CASE – SINGLETON

package myentitites

import case.lang.System

namespace EntitiesNameSpace {

@module

String->Object->Main

#public class Program

[public Program(String [] args)

[EntityPool Pool = EntityPool.getEntityPool]

assert(Pool) //asserts that Pool exists and has a value

Stream (n) String

Int MyInt = EntityPool.getStreamMemory() //retrieve mem from pool

Int GetInt = EntityPool.get(“MyInt”) //pointer to MyInt using pool get

//get pointer to CurrentLocationInList from the pool

Int ListStatus = n.get(“CurrentLocationInList”)

]

#end class

**Anti-Imperative**

CASE is an “anti-imperative” language, What this means is that the code for CASE doesn’t revolve around step by step interusctions, and it doesn’t revole around recursion like functional languages. CASE is entirely StateFul. An instantiateed class in CASE is an object similar to C or Java, however it is considred “live” that is, it has a specific state. One object is tied via code to another object. The state of that one object is now bound to that of the other. Example:

Public Class CoffeeMug

Int a = “Coffee Mugs” //multiple coffee mugs

Public Class Coffee

**Stateful** CofffeMug.a = Int a = “Thirsty Drinkers” **Tie** (operator)

In the above, **stateful** ties the state of the Coffee Mugs to Int a of class coffee. Now whenever CoffeeMug is changed, a correspondign change occurs in Int a of Class Coffee. The  **statement Tie** followed byan operator (in this case a generic ‘operator’)

, the same operator, that is always called whenever a change occures in Coffee Mugs. Here is another example:

Public Class Video Game Contest WInner

expertInMarioBros;

Public Class VideoGameContest

Contestants [ **stateful** Bob, Joe, Jason, Jack ] **tie** expertInMarioBros ( equiv. Bob)

Above there are four stateful contestants and one winner. According to the **tie-in** Bob is set to work, kicking backwards to the **stateful** keyword array where Bob state is sent to expertInMarioBros.

**Equiv statement**

As seen above in the anti-imperative statements, **equiv is like an assignment statement,** however equiv only takes one operand, not two. Equiv is used when the context is clearly know, just as above where we undertsand tie in experitinMarioBros is going to be the winner and we know that winnner is going to be Bob. So the first, implicit, operand is expertInMarioBros and the second, given via **equiv**, is Bob.