Conversion Tools - Bug #3516

Conversion issue for lambda generated code and library_calls testcases

03/14/2018 11:10 AM - Eugenie Lyzenko

Status:	New	**Start date:**	03/14/2018
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 03/14/2018 11:12 AM - Eugenie Lyzenko

The conversion of the library_calls tests using current lambda based approach now fails to compile with many errors like:

```
[ant:javac] Compiling 266 source files to /home/evl/timco_new/p2j/build/classes
[ant:javac] /home/evl/projects/p2j/src/com/goldencode/testcases/library_calls/input/InputDoubleParameterIntoIn
t64IsJunk.java:34: error: local variables referenced from a lambda expression must be final or effectively fin
al
[ant:javac]          dInput = ArrayAssigner.resize(dInput, sz);
[ant:javac]                                        ^
[ant:javac] /home/evl/projects/p2j/src/com/goldencode/testcases/library_calls/input/InputDoubleParameterIntoIn
t64IsJunk.java:34: error: local variables referenced from a lambda expression must be final or effectively fin
al
[ant:javac]          dInput = ArrayAssigner.resize(dInput, sz);
[ant:javac]          ^
[ant:javac] /home/evl/projects/p2j/src/com/goldencode/testcases/library_calls/input/InputDoubleParameterIntoIn
t64IsJunk.java:40: error: local variables referenced from a lambda expression must be final or effectively fin
al
[ant:javac]             assignSingle(dInput, idx, comps.getDouble(plus(multiply(minus(i, 1), 8), 1)));
[ant:javac]                          ^
[ant:javac] /home/evl/projects/p2j/src/com/goldencode/testcases/library_calls/input/InputDoubleParameterIntoIn
t64IsJunk.java:47: error: local variables referenced from a lambda expression must be final or effectively fin
al
[ant:javac]             ptr.setDouble(subscript(dInput, i), 1);
[ant:javac]                                     ^
[ant:javac] 4 errors
:ant-compile FAILED
```

The original code for this issue:

```
...
def var d-input as dec extent.
...
extent(d-input) = sz. <--------- Error
...
do i = sz to 1 by -1:
   idx = (sz - i) + 1.
   d-input[idx] = get-double(comps, ((i - 1) * 8) + 1). <---------- Error
end.
...
```

Converted code:

```
public class InputDoubleParameterIntoInt64IsJunk
{
   Stream logStrStream = new StreamWrapper("log-str");
...
   public void execute()
   {
...
      decimal[] dInput = UndoableFactory.decimalExtent();

      externalProcedure(InputDoubleParameterIntoInt64IsJunk.this, new Block((Body) () ->
      {
         TransactionManager.registerTopLevelFinalizable(logStrStream, true);
         ControlFlowOps.invokeWithMode("log-it", "I", new character("input_double_paramter_into_int64_is_junk
start"));
         ControlFlowOps.invokeWithMode("dvalues_size", "O", sz);
         ControlFlowOps.invoke("init_dvalues");
         ptr.setLength(8);
         comps.setLength(multiply(sz, 8));
         ControlFlowOps.invokeWithMode("copy_dvalues", "U", comps);
         dInput = ArrayAssigner.resize(dInput, sz); <--------- Error

         loopLabel0:
         for (i.assign(sz); _isGreaterThanOrEqual(i, 1); i.decrement())
         {
            idx.assign(plus(minus(sz, i), 1));
            assignSingle(dInput, idx, comps.getDouble(plus(multiply(minus(i, 1), 8), 1))); <--------- Error
         }
...
```

The variable dInput is out of scope from the code inside lambda. The workaround is to move dInput definition to the class member level:

```
public class InputDoubleParameterIntoInt64IsJunk
{
   Stream logStrStream = new StreamWrapper("log-str");
   decimal[] dInput = UndoableFactory.decimalExtent();
...
   public void execute()
   {
...
      externalProcedure(InputDoubleParameterIntoInt64IsJunk.this, new Block((Body) () ->
      {
         TransactionManager.registerTopLevelFinalizable(logStrStream, true);
```

The many source files from library_calls/(input|input_output) are touched by this issue.