Base Language - Feature #2048

BUFFER parameters without name in forward function definitions

02/25/2013 08:53 AM - Constantin Asofiei

Status:	Closed	**Start date:**	02/25/2013
Priority:	Normal	**Due date:**	
Assignee:	Costin Savin	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Conversion Support for Server Features		
billable:	No	**vendor_id:**	GCD

Description

History

#1 - 02/25/2013 08:56 AM - Constantin Asofiei

- Target version set to Milestone 4

Conversion fails in this case:

```
function func0 returns char (buffer for book) forward.
```

as forward function definitions allow the parameters to be defined without explicitly setting the name.

#2 - 02/25/2013 01:59 PM - Costin Savin

The conversion fails because of code in record_scoping_prep.rules which assumes the buffer annotation is in the copy.parent.prevSibling which in the case of
function func0 returns char (buffer for book) forward. is not true and the prevSibling node does not exist
A null check for copy.parent.prevSibling and getting the annotation from the current node in the case this is null solved the problem.
However the code is still not converted correctly or persistent:

Example:

forward.p

```
FUNCTION doubler RETURNS INTEGER (buffer for book) FORWARD.
FUNCTION doubler RETURNS INTEGER (buffer bb for book). end.

FUNCTION doubler2 RETURNS INTEGER (buffer cc for book) FORWARD.
FUNCTION doubler2 RETURNS INTEGER (buffer cc for book). end.
```

the first function doubler converts wrong to

```
   public integer doubler(final Book.Buf bb)
   {
      return function(integer.class, new Block()
      {
         public void init()
         {
            Winhand.this.bb = bb;
         }

         public void body()
         {
```

```
            RecordBuffer.openScope(bb, book);
         }
      });
   }
```

This happens because book is put as a javaname annotation together with bb which are then are used as parameter for emiting RecordBuffer.openScope statement

in the second case it converts correctly to:

```
   public integer doubler2(final Book.Buf ccBuf2)
   {
      return function(integer.class, new Block()
      {
         public void init()
         {
            Winhand.this.ccBuf2 = ccBuf2;
         }
         public void body()
         {
            RecordBuffer.openScope(ccBuf2, ccBuf2);
         }
      });
   }
```

but in this case the javaname annotations are emitted correctly to ccBuf2.
I'll think on how can this be fixed.

#3 - 02/25/2013 02:19 PM - Greg Shah

Actually, both results look wrong. doubler2() gets an openScope() with the same buffer name passed twice. That may be OK, but is definitely not what we want. The core remaining issue is that there should not ever be 2 buffers emitted in an openScope() for a single buffer parameter.

#4 - 02/25/2013 02:20 PM - Greg Shah

Keep in mind that the FORWARD declaration should not be changing buffer scoping at all. My guess is that somewhere we are not bypassing the addition of a new buffer when it is a FORWARD function.

#5 - 02/26/2013 04:44 AM - Costin Savin

- File cs_upd20130226a.zip added

Added proposed update

#6 - 02/26/2013 04:46 AM - Constantin Asofiei

The update looks OK, is going through conversion regression testing now.

#7 - 02/26/2013 05:44 AM - Constantin Asofiei

Update has passed conversion regression testing, please check in and distribute.

#8 - 02/26/2013 06:17 AM - Costin Savin

Commited to bazaar as revision number 10209.

#9 - 02/26/2013 08:21 AM - Greg Shah

- Status changed from New to Closed

#10 - 02/26/2013 06:00 PM - Eric Faulhaber

- % Done changed from 0 to 100

#11 - 11/16/2016 11:07 AM - Greg Shah

- Target version changed from Milestone 4 to Conversion Support for Server Features

Files

cs_upd20130226a.zip	41.2 KB	02/26/2013	Costin Savin