Base Language - Bug #4455

Extent field on unsubscripted next-prompt not converted properly

12/09/2019 01:00 PM - Roger Borrello

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

Description

History

#1 - 12/09/2019 02:18 PM - Roger Borrello

Description

When next-prompt is passed an extent field of a shared frame, but no subscript, we generate an unresolved symbol in Java code. In Progress, the field defaults to the first element of the extent.

20191231 Update: Fixed in 4207a-11355

Testcase

Below is the testcase checked in as uast/input/extent_nosubscript_next-prompt.p.

Update: testcase has been modified, and a new one added.

- uast/input/extent_nosubscript_fld_next-prompt.p
- uast/input/extent_nosubscript_var_next-prompt.p

```
define variable mynum as integer initial 1 no-undo.
define shared frame person.

form
    schedule[1]
    schedule[2]
    mynum
    with frame person.

find first person
   where emp-num > 0 no-error.

do with frame person:
  update
    schedule[1]
    schedule[2]
    mynum
  editing:
    readkey.
    if input schedule[1] = "" then
    do:
      message "Blank".
      next-prompt schedule.
      next.
    end.
    if input schedule[1] = "" then
    do:
      message "Not blank".
      next-prompt schedule.
      message schedule[1].
    end.
```

```
  end.
end.
message "Done".
```

Generated Java

This results in the below snippet in ExtentNosubscriptNextPrompt.java :

```
                if (_isEqual(personFrame.getScheduleArray0(), ""))
                {
                   message("Blank");
                   personFrame.nextPrompt(personFrame.widgetSchedule());
                   next("editingLabel0");
                }
```

The below FillInWidget's are generated in ui/shared_frames/FramePerson_1.java:

```
   public FillInWidget widgetScheduleArray0();

   public FillInWidget widgetScheduleArray(NumberType parm);

   public FillInWidget widgetScheduleArray(double parm);

   public FillInWidget widgetScheduleArray1();

   public FillInWidget widgetMynum();
```

If the define shared frame person is removed, this is the result in ui/abl/ExtentNosubscriptNextPromptPerson.java:

```
   public FillInWidget widgetScheduleArray0();

   public FillInWidget widgetScheduleArray(NumberType parm);

   public FillInWidget widgetScheduleArray(double parm);

   public FillInWidget widgetScheduleArray1();

   public FillInWidget widgetMynum();

   public FillInWidget widgetSchedule();
```

#3 - 12/09/2019 04:54 PM - Roger Borrello

Roger Borrello wrote:

Description

When next-prompt is passed an extent field of a shared frame, but no subscript, we generate an unresolved symbol in Java code. In Progress, the field defaults to the first element of the extent.

Testcase

Below is the testcase checked in as uast/input/extent_nosubscript_next-prompt.p. Note that if you remove define shared frame person. the testcase is successful.

[...]

Generated Java

This results in the below snippet in ExtentNosubscriptNextPrompt.java :

[...]

The below FillInWidget's are generated in ui/shared_frames/FramePerson_1.java:

[...]

If the define shared frame person is removed, this is the result in ui/abl/ExtentNosubscriptNextPromptPerson.java:

[...]

Looking at convert/array_expansion.rules the code is checking for extent > 0 and descendant(prog.kw_for, 2) which I believe is where we also need to check for prog.kw_next_pmt so that the FIELD_CHAR can have an implicit LBRACKET/NUM_LITERAL added, with text="1".

Working on understanding how to do that...

#4 - 12/09/2019 05:00 PM - Constantin Asofiei

Roger, the public FillInWidget widgetSchedule(); line doesn't seem right, to be generated. You say that the first element in the array is the one which NEXT-PROMPT targets? If so, why not create a [1] extent for such a case, during annotations phase?

#5 - 12/09/2019 05:20 PM - Roger Borrello

Constantin Asofiei wrote:

> Roger, the public FillInWidget widgetSchedule(); line doesn't seem right, to be generated. You say that the first element in the array is the one which NEXT-PROMPT targets? If so, why not create a [1] extent for such a case, during annotations phase?

It definitely is wrong. It would add an additional FillInWidget which isn't coded in the FRAME.

I will look closer at that option. There is a lot more to annotations/array_expansion.rules that look like the appropriate way to add an implicit [1].

#6 - 12/09/2019 05:23 PM - Greg Shah

> Note that if you remove define shared frame person. the testcase is successful.

This is not the right way to say it. With the shared frame, the conversion does not abend but the results are very wrong/will not work properly.

> You say that the first element in the array is the one which NEXT-PROMPT targets? If so, why not create a [1] extent for such a case, during annotations phase?

Yes, this is the plan. It is not clear if it is best to remove the extra widgets after expansion or if we customize the expansion to only emit 1 widget in this case.

#7 - 12/09/2019 05:31 PM - Greg Shah

The primary issue is that next-prompt is not honored as a statement in which we do array expansion (because we always only expected a single var not an unsubscripted array ref).

> extent > 0 and descendant(prog.kw_for, 2) which I believe is where we also need to check for prog.kw_next_pmt

Yes, it is inside this location. But you must look at the AST for next-prompt to see if it matches existing logic or if we need new code.

> so that the FIELD_CHAR can have an implicit LBRACKET/NUM_LITERAL added, with text="1".

This should work, but then we probably want to bypass the normal array expansion processing because it would be doing many things that must be avoided. You will have to duplicate any annotations that would be there if the subscript was explicit.

