Base Language - Bug #4563

assigning a dynamic extent var to a OO method call as rvalue

03/03/2020 08:14 AM - Constantin Asofiei

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

Description

Related issues:

Related to Base Language - Feature #4373: finish core OO 4GL support	**New**

History

#1 - 03/03/2020 08:14 AM - Constantin Asofiei

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

#2 - 03/03/2020 08:22 AM - Constantin Asofiei

The testcases/uast/ext_arg_*.p tests explore how a parameter/variable (dynamic extent or not) used as an argument emits - this requires to be a two-dimensional array, as the previously ExtentExpr inner class is now emitted inline, as an anon class - this is required to avoid promoting any local vars/parameters (to a method/proc/function).

There are two issues found when working on this.

This test fails in FWD because it does not emit a real Java assignment - the lvalue b4 is missing in the converted code.

```
def var b4 as  char  extent.
def var js as progress.json.objectmodel.jsonobject.
assign b4 = js:getNames().
```

The other issue is related to local functions returning an extent object:

```
function funcx returns progress.lang.object extent 5.
end.
def var b4 as  progress.lang.object  extent.
assign b4 = funcx().
```

This does not convert properly - the generic cast emitted for assignMulti is using the 4GL class name, and not the Java name.

When working on this, we should explore how 4GL class vars/properties work when used in a similar approach as described by the ext_arg_ tests.

The previous fixes are in 4207a, rev 11423, 11424 11412, 11405. The files affected were:

```
  rules/annotations/output_parameters.rules
  rules/convert/variable_definitions.rules
```

I suspect the fix for the first issue will be in rules/convert/assignments.rules, line 283, where this comment exists:

```
                    <!-- only if the destination variable has an undeterminate extent its size
                         will be changed and needs to be assigned back. Otherwise both variables
                         MUST have the same EXTENT size. See error (14905)-->
                    <rule>#(long) def.getAnnotation("extent") == #(long) (-1)
                       <action>rvalueExp = copy.getChildAt(1)</action>
```

This will need to include OO method calls. The change will not be able to rely on refid, as it doesn't exist for builtin OO method calls.

#3 - 03/03/2020 01:50 PM - Constantin Asofiei

Static vars/properties (with dynamic extent) should be included.