CASE PROGRAM – CONTROL STRUCTURES

C.D. Jones

package myentitites

import case.lang.System

import case.lang.Image

namespace EntitiesNameSpace {

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”)

//output

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

]

[public void ifandwhilestmt(()

atom Int test = 5

while (test > 5)

{

if (test > 5)  
 { test = 3 }

else if (test < 5)

{ test = 5 }

else if (test == 0)

{continue}

else

{ return }

}

]

[public void dowhilewhere()

//slide show example

//iterate through a photoalbum. When does execute where statement

atom LastPageNumber = 10

atom CurrentPageNumber = 0

atom MiddlePageNumber = 5

atom Image [] Images

do

{

CurrentPagNumber++

Images[CurrentPageNumber]

}

While( CutrrentPageNumber >= LastPageNumnber)

=> { where PhotoAlbumTouchup() }

]

[ public void PhotoAlbumTouhup()

//do some sort of operation on the picture

]

//until structure

[public void UntilStructure()

//below

Do { int i++; i<10; String “Hello”; docs.read() } (Until i<20)

]

//residue

[public void ResidueControl()

For (int i=0;i<sizse;i++)  
 myAddress.changeAddress()

myRocket.getLaunchDate()

**residue** myRocketheight}

//results

atom height = residue //this retrieves the residue value

//The result is:  
//myRocketHeight = **reside**;

**//Residue** in this case will return the rocket height and assignt it to the lhs.

]

[public void resolveFunction()