#8 - 12/09/2019 05:39 PM - Constantin Asofiei

At core conversion time may be too late to do this, without having to go and 'hack' the frame generation to exclude this case, and maybe other parts. That's why for me is cleaner to fix the AST as if the [1] was always there, and let the annotations and conversion phases do their jobs unchanged.

#9 - 12/09/2019 05:46 PM - Greg Shah

I'm not suggesting doing it in core conversion. The alternative idea was to add next-prompt to the list of normal expansions and then later (but still in annotations) just remove the extra widgets. We might even be able to do the cleanup in the same array expansion ruleset.

But the addition of the subscript may be the better way anyway.

#10 - 12/09/2019 06:14 PM - Roger Borrello

Greg Shah wrote:

> I'm not suggesting doing it in core conversion. The alternative idea was to add next-prompt to the list of normal expansions and then later (but still in annotations) just remove the extra widgets. We might even be able to do the cleanup in the same array expansion ruleset.
>
> But the addition of the subscript may be the better way anyway.

So would this be handled similarly to (but adding parent.type == prog.kw_next_pmt condition?):

```
            <!-- set, update, prompt-for, disable, enable and assign all
                 have direct attachment (no intermediate nodes) -->
            <rule>
               (parent.type == prog.kw_set       or
                parent.type == prog.kw_update    or
                parent.type == prog.kw_prmt_for  or
                parent.type == prog.kw_disable   or
                parent.type == prog.kw_enable    or
                parent.type == prog.kw_assign) and
               parent.parent.type == prog.statement

               <action>attachIdx = copy.indexPos</action>
               <action>parentRef = copy.parent</action>
            </rule>
```

#11 - 12/10/2019 10:02 AM - Greg Shah

Correct, so long as the child of the kw_next_pmt is the field/var itself. For example, for anything with a form_item, it would be a different place to match.

#12 - 12/10/2019 11:28 AM - Roger Borrello

Greg Shah wrote:

> Correct, so long as the child of the kw_next_pmt is the field/var itself. For example, for anything with a form_item, it would be a different place to match.

I like the idea you brought up with crating a new template in ./rules/annotations/progress_templates.tpl but I am not familiar with the process. An accurate example of the current AST would look like this:

```
<ast col="0" id="0" line="0" text="statement" type="STATEMENT">
<ast col="7" id="0" line="22" text="next-prompt" type="KW_NEXT_PMT">
  <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
  <annotation datatype="java.lang.Long" key="scope-id" value="0"/>
  <annotation datatype="java.lang.Long" key="frame-id" value="0"/>
  <annotation datatype="java.lang.Long" key="peerid" value="0"/>
  <ast col="19" id="0" line="22" text="schedule" type="FIELD_CHAR">
    <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
    <annotation datatype="java.lang.String" key="schemaname" value="p2j_test.person.schedule"/>
    <annotation datatype="java.lang.String" key="bufname" value="p2j_test.person"/>
    <annotation datatype="java.lang.String" key="dbname" value="p2j_test"/>
    <annotation datatype="java.lang.Long" key="recordtype" value="12"/>
    <annotation datatype="java.lang.String" key="name" value="schedule"/>
    <annotation datatype="java.lang.Long" key="type" value="402"/>
    <annotation datatype="java.lang.String" key="format" value="&amp;quot;x(64)&amp;quot;"/>
    <annotation datatype="java.lang.Long" key="extent" value="5"/>
    <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
    <annotation datatype="java.lang.Boolean" key="is_meta" value="false"/>
    <annotation datatype="java.lang.String" key="fieldname" value="schedule"/>
    <annotation datatype="java.lang.String" key="bufrefkey" value="p2j_test.person,p2j_test.person,-1"/>
    <annotation datatype="java.lang.Long" key="bufreftype" value="18"/>
    <annotation datatype="java.lang.String" key="uniquename" value="p2j_test.person_p2j_test.person"/>
    <annotation datatype="java.lang.String" key="methodtxt" value="getSchedule"/>
    <annotation datatype="java.lang.String" key="getter" value="getScheduleArray0"/>
    <annotation datatype="java.lang.String" key="setter" value="setScheduleArray0"/>
    <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
    <annotation datatype="java.lang.String" key="javaname" value="scheduleArray0"/>
    <annotation datatype="java.lang.String" key="accessor" value="widgetScheduleArray0"/>
    <annotation datatype="java.lang.Boolean" key="next2lastref" value="true"/>
    <annotation datatype="java.lang.Long" key="frame-id" value="0"/>
    <annotation datatype="java.lang.Long" key="peerid" value="0"/>
    <ast col="27" hidden="true" id="0" line="22" text="[" type="LBRACKET">
      <ast col="0" id="0" line="0" text="expression" type="EXPRESSION">
        <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
        <ast col="28" id="0" line="22" text="1" type="NUM_LITERAL">
          <annotation datatype="java.lang.Boolean" key="is-literal" value="true"/>
          <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
          <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
        </ast>
      </ast>
    </ast>
  </ast>
</ast>
</ast>
```

Obviously the id="0" is easy. Do the col="" and line="" get removed, and the introduction of the template populate them in the AST? More questions, but in general, is there already a task documenting "templatizing" an AST node?

#13 - 12/10/2019 11:40 AM - Greg Shah

There is no documentation/task for "templatizing".

I think the annotations can all be left out. The line and column can be removed. You should ensure that you have text (empty string is OK) and type set properly. As you note, id should be set to 0. That is it. In this case, you really don't need the peernode annotation either.

#14 - 12/10/2019 11:58 AM - Roger Borrello

Greg Shah wrote:

> There is no documentation/task for "templatizing".
>
> I think the annotations can all be left out. The line and column can be removed. You should ensure that you have text (empty string is OK) and type set properly. As you note, id should be set to 0. That is it. In this case, you really don't need the peernode annotation either.

