CASE – DEPENDENCY INJECTION

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”)

]

View can only access the model (database) through the controller, as well as the database can only access the view through the controller. Essentially it is divided into three parts. MVCs are connected via the following prototype code:

@Controller:MyAppController

String->Object->ViewClass

#public class ViewClass

#end class

@Controller:MyAppController

String->Object->ModelClass

#public class ModelClass

#end class

//The annotations above indicate what the class is connected to. Messages can be //passed in one direction from one unit to another.

//You can add more controllers simply by

@Controller:MyAppController1, MyAppController2

And more views and models by

@Controller:MyAppController

ViewClass1

….

@Controller:MyAppController

ViewClass2

And more models in the same fashion

@Controller:MyAppController

ModelClass1, ModelClass2

Finally, you have to define the controller itself

Simply, use the @Controller without definintg a :Class

Ie..,

@Controller

String->Object->Main

In this case theMain class is the controller

MVC in CASE is smart enough to update the view when the controller or model changes, as well as update the MODEL when the controller changes or the view changes and so on…

**bind** MyController MyView

Binds the MyController class to the MyView class and automatically

Keeps their data members in synchronization with each other.

If a change occurs in MyController it occurs in MyView and vice-versa.

The bindings are like variable names to which it injects.

One way databinding

**Onewaybind** MyController MyView

Only wires MyController to MyView, not the other way around