CASE PROGRAMS – Various Types of Classes

package InstantiationClasses

namespace Instantiation

{

String->Object->Main

#public class Program

//instance fields

//instance variables are declared this way

//as a shortcut to using the EntityPool declaration

instance Int IntVar

instance String happyHalloween = “Happy Halloween”

instance Double DoubleVar = 5.0d

//another way to initialize instance methods

[Int IntVarAlternate]

[String MerryChristmas = “Merry Christmas”]

[Double DoubleVarAlt = 5.0d]

[pubilc MerryCHristmas()

Print “Merry Christmas”

]

[public Initialize(String [] args)

[EntityPool Pool = EntityPool.getEntityPool]

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

//classes are for instantiating types

//CASE contains built in types like Int

//to use a non-built in type declare it as a a type

///then use it like Int for example

//**<**TypeName> ClassName myClass = new <>()

<ListDemo> AdvancedList = new <>()

stream (v) AdvancedList

MyList (v) AdvancedList

//assign a value to this instance from the class method

stream ( k ) Integer

MyAmtOfCandy ( k ) Integer = this.TrickOrTreat()

//or

MyAmtOfCandy2 (k ) Integer = TricksClass.class.CandyAmt

]

[public static Int TrickOrTreat

//this is how you define a static (class) method in CASE

//this static class is calling another static class’s variable

IntVarAlternate = testClass.returnStaticInts()

return IntVarAlternate

]

#end class

String->Object->AppleCiderClass

#public class(int constructorval1, int constructorval2) extends Program

//alternative form of a constructor

//the parameters above have a scope of initialization time

[public void baseMethod()

base.Program.Initialize() //calls this class’s base class

]

[public void Initializes()

Print “Mmmm Cider”

}