So something more like this...

```
<ast-root name="implied_extent_first_element" terse="true" ast_class="com.goldencode.p2j.uast.ProgressAst">
   <ast id="0" text="statement" type="STATEMENT">
      <ast id="0" text="next-prompt" type="KW_NEXT_PMT">
        <ast id="0" text="schedule" type="FIELD_CHAR">
          <ast hidden="true" id="0" text="[" type="LBRACKET">
            <ast id="0" text="expression" type="EXPRESSION">
              <ast id="0" text="1" type="NUM_LITERAL">
              </ast>
            </ast>
          </ast>
        </ast>
      </ast>
   </ast>
</ast-root>
```

And are we really looking at just the LBRACKET/EXPRESSION/NUM_LITERAL portion be grafted after any STATEMENT/KW_NEXT_PMT/FIELD_ that doesn't already have child LBRACKET?

#15 - 12/10/2019 12:02 PM - Constantin Asofiei

I think you need to graft only the LBRACKET portion. As a side note, the fix should be generic for any extent - be it a variable or field. So please create a test which has an extent var instead of field, to make sure that is fixed, too.

#16 - 12/10/2019 12:13 PM - Greg Shah

This:

```
<ast id="0" text="1" type="NUM_LITERAL">
</ast>
```

should be written like this:

<ast id="0" text="1" type="NUM_LITERAL" />

when there are no annotations.

#17 - 12/10/2019 12:14 PM - Greg Shah

Constantin Asofiei wrote:

> I think you need to graft only the LBRACKET portion. As a side note, the fix should be generic for any extent - be it a variable or field. So please create a test which has an extent var instead of field, to make sure that is fixed, too.

Yes, this is important. There may be a frame phrase and the original code would have to be removed if you try to add a new KW_NEXT_PMT etc... Keep the solution simple.

#18 - 12/10/2019 12:58 PM - Roger Borrello

Greg Shah wrote:

> Constantin Asofiei wrote:
>
> > I think you need to graft only the LBRACKET portion. As a side note, the fix should be generic for any extent - be it a variable or field. So please create a test which has an extent var instead of field, to make sure that is fixed, too.
>
> Yes, this is important. There may be a frame phrase and the original code would have to be removed if you try to add a new KW_NEXT_PMT etc... Keep the solution simple.

So this is the cut-down version:

```
<ast-root name="implied_extent_first_element" terse="true" ast_class="com.goldencode.p2j.uast.ProgressAst">
  <ast hidden="true" id="0" text="[" type="LBRACKET">
    <ast id="0" text="expression" type="EXPRESSION">
       <ast id="0" text="1" type="NUM_LITERAL" />
    </ast>
  </ast>
</ast-root>
```

Is the implied_extent_first_element name accurate? As far as where to place the graftAt() method, annotations/array_expansion.rules seems to be the obvious file. In fact, I can envision a function similar to duplicateArrayNode. Would the correct rule be within the evalLib("fields")?

#19 - 12/10/2019 01:01 PM - Greg Shah

Please remove the hidden="true". Otherwise it won't work.

> Is the implied_extent_first_element name accurate?

I think it is more generic than that. Call it first_element_subscript.

> As far as where to place the graftAt() method, annotations/array_expansion.rules seems to be the obvious file. In fact, I can envision a function similar to duplicateArrayNode.

Yes.

> Would the correct rule be within the evalLib("fields")?

That would handle fields but not variables. As Constantin notes you must handle extent variables too.

#20 - 12/10/2019 03:42 PM - Roger Borrello

While building up a testcase to use variables instead of fields, I noticed this situation. It may be a red herring, but perhaps it could lead to a simpler solution.

```
10: do with frame tt:
11:   update
12:     tt-desc[1]
13:     tt-desc[2]
14:     mynum[1]
15:  editing:
16:    readkey.
17:    if input tt-desc[1] = "" then
18:    do:
19:      message "Blank".
20:      next-prompt tt-desc.
21:      next.
22:    end.
```

On line 20, I don't have any subscript. This results in AST generation of:

```
273:            <ast col="0" id="768799146211" line="0" text="" type="FRAME_ELEMENT">
274:              <annotation datatype="java.lang.Long" key="peerid" value="773094113380"/>
275:              <ast col="5" id="768799146212" line="12" text="tt-desc" type="VAR_CHAR">
276:                <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
277:                <annotation datatype="java.lang.Long" key="extent" value="2"/>
278:                <annotation datatype="java.lang.Long" key="refid" value="768799146003"/>
279:                <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
280:                <annotation datatype="java.lang.String" key="getter" value="getTtDescArray0"/>
281:                <annotation datatype="java.lang.String" key="setter" value="setTtDescArray0"/>
282:                <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
283:                <annotation datatype="java.lang.String" key="javaname" value="ttDescArray0"/>
284:                <annotation datatype="java.lang.String" key="accessor" value="widgetTtDescArray0"/>
285:                <annotation datatype="java.lang.Boolean" key="firstref" value="true"/>
286:                <annotation datatype="java.lang.Boolean" key="lastref" value="true"/>
287:                <annotation datatype="java.lang.Long" key="peerid" value="773094113382"/>
288:                <ast col="12" id="768799146213" line="12" text="[" type="LBRACKET">
289:                  <annotation datatype="java.lang.Long" key="peerid" value="773094113381"/>
290:                  <ast col="0" id="768799146214" line="0" text="expression" type="EXPRESSION">
291:                    <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
292:                    <annotation datatype="java.lang.Long" key="peerid" value="773094113383"/>
293:                    <ast col="13" id="768799146215" line="12" text="1" type="NUM_LITERAL">
294:                      <annotation datatype="java.lang.Boolean" key="is-literal" value="true"/>
295:                      <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
296:                      <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
297:                      <annotation datatype="java.lang.Long" key="peerid" value="773094113384"/>
298:                    </ast>
299:                  </ast>
300:                </ast>
301:              </ast>
```

