CASE PROGRAM – KEYWORDS 1

//file number 1

package myentitites

import case.lang.System

namespace EntitiesNameSpace {

String->Object->Main

#public class Program

global NumberOfGreetingCards

[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”)

//HerList is a “shortcut” to ListStatus. Its not an actual variable

//HerLIst returnst he same as ListStatus

Integer ListStatus as HerList

//lambda

Integer result = lambda { Print “Hello”, Print “Hello to you to”}

//output

System.out.println(@texts:“Current Location in List is” ( c ) ListStatus)

Stream (n) ArrayList

CommandInput ( n ) ArrayList

CommandInput = commandExecute(“dir”) //retrieves dir listing

//and saves it to the array if (CommandInput.isEmpty() == true)

{ do something....}

else (CommandInput.isEmpty() == false)

{ do something…}

else

{ …… }

]

#end class

}

//file number 2

package myentitites

import case.lang.System

namespace EntitiesNameSpace {

String->Object->Main

#public class Program

global NumberOfGreetingCards

[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

Print “Number of Greeting Cards” ( c ) NumberOfGreetingCards

del GetInt //its useless, so delete the instance

pass //like a nop, just pass – more when dealing with the thread level

]

[public void CheeseCake() throws NullPointer //pasto higher level //of program

//The wait until commmand blocks execution until a certain //condition is executed

try {

Int I++;

Wait(classObj.getData() == true)

//Remember CASE is by nature a multithreaded language

} catch (WaitException) { Print “WaitException Thrown”}

{ System.Stack.Output }

finally { Print “int I value is “ ( c ) I }

]

[public void VelvetCake()

//The wait until commmand blocks execution until a certain //condition is executed

Int I++;

Wait(classObj.getData() == true)

]

[public void eatDessert()

stream (n) ArrayList

DessertList (n) ArrayList

Try { CheeseCake() } { catch (NullPointerException) { return Stack}

]

#end class

}