads from the entire hierarchy and not just the target type, I'll look into this.

This works with the standalone and unit tests. Fixed in 14759.

#47 - 09/29/2023 05:07 PM - Greg Shah

Code Review Task Branch 7299a Revisions 14756 through 14763

Overall, it looks good. The refactoring made the review a bit harder so I hope I didn't miss anything.

The one thing that looked wrong was this code in ClassDefinition.annotateMethodCall():

```
                    for (int i = 0; i < mdat.signature.length; i++)
                    {
                       ParameterKey pkey = mdat.signature[i];
                       String ptype = pkey.toJava();

                       node.putAnnotation("dynamic-poly-sig", ptype, -1);
                    }
```

Doesn't it just leave behind one annotation of the last element of the signature?

#48 - 10/07/2023 01:57 PM - Constantin Asofiei

- % Done changed from 0 to 90

7299a was rebased and rev 14777/14778 contains fixes to add ObjectOps.poly(Object)Arg wrapper for cases when a direct method call gets converted to a dynamic-poly method call (not a direct Java method call).

I'm running conversion testing again, if this is OK I'll go with runtime testing.

#49 - 10/09/2023 09:22 AM - Constantin Asofiei

- Status changed from WIP to Review

- % Done changed from 90 to 100

Greg, please review 7299a starting from rev 14777 to 14784 (i.e. compared with 14776, the last reviewed revision).

Conversion is OK and also app testing.

#50 - 10/10/2023 03:17 PM - Greg Shah

Code Review Task Branch 7299a Revisions 14777 through 14784

No objections.

You can merge to trunk.

#51 - 10/11/2023 03:19 AM - Constantin Asofiei

7299a was merged to trunk rev 14770 and archived.

#52 - 10/11/2023 07:54 AM - Greg Shah

- Status changed from Review to Test

#53 - 10/13/2023 01:38 PM - Constantin Asofiei

There is a regression in the conversion of #7156 project: a LPARENS can enclose an argument. I've created 7299b from trunk rev 14773 and the fix is in rev 14774.

#54 - 10/13/2023 03:34 PM - Greg Shah

Code Review Task Branch 7299b Revison 14774

The change is good.

#55 - 10/13/2023 03:34 PM - Greg Shah

It seems pretty safe. I'm OK with you merging to trunk.

#56 - 10/14/2023 06:32 AM - Constantin Asofiei

Greg Shah wrote:

> It seems pretty safe. I'm OK with you merging to trunk.

I'll delay this until I get the customer's go ahead that their app has completely converted and compiled (probably on Tuesday).

#57 - 10/19/2023 07:15 AM - Constantin Asofiei

7299b rev 14776 will show linkage errors when resolving Java class names during conversion. The problem here is that the class is in the Java classpath, but can't be loaded in the JVM.

#58 - 10/19/2023 10:05 AM - Constantin Asofiei

Greg, please review 7299b (revs 14778 - 14780) - this contains regression fixes for #6501-359 and #7929

#59 - 10/20/2023 10:15 AM - Greg Shah

Code Review Task Branch 7299b Revisions 14778 through 14780

The changes are good.

#60 - 10/20/2023 11:34 AM - Constantin Asofiei

Branch 7299b was merged to trunk rev 14779 and archived

#61 - 10/23/2023 11:16 AM - Constantin Asofiei

Created task branch 7299c from trunk rev 14786.

7299c rev 14787 fixed another regression in trunk rev 14770, for unknown type instances used as arguments for dynamic calls.

#62 - 10/23/2023 03:02 PM - Greg Shah

Code Review Task Branch 7299c Revision 14787

No objection. This seems pretty safe. Do you think it can be merged to trunk now?

#63 - 10/23/2023 03:07 PM - Constantin Asofiei

7299c was merged to trunk as rev. 14787 and archived.

#64 - 10/26/2023 02:31 PM - Constantin Asofiei

Created task branch 7299d from trunk rev 14791.

7299d rev 14792 contains a regression fix for static poly calls - the string representation of the qualified legacy class name is not needed, and having a lowercase qualified name actually poses problems for case-sensitive apps. The change is not yet done, I need to remove the 'clsName' parameter completely, which will require full conversion.

I'm waiting some results from testing on a Linux machine this case:

```
class oo.Foo:
   method public static void m1().
      message "found me!".
   end.
end.
```

and run it like:

```
dynamic-invoke("oo.foo", "m1").
```

Note the lowercase class name - if OO resolves this on a case-sensitive machine, then the SourceNameMapper lookup needs to consider .cls lookup as special.

#66 - 10/30/2023 03:04 PM - Constantin Asofiei

The fixes for #7299-64 and #7976 are in 7299d rev 14793 - please review.

I'll need to go through conversion testing again with this fix.

#67 - 10/30/2023 03:48 PM - Greg Shah

Code Review Task Branch 7299d Revisions 14792 and 14793

In oo_references.rules line 453, don't we need to remove the double quotes around the classname?

Otherwise everything looks good.

#68 - 10/31/2023 05:41 AM - Constantin Asofiei

Greg Shah wrote:

> In oo_references.rules line 453, don't we need to remove the double quotes around the classname?

Yes, I've removed them in rev 14794.

#69 - 10/31/2023 10:57 AM - Greg Shah

Code Review Task Branch 7299d Revision 14794

No objections.

#70 - 10/31/2023 10:57 AM - Greg Shah

Is any more testing needed before merge?

#71 - 10/31/2023 10:57 AM - Constantin Asofiei

Greg Shah wrote:

> Is any more testing needed before merge?

No, it can be queued for merge.

#72 - 10/31/2023 11:10 AM - Greg Shah

It can merge after 7650a.

#73 - 10/31/2023 11:35 AM - Constantin Asofiei

Greg Shah wrote:

> It can merge after 7650a.

I was wrong - I need to do a round of conversion testing with 7299d, I haven't done this yet.

#74 - 10/31/2023 01:57 PM - Constantin Asofiei

7299d was rebased from trunk rev 14799 - new rev 14802.

#75 - 11/01/2023 06:09 AM - Constantin Asofiei

7299d was merged to trunk as revision 14809 and archived.