When line 20 is updated to have [1], There is key="firstref" left out on line **285**, resulting in:

```
273:            <ast col="0" id="768799146211" line="0" text="" type="FRAME_ELEMENT">
274:              <annotation datatype="java.lang.Long" key="peerid" value="773094113380"/>
275:              <ast col="5" id="768799146212" line="12" text="tt-desc" type="VAR_CHAR">
276:                <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
277:                <annotation datatype="java.lang.Long" key="extent" value="2"/>
278:                <annotation datatype="java.lang.Long" key="refid" value="768799146003"/>
279:                <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
280:                <annotation datatype="java.lang.String" key="getter" value="getTtDescArray0"/>
281:                <annotation datatype="java.lang.String" key="setter" value="setTtDescArray0"/>
282:                <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
283:                <annotation datatype="java.lang.String" key="javaname" value="ttDescArray0"/>
284:                <annotation datatype="java.lang.String" key="accessor" value="widgetTtDescArray0"/>
285:                <annotation datatype="java.lang.Boolean" key="lastref" value="true"/>
286:                <annotation datatype="java.lang.Long" key="peerid" value="773094113382"/>
287:                <ast col="12" id="768799146213" line="12" text="[" type="LBRACKET">
288:                  <annotation datatype="java.lang.Long" key="peerid" value="773094113381"/>
```

```
289:                  <ast col="0" id="768799146214" line="0" text="expression" type="EXPRESSION">
290:                    <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
291:                    <annotation datatype="java.lang.Long" key="peerid" value="773094113383"/>
292:                    <ast col="13" id="768799146215" line="12" text="1" type="NUM_LITERAL">
293:                      <annotation datatype="java.lang.Boolean" key="is-literal" value="true"/>
294:                      <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
295:                      <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
296:                      <annotation datatype="java.lang.Long" key="peerid" value="773094113384"/>
297:                    </ast>
298:                  </ast>
299:                </ast>
300:              </ast>
```

This section is from line 12 of the source in the update, not even near line 20. The firstref annotation is added in the frame_scoping.rules when a widget first appears in a frame.

Does that sound like something to pursue?

#21 - 12/10/2019 04:28 PM - Roger Borrello

New Testcases
I have updated the testcase to not use the p2j_test.df, instead using TEMP-TABLE. I also added a new one to utilize variables, not fields.

```
uast/input/extent_nosubscript_fld_next-prompt.p
uast/input/extent_nosubscript_var_next-prompt.p
```

#22 - 12/10/2019 05:15 PM - Greg Shah

> On line 20, I don't have any subscript. This results in AST generation of:

Are you saying that the input/extent_nosubscript_var_next-prompt.p does not fail in trunk while input/extent_nosubscript_fld_next-prompt.p does fail?

> This section is from line 12 of the source in the update, not even near line 20. The firstref annotation is added in the frame_scoping.rules when a widget first appears in a frame.
>
> Does that sound like something to pursue?

No, it is a red herring.

#23 - 12/10/2019 05:24 PM - Roger Borrello

Greg Shah wrote:

> On line 20, I don't have any subscript. This results in AST generation of:

> Are you saying that the input/extent_nosubscript_var_next-prompt.p does not fail in trunk while input/extent_nosubscript_fld_next-prompt.p does fail?

No. I had created a version of the 4gl with [1] on the field/variable in order to create a "model" AST, to see what might be generated in the explicit instance. I was noting there were differences I didn't expect.

> This section is from line 12 of the source in the update, not even near line 20. The firstref annotation is added in the frame_scoping.rules when a widget first appears in a frame.
>
> Does that sound like something to pursue?

> No, it is a red herring.

Good. I have a potential fix to review.

#24 - 12/10/2019 06:10 PM - Roger Borrello

Updated

Below are my updates and findings

rules/annotations/progress_templates.tpl

I have added this ast-root:

```
  <ast-root name="first_element_subscript" terse="true" ast_class="com.goldencode.p2j.uast.ProgressAst">
    <ast id="0" text="[" type="LBRACKET">
```

```
      <ast id="0" text="expression" type="EXPRESSION">
         <ast id="0" text="1" type="NUM_LITERAL" />
      </ast>
    </ast>
  </ast-root>
```

rules/annotations/array_expansion.rules

Added this worker class and variable:

```
  <worker class="com.goldencode.p2j.pattern.TemplateWorker"     namespace="tpl" />
  <variable name="tref"       type="com.goldencode.ast.Aast" />
```

Within the walk-rules there is the initial rule for determining if you are a variable or field:

```
     <!-- all variables or fields -->
     <rule>
        ((evalLib("variables") or evalLib("widget_references")) and
         isNote("refid")) or
        evalLib("fields")
```

Inside that rule, there was a sub-rule to determine if this was an implicit full array reference OR a range array reference:

```
        <!-- is this an implicit full array reference OR a range array reference? -->
        <rule>
           extent > 0                     and
           (!descendant(prog.lbracket, 1) or
            descendant(prog.kw_for, 2))
```

It is within this rule that I have nested a new rule, which validates we are down from "STATEMENT/KW_NEXT_PMT". I verified this with printfln statements, and various checks for prog.kw_next_pmt and prog.statement, but ultimately settled on this rule, since we have just determined that we have extent > 0 and we either don't have prog.lbracket as a child, or we do have prog.kw_for as a grand-child. *Note that I am still trying to find where the descendant() method is defined, so I'm taking a guess.*

```
           <rule>upPath("STATEMENT/KW_NEXT_PMT")
              <action>tref = tpl.graft("first_element_subscript", null, copy)</action>
           </rule>
```

AST Results

Using these updates, I ran against my 2 testcases (variables and fields).

