Base Language - Bug #6993

OO: incorrect field names in DataTypeEnum.java

12/13/2022 06:24 AM - Vladimir Tsichevski

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

Description

History

#1 - 12/13/2022 06:31 AM - Vladimir Tsichevski

The fields in OpenEdge.Core.DataTypeEnum are named after the corresponding Progress base data types, like character, decimal etc.
FWD runtime contains the corresponding Java source, created **manually**.
Current conversion rules imply that if the legacy class field name matches 4gl type name, an underscore must be appended to the converted Java field name.
The field names in DataTypeEnum.java do **not** conform this rule. As the result, converted Java code which uses this legacy class do not compile.

#2 - 12/13/2022 06:42 AM - Greg Shah

- Project changed from Runtime Infrastructure to Base Language

Please do fix this.

#3 - 12/13/2022 06:45 AM - Vladimir Tsichevski

- Status changed from New to WIP

#5 - 12/13/2022 06:59 AM - Vladimir Tsichevski

- Status changed from WIP to Review

- % Done changed from 0 to 100

Fixed in 3821c rev. 14438.

#6 - 12/13/2022 07:13 AM - Marian Edu

Vladimir Tsichevski wrote:

> The fields in OpenEdge.Core.DataTypeEnum are named after the corresponding Progress base data types, like character, decimal etc.
> FWD runtime contains the corresponding Java source, created **manually**.

All classes in OpenEdge and Progress packages were created **manually**, this is the expected approach since we do not have the 4GL code to convert, what am I missing?

> Current conversion rules imply that if the legacy class field name matches 4gl type name, an underscore must be appended to the converted Java field name.

As far as I understand conversion rules applies to code that is being converted, if there are rules that must be followed when **manually** implementing corresponding 4GL base classes those were added later and were not communicated so all the code base for current OO implementation certainly does not follow those rules.

> The field names in DataTypeEnum.java do **not** conform this rule. As the result, converted Java code which uses this legacy class do not compile.

Why isn't that compiling exactly? There is a proper @LegacySignature annotation with the proper `name`?

#7 - 12/13/2022 07:24 AM - Vladimir Tsichevski

Marian Edu wrote:

> All classes in OpenEdge and Progress packages were created **manually**, this is the expected approach since we do not have the 4GL code to convert, what am I missing?

No, you are perfectly correct. I see no problem here.

> As far as I understand conversion rules applies to code that is being converted,

Correct

> if there are rules that must be followed when **manually** implementing corresponding 4GL base classes those were added later and were not communicated so all the code base for current OO implementation certainly does not follow those rules.

But the names of public fields and methods in the Java sources created manually must be that expected by conversion rules.

> > The field names in DataTypeEnum.java do **not** conform this rule. As the result, converted Java code which uses this legacy class do not compile.
>
> Why isn't that compiling exactly? There is a proper @LegacySignature annotation with the proper `name`?

No, it is the Java sources which will not compile.

Example:

The following test referring the OpenEdge.Core.DataTypeEnum:Character

```
@Test(expected = "OpenEdge.Core.AssertionFailedError").
METHOD PUBLIC VOID handleIsTypeMustFail():
  Assert:IsType(handle1, OpenEdge.Core.DataTypeEnum:Character).
END METHOD.
```

converts to:

```
   @LegacySignature(type = Type.METHOD, name = "handleIsTypeMustFail")
   @Test(expected = "OpenEdge.Core.AssertionFailedError")
   public void handleIsTypeMustFail()
   {
      internalProcedure(TestAssert.class, this, "handleIsTypeMustFail", new Block((Body) () ->
      {
         com.goldencode.p2j.oo.core.Assert.isType(handle1, com.goldencode.p2j.oo.core.DataTypeEnum.character_)
;
                                                                                note the trailing underscore here ^^^
      }));
   }
```

Note: the name if the field referenced is character_, not character.

#8 - 12/13/2022 07:30 AM - Marian Edu

Vladimir Tsichevski wrote:

> No, it is the Java sources which will not compile.
> Note: the name if the field referenced is character_, not character.

I understand that but as I've said all the current OO implementation was **manually** created and does not comply to the conversion rule so imho this should be fixed in conversion - there is a LegacySignature annotation that should be used. When there was a conflict with a Java keyword the property/method might need to be altered and the mapping is done exactly through that annotation or that is not used anymore? If conversion rules are changing the existing OO code base will need to be updated to comply with those changing rules, what is the point of using legacy annotations then?

#9 - 12/13/2022 02:04 PM - Greg Shah

Our OO implementation is much more advanced in vranch 6129b than in 3821c. In 6129b, at this time we only use LegacySignature for 2 things during conversion:

- For loading a ClassDefinition instance from a built-in OO Java class.

- Calculating gap-marking.

The converted method name should come from the ClassDefinition instance, so it is possible that in 6129b this Java compilation issue does not exist.

