A note on call-by-visit in the BC-Emerald implementation and break-even point.

Eric Jul, 2018-03-08

call-by-visit and call-by-move do NOT work properly in the implementation that we are using.

Here is a work-around:

As `call-by-move` and `call-by-visit` essentially are optimized versions of what a programmer can program anyway, you can simulate `call-by-visit` in the following manner, for example, given the call:

```
caller.call[move testdata]
```

replace this by:

```
move testdata to caller
caller.call[testdata]
```

Doing `call-by-visit` is more difficult because we have to ask the called object to move the parameter back, BUT the called object does NOT know from where it was called. We therefore have to add a parameter to the call that tells where to send the data object back to:

```
caller.call[testdata, self]
```

And we have to modify the called object, *e.g,*, here in the original form:

```
const caller <- object Caller
    export operation call[data: DataType] -> []
        data.test[]
    end call
end Caller
```

and here with the "manual" return of the parameter object:

```
const caller <- object Caller
    export operation call[data: DataType, returnTo: Any] -> []
        data.test[]
        move data to returnTo
    end call
end Caller
```

So find the break even point using the "simulated" `call-by-visit` as outlined above.