Variable Case

Here is a snippet of the AST for the variable case:

```
<ast col="0" id="785979015318" line="0" text="statement" type="STATEMENT">
  <ast col="7" id="785979015319" line="20" text="next-prompt" type="KW_NEXT_PMT">
    <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
    <annotation datatype="java.lang.Long" key="scope-id" value="785979015396"/>
    <annotation datatype="java.lang.Long" key="frame-id" value="785979015399"/>
    <annotation datatype="java.lang.Long" key="peerid" value="790273982630"/>
    <ast col="19" id="785979015321" line="20" text="tt-desc" type="VAR_CHAR">
      <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
      <annotation datatype="java.lang.Long" key="extent" value="2"/>
      <annotation datatype="java.lang.Long" key="refid" value="785979015187"/>
      <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
      <annotation datatype="java.lang.String" key="getter" value="getTtDescArray0"/>
      <annotation datatype="java.lang.String" key="setter" value="setTtDescArray0"/>
```

```
      <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
      <annotation datatype="java.lang.String" key="javaname" value="ttDescArray0"/>
      <annotation datatype="java.lang.String" key="accessor" value="widgetTtDescArray0"/>
      <annotation datatype="java.lang.Boolean" key="firstref" value="true"/>
      <annotation datatype="java.lang.Long" key="frame-id" value="785979015399"/>
      <annotation datatype="java.lang.Long" key="peerid" value="790273982632"/>
      <ast col="0" hidden="true" id="785979015390" line="0" text="[" type="LBRACKET">
        <ast col="0" id="785979015391" line="0" text="expression" type="EXPRESSION">
          <ast col="0" id="785979015392" line="0" text="1" type="NUM_LITERAL">
            <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
          </ast>
        </ast>
      </ast>
    </ast>
  </ast>
```

Field Case

Below is the output from the field testcase:

```
  <ast col="0" id="803158884617" line="0" text="statement" type="STATEMENT">
    <ast col="7" id="803158884618" line="35" text="next-prompt" type="KW_NEXT_PMT">
      <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
      <annotation datatype="java.lang.Long" key="scope-id" value="803158884709"/>
      <annotation datatype="java.lang.Long" key="frame-id" value="803158884710"/>
      <annotation datatype="java.lang.Long" key="peerid" value="820338753761"/>
      <ast col="19" id="803158884620" line="35" text="tt-desc" type="FIELD_CHAR">
        <annotation datatype="java.lang.Long" key="oldtype" value="2782"/>
        <annotation datatype="java.lang.String" key="schemaname" value="tt.tt-desc"/>
        <annotation datatype="java.lang.String" key="bufname" value="tt"/>
        <annotation datatype="java.lang.String" key="dbname" value=""/>
        <annotation datatype="java.lang.Long" key="recordtype" value="14"/>
        <annotation datatype="java.lang.String" key="name" value="tt-desc"/>
        <annotation datatype="java.lang.Long" key="type" value="402"/>
        <annotation datatype="java.lang.Long" key="extent" value="2"/>
        <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
        <annotation datatype="java.lang.Boolean" key="is_meta" value="false"/>
        <annotation datatype="java.lang.String" key="fieldname" value="ttDesc"/>
        <annotation datatype="java.lang.String" key="bufrefkey" value="tt,tt,-1"/>
        <annotation datatype="java.lang.Long" key="bufreftype" value="18"/>
        <annotation datatype="java.lang.String" key="uniquename" value="tt_tt"/>
        <annotation datatype="java.lang.String" key="methodtxt" value="getTtDesc"/>
        <annotation datatype="java.lang.String" key="getter" value="getTtDescArray0"/>
        <annotation datatype="java.lang.String" key="setter" value="setTtDescArray0"/>
        <annotation datatype="java.lang.String" key="widgettype" value="FillInWidget"/>
        <annotation datatype="java.lang.String" key="javaname" value="ttDescArray0"/>
        <annotation datatype="java.lang.String" key="accessor" value="widgetTtDescArray0"/>
        <annotation datatype="java.lang.Long" key="frame-id" value="803158884710"/>
        <annotation datatype="java.lang.Long" key="peerid" value="820338753763"/>
        <ast col="0" hidden="true" id="803158884695" line="0" text="[" type="LBRACKET">
          <ast col="0" id="803158884696" line="0" text="expression" type="EXPRESSION">
            <ast col="0" id="803158884697" line="0" text="1" type="NUM_LITERAL">
              <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
            </ast>
          </ast>
        </ast>
      </ast>
    </ast>
  </ast>
```

Should any of the information within the template have been filled in more, like the line numbers? The overall structure looks good.

Java Results

Here are the pertinent files.