Vladimir: Please check your failing scenario in 6129b and let us know what you find.

#10 - 12/13/2022 03:43 PM - Vladimir Tsichevski

Greg Shah wrote:

> Vladimir: Please check your failing scenario in 6129b and let us know what you find.

I cannot convert with that revision. The class to convert:

```
CLASS unittests.TestDataTypeEnum:
  DEFINE PRIVATE VARIABLE dte AS OpenEdge.Core.DataTypeEnum NO-UNDO.

  METHOD PUBLIC VOID test():
    dte = OpenEdge.Core.DataTypeEnum:Character.
  END METHOD.

END CLASS.
```

causes the following conversion error:

```
     [java]    Lvl01 DONE:  ./abl/unittests/TestDataTypeEnum.cls
     [java]
     [java] --------------------------------------------------------------------------------
     [java] Elapsed job time:  00:00:01
     [java]
     [java]
     [java] --------------------------------------------------------------------------------
     [java] Post-Parse Fixups
     [java] --------------------------------------------------------------------------------
     [java]
     [java] Elapsed job time:  00:00:00.064
     [java] ERROR:
     [java] java.lang.RuntimeException: ERROR!  Active rule could not be determined: Error loading configurati
on: fixups/post_parse_fixups; XML path: /cfg/include
     [java]     at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:1098)
     [java]     at com.goldencode.p2j.convert.TransformDriver.processTrees(TransformDriver.java:585)
     [java]     at com.goldencode.p2j.convert.TransformDriver.front(TransformDriver.java:259)
     [java]     at com.goldencode.p2j.convert.TransformDriver.executeJob(TransformDriver.java:985)
     [java]     at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1284)
     [java] Caused by: com.goldencode.p2j.cfg.ConfigurationException: Error loading configuration: fixups/post
_parse_fixups; XML path: /cfg/include
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.loadImpl(ConfigLoader.java:640)
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.load(ConfigLoader.java:543)
     [java]     at com.goldencode.p2j.pattern.PatternEngine.initialize(PatternEngine.java:1189)
     [java]     at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:1026)
     [java]     ... 4 more
     [java] Caused by: com.goldencode.p2j.cfg.ConfigurationException: Error loading configuration: common-prog
ress; XML path: /rule-set/func-library/function/variable
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.loadImpl(ConfigLoader.java:640)
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.include(ConfigLoader.java:1124)
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.processChildElements(ConfigLoader.java:727)
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.loadImpl(ConfigLoader.java:610)
     [java]     ... 7 more
     [java] Caused by: com.goldencode.p2j.cfg.ConfigurationException: Error initializing user variable ccn
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.variable(ConfigLoader.java:873)
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.processChildElements(ConfigLoader.java:707)
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.function(ConfigLoader.java:1174)
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.processChildElements(ConfigLoader.java:715)
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.library(ConfigLoader.java:1148)
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.processChildElements(ConfigLoader.java:731)
```

```
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.ruleSet(ConfigLoader.java:1102)
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.loadImpl(ConfigLoader.java:606)
     [java]     ... 10 more
     [java] Caused by: java.lang.ClassNotFoundException: com.goldencode.p2j.uast.ConvertedClassName
     [java]     at java.net.URLClassLoader.findClass(URLClassLoader.java:387)
     [java]     at java.lang.ClassLoader.loadClass(ClassLoader.java:418)
     [java]     at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:352)
     [java]     at java.lang.ClassLoader.loadClass(ClassLoader.java:351)
     [java]     at java.lang.Class.forName0(Native Method)
     [java]     at java.lang.Class.forName(Class.java:264)
     [java]     at com.goldencode.p2j.pattern.ConfigLoader.variable(ConfigLoader.java:849)
     [java]     ... 17 more
```

#11 - 12/13/2022 04:29 PM - Constantin Asofiei

Check your classpath. ConvertedClassName no longer exists in 6129b.

#12 - 12/13/2022 04:53 PM - Vladimir Tsichevski

Constantin Asofiei wrote:

> Check your classpath. ConvertedClassName no longer exists in 6129b.

Yes, I forgot to adjust p2j.cfg.xml, so rules from a different FWD branch were still used. Thank you!

#13 - 12/13/2022 04:54 PM - Vladimir Tsichevski

Greg Shah wrote:

> Vladimir: Please check your failing scenario in 6129b and let us know what you find.

Yes, in 6129b this #6993 issue does not exist.

#14 - 12/14/2022 07:32 AM - Greg Shah

Can you do something low effort to work around this issue until we have 6129b changes ready?

#15 - 12/14/2022 07:42 AM - Vladimir Tsichevski

Greg Shah wrote:

> Can you do something low effort to work around this issue until we have 6129b changes ready?

I think, we better just revert this change when the 6129b is merged into the trunk. In any case, we will not forget to do this, since the unit test code will become uncompilable otherwise.