**Lab Book – Glenn Wilkie-Sullivan (40208762)**

Exercise 2-1:

(Multiplier Class)

// write i \* factor to outChannel

outChannel.write(i \* factor);

// read in the next value of i

i = inChannel.read();

(Consumer Class)

//insert a modified println statement

println ("New Value: " + i)

i = inChannel.read()

(RunMultiplier Class)

//insert here an instance of multiplier with a multiplication factor of 4

**new** Multiplier ( inChannel : connect1.in(),

outChannel : connect2.out() ),

Pipeline Diagram:



Output:



Exercise 2-2:

(GenerateSetsOfThree Class)

//write the terminating List as per exercise definition

outChannel.write([-1, -1, -1])

(ListToStream Class)

// hint: output list elements as single integers

**for**(i **in** 0 ..< inList.size)outChannel.write(inList[i])

inList = inChannel.read()

(CreateSetsOfEight Class)

**for** ( i **in** 0 .. 7 ) {

// put v into outList and read next input

outList.add(v);

v = inChannel.read();

}

println " Eight Object is ${outList}"

outList.clear();

Pipeline Diagram:



Output:



**Question 1 (What change is required to output objects containing six integers?)**

Within our “CreateSetsOfEight” class, we read each element of the incoming tuples and add them to the outlist, which has the given range 0-7. Simply changing this to 0-5 generates list of six.

**Question 2 (How could you parameterise this in the system to output objects that contain any number of integers (e.g. 2, 4, 8, 12) ?)**

As before, we change the range parameter to whatever size of tuple we require.

**Question 3 (What happens if the number of integers required in the output stream is not a factor of the total number of integers in the input stream (e.g. 5 or 7) ?)**

If a new list of full size cannot be created, the remainder of integers are added to the previous list.