Variable Case ExtentNosubscriptVarNextPrompt.java

```
package com.goldencode.testcases.input;

import com.goldencode.p2j.util.*;
import com.goldencode.testcases.ui.input.*;
import com.goldencode.p2j.ui.*;

import static com.goldencode.p2j.util.BlockManager.*;
import static com.goldencode.p2j.util.CompareOps.*;
import static com.goldencode.p2j.util.ArrayAssigner.*;
import static com.goldencode.p2j.ui.LogicalTerminal.*;

/**
 * Business logic (converted to Java from the 4GL source code
 * in input/extent_nosubscript_var_next-prompt.p).
 */
public class ExtentNosubscriptVarNextPrompt
{
   ExtentNosubscriptVarNextPromptTt ttFrame = GenericFrame.createFrame(ExtentNosubscriptVarNextPromptTt.class,
 "tt");

   ExtentNosubscriptVarNextPromptMyfr myfrFrame = GenericFrame.createFrame(ExtentNosubscriptVarNextPromptMyfr.
class, "myfr");

   /**
    * External procedure (converted to Java from the 4GL source code
    * in input/extent_nosubscript_var_next-prompt.p).
    */
   public void execute()
   {
      integer[] mynum = UndoableFactory.integerExtent(2);
      character[] ttDesc = UndoableFactory.characterExtent(2);

      externalProcedure(ExtentNosubscriptVarNextPrompt.this, new Block((Body) () ->
      {
         myfrFrame.openScope();

         doBlock("blockLabel0", new Block((Init) () ->
         {
            ttFrame.openScope();
         },
         (Body) () ->
         {
            FrameElement[] elementList0 = new FrameElement[]
            {
               new Element(subscript(ttDesc, 1), ttFrame.widgetTtDescArray0()),
               new Element(subscript(ttDesc, 2), ttFrame.widgetTtDescArray1()),
               new Element(subscript(mynum, 1), ttFrame.widgetMynumArray0())
            };

            updateEditing(ttFrame, elementList0, "editingLabel0", new Block((Body) () ->
            {
               KeyReader.readKey();

               if (_isEqual(myfrFrame.getTtDescArray0(), ""))
               {
                  message("Blank");
                  ttFrame.nextPrompt(ttFrame.widgetTtDescArray0());
                  next("editingLabel0");
               }

               if (_isNotEqual(myfrFrame.getTtDescArray0(), ""))
               {
                  message("Not blank");
                  ttFrame.nextPrompt(ttFrame.widgetTtDescArray0());
                  next("editingLabel0");
               }
            }));
         }));

         message("Done");
      }));
   }
}
```

Variable Case UI ExtentNosubscriptVarNextPromptTt.java

```
package com.goldencode.testcases.ui.input;

import com.goldencode.p2j.util.*;
import com.goldencode.p2j.ui.*;

public interface ExtentNosubscriptVarNextPromptTt
extends CommonFrame
{
   public static final Class configClass = ExtentNosubscriptVarNextPromptTtDef.class;

   public character getTtDescArray0();

   public character getTtDescArray(NumberType parm);

   public character getTtDescArray(double parm);

   public void setTtDescArray0(character parm);

   public void setTtDescArray0(String parm);

   public void setTtDescArray0(BaseDataType parm);

   public FillInWidget widgetTtDescArray0();

   public FillInWidget widgetTtDescArray(NumberType parm);

   public FillInWidget widgetTtDescArray(double parm);

   public character getTtDescArray1();

   public void setTtDescArray1(character parm);

   public void setTtDescArray1(String parm);

   public void setTtDescArray1(BaseDataType parm);

   public FillInWidget widgetTtDescArray1();

   public integer getMynumArray0();

   public integer getMynumArray(NumberType parm);

   public integer getMynumArray(double parm);

   public void setMynumArray0(integer parm);

   public void setMynumArray0(int parm);

   public void setMynumArray0(BaseDataType parm);

   public FillInWidget widgetMynumArray0();

   public FillInWidget widgetMynumArray(NumberType parm);

   public FillInWidget widgetMynumArray(double parm);

   public static class ExtentNosubscriptVarNextPromptTtDef
   extends WidgetList
   {
      FillInWidget ttDescArray0 = new FillInWidget();

      FillInWidget ttDescArray1 = new FillInWidget();

      FillInWidget mynumArray0 = new FillInWidget();

      public void setup(CommonFrame frame)
      {
         frame.setDown(1);
         ttDescArray0.setIndex(1);
         ttDescArray0.setDataType("character");
         ttDescArray1.setIndex(2);
```

```
         ttDescArray1.setDataType("character");
         mynumArray0.setIndex(1);
         mynumArray0.setDataType("integer");
         ttDescArray0.setLabel("tt-desc[1]");
         ttDescArray1.setLabel("tt-desc[2]");
         mynumArray0.setLabel("mynum[1]");
      }

      {
         addWidget("ttDescArray0", "tt-desc", ttDescArray0);
         addWidget("ttDescArray1", "tt-desc", ttDescArray1);
         addWidget("mynumArray0", "mynum", mynumArray0);
      }
   }
}
```

Field Case ExtentNosubscriptFldNextPrompt.java

```
package com.goldencode.testcases.input;

import com.goldencode.p2j.util.*;
import com.goldencode.testcases.ui.input.*;
import com.goldencode.p2j.ui.*;
import com.goldencode.p2j.persist.*;
import com.goldencode.testcases.dmo._temp.*;
import com.goldencode.p2j.persist.lock.*;

import static com.goldencode.p2j.util.BlockManager.*;
import static com.goldencode.p2j.util.CompareOps.*;
import static com.goldencode.p2j.util.ArrayAssigner.*;
import static com.goldencode.p2j.ui.LogicalTerminal.*;
import static com.goldencode.p2j.util.ErrorManager.*;

/**
 * Business logic (converted to Java from the 4GL source code
 * in input/extent_nosubscript_fld_next-prompt.p).
 */
public class ExtentNosubscriptFldNextPrompt
{
   Tt_1_1.Buf tt = SharedVariableManager.addTempTable("tt", Tt_1_1.Buf.class, "tt", "tt");

   ExtentNosubscriptFldNextPromptTt ttFrame = GenericFrame.createFrame(ExtentNosubscriptFldNextPromptTt.class,
 "tt");

   /**
    * External procedure (converted to Java from the 4GL source code
    * in input/extent_nosubscript_fld_next-prompt.p).
    */
   public void execute()
   {
      integer[] mynum = UndoableFactory.integerExtent(2);

      externalProcedure(ExtentNosubscriptFldNextPrompt.this, new Block((Body) () ->
      {
         ttFrame.openScope();
         RecordBuffer.openScope(tt);
         tt.create();
         tt.setTtDesc(0, new character("Roger"));
         tt.setTtDesc(1, new character("Borrello"));
         assignSingle(mynum, 1, 1);
         tt.create();
         tt.setTtDesc(0, new character("Greg"));
         tt.setTtDesc(1, new character("Shah"));
         assignSingle(mynum, 1, 2);
         silent(() -> new FindQuery(tt, "?", null, "tt.id asc", new Object[]
         {
            isGreaterThan(subscript(mynum, 1), 1)
         }).first());

         doBlock("blockLabel0", new Block((Body) () ->
         {
            FrameElement[] elementList0 = new FrameElement[]
```

```
            {
               new Element(new FieldReference(tt, "ttDesc", 0), ttFrame.widgetTtDescArray0()),
               new Element(new FieldReference(tt, "ttDesc", 1), ttFrame.widgetTtDescArray1()),
               new Element(subscript(mynum, 1), ttFrame.widgetMynumArray0())
            };

            updateEditing(ttFrame, elementList0, "editingLabel0", new Block((Body) () ->
            {
               KeyReader.readKey();

               if (_isEqual(ttFrame.getTtDescArray0(), ""))
               {
                  message("Blank");
                  ttFrame.nextPrompt(ttFrame.widgetTtDescArray0());
                  next("editingLabel0");
               }

               if (_isNotEqual(ttFrame.getTtDescArray0(), ""))
               {
                  message("Not blank");
                  ttFrame.nextPrompt(ttFrame.widgetTtDescArray0());
                  next("editingLabel0");
               }
            }));
         }));

         message("Done");
      }));
   }
}
```

Field Case UI ExtentNosubscriptFldNextPromptTt.java

```
package com.goldencode.testcases.ui.input;

import com.goldencode.p2j.util.*;
import com.goldencode.p2j.ui.*;

public interface ExtentNosubscriptFldNextPromptTt
extends CommonFrame
{
   public static final Class configClass = ExtentNosubscriptFldNextPromptTtDef.class;

   public character getTtDescArray0();

   public character getTtDescArray(NumberType parm);

   public character getTtDescArray(double parm);

   public void setTtDescArray0(character parm);

   public void setTtDescArray0(String parm);

   public void setTtDescArray0(BaseDataType parm);

   public FillInWidget widgetTtDescArray0();

   public FillInWidget widgetTtDescArray(NumberType parm);

   public FillInWidget widgetTtDescArray(double parm);

   public character getTtDescArray1();

   public void setTtDescArray1(character parm);

   public void setTtDescArray1(String parm);

   public void setTtDescArray1(BaseDataType parm);

   public FillInWidget widgetTtDescArray1();

   public integer getMynumArray0();
```

```
   public integer getMynumArray(NumberType parm);

   public integer getMynumArray(double parm);

   public void setMynumArray0(integer parm);

   public void setMynumArray0(int parm);

   public void setMynumArray0(BaseDataType parm);

   public FillInWidget widgetMynumArray0();

   public FillInWidget widgetMynumArray(NumberType parm);

   public FillInWidget widgetMynumArray(double parm);

   public static class ExtentNosubscriptFldNextPromptTtDef
   extends WidgetList
   {
      FillInWidget ttDescArray0 = new FillInWidget();

      FillInWidget ttDescArray1 = new FillInWidget();

      FillInWidget mynumArray0 = new FillInWidget();

      public void setup(CommonFrame frame)
      {
         frame.setDown(1);
         ttDescArray0.setIndex(1);
         ttDescArray0.setTable("tt");
         ttDescArray0.setDataType("character");
         ttDescArray1.setIndex(2);
         ttDescArray1.setTable("tt");
         ttDescArray1.setDataType("character");
         mynumArray0.setIndex(1);
         mynumArray0.setDataType("integer");
         ttDescArray0.setTable("tt");
         ttDescArray1.setTable("tt");
         ttDescArray0.setTable("tt");
         ttDescArray0.setTable("tt");
         ttDescArray0.setTable("tt");
         ttDescArray0.setTable("tt");
         ttDescArray0.setLabel("tt-desc[1]");
         ttDescArray1.setLabel("tt-desc[2]");
         mynumArray0.setLabel("mynum[1]");
      }

      {
         addWidget("ttDescArray0", "tt-desc", ttDescArray0);
         addWidget("ttDescArray1", "tt-desc", ttDescArray1);
         addWidget("mynumArray0", "mynum", mynumArray0);
      }
   }
}
```

#25 - 12/10/2019 06:59 PM - Greg Shah

Code Review Task Branch 4207a Revisions 11353 - 11354

The changes are good.

The only problem is the range extent case. Please test this syntax (it would like something like NEXT-PROMPT MY-ARRAY-VAR[2 FOR 3].). Your changes won't work in this case, BUT it is possible that the 4GL doesn't allow this usage anyway. If disallowed, then you can just add a comment to your changes that the range syntax cannot be used in NEXT-PROMPT and you are done. If allowed, then your change will need to be different for that case.

#26 - 12/11/2019 09:02 AM - Roger Borrello

Greg Shah wrote:

> Code Review Task Branch 4207a Revisions 11353 - 11354
>
> The changes are good.
>
> The only problem is the range extent case. Please test this syntax (it would like something like NEXT-PROMPT MY-ARRAY-VAR[2 FOR 3].). Your changes won't work in this case, BUT it is possible that the 4GL doesn't allow this usage anyway. If disallowed, then you can just add a comment to your changes that the range syntax cannot be used in NEXT-PROMPT and you are done. If allowed, then your change will need to be different for that case.

Thanks for the feedback. I'll look into it.

#27 - 12/11/2019 09:14 AM - Roger Borrello

Greg Shah wrote:

> Code Review Task Branch 4207a Revisions 11353 - 11354
>
> The changes are good.
>
> The only problem is the range extent case. Please test this syntax (it would like something like NEXT-PROMPT MY-ARRAY-VAR[2 FOR 3].). Your changes won't work in this case, BUT it is possible that the 4GL doesn't allow this usage anyway. If disallowed, then you can just add a comment to your changes that the range syntax cannot be used in NEXT-PROMPT and you are done. If allowed, then your change will need to be different for that case.

If the rule condition were changed to:

```
            <rule>upPath("STATEMENT/KW_NEXT_PMT") and !descendant(prog.kw_for, 2)
               <action>tref = tpl.graft("first_element_subscript", null, copy)</action>
            </rule>
```

There would be protection against the case where there is a range. If there is a range, there's no reason to graft our [1] since there is already a subscript.

#28 - 12/11/2019 09:16 AM - Greg Shah

Yes, that is the safe thing to do. Please also report here if the 4GL allows the range case at all.

#29 - 12/11/2019 09:33 AM - Roger Borrello

Greg Shah wrote:

> Yes, that is the safe thing to do. Please also report here if the 4GL allows the range case at all.

Syntactically, 4GL allows this:

```
    if input tt-desc[1] = "" then
    do:
      message "Blank".
      next-prompt tt-desc[1 FOR 2].
      next.
    end.
```

It does run, and places the cursor in the first field.

FWD gets compilation errors:

```
    [javac] /home/rfb/projects/testcases/uast/uast/buildarea/src/com/goldencode/testcases/abl/ExtentNosubscrip
tVarNextPrompt.java:55: error: illegal start of expression
    [javac]                   ttFrame.nextPrompt(ttFrame.widgetTtDescArray0([0]));
    [javac]                                                                      ^
    [javac] /home/rfb/projects/testcases/uast/uast/buildarea/src/com/goldencode/testcases/abl/ExtentNosubscrip
tVarNextPrompt.java:55: error: ')' expected
    [javac]                   ttFrame.nextPrompt(ttFrame.widgetTtDescArray0([0]));
    [javac]                                                                       ^
    [javac] /home/rfb/projects/testcases/uast/uast/buildarea/src/com/goldencode/testcases/abl/ExtentNosubscrip
tVarNextPrompt.java:55: error: ';' expected
    [javac]                   ttFrame.nextPrompt(ttFrame.widgetTtDescArray0([0]));
```

#30 - 12/11/2019 09:37 AM - Greg Shah

OK, just add a TODO comment in to the array expansion rules to explain that the range case works in the 4GL (acts like it is just the first widget) but needs additional changes to work in FWD.

#31 - 12/11/2019 10:01 AM - Roger Borrello

Greg Shah wrote:

> OK, just add a TODO comment in to the array expansion rules to explain that the range case works in the 4GL (acts like it is just the first widget) but needs additional changes to work in FWD.

I also noticed we don't seem to support including a variable in the range. 4GL documentation states:

> ... In a range, you can use a variable for the first element, but the second element must be a constant...

When I made a quick update (that works in 4GL):

```
if input tt-desc[1] = "" then
do:
  message "Blank".
  next-prompt tt-desc[mynum[1] for 2].
  next.
end.
```

We get Core Code Conversion failure:

```
     [java] Not supported statement in dynamic array expansion: tt-desc [VAR_CHAR]:12884902041 @20:19
     [java]    [ [LBRACKET]:12884902042 @20:26
     [java]       expression [EXPRESSION]:12884902043 @0:0
     [java]          mynum [VAR_INT]:12884902044 @20:27
     [java]             [ [LBRACKET]:12884902045 @20:32
     [java]                1 [NUM_LITERAL]:12884902046 @20:33
     [java]       for [KW_FOR]:12884902049 @20:36
     [java]       2 [NUM_LITERAL]:12884902051 @20:40
     [java]    [ [LBRACKET]:12884902121 @0:0
     [java]       expression [EXPRESSION]:12884902122 @0:0
     [java]          1 [NUM_LITERAL]:12884902123 @0:0
```

I'll add that info in the TODO.

#32 - 12/11/2019 10:10 AM - Roger Borrello

- % Done changed from 0 to 100

- Status changed from New to WIP

- Assignee set to Roger Borrello

#33 - 12/31/2019 11:58 AM - Greg Shah

- Status changed from WIP to Test

#34 - 03/03/2020 02:49 PM - Roger Borrello

Task branch 4207a was merged to trunk as revision 11344.

#35 - 03/04/2020 10:29 AM - Greg Shah

- Status changed from Test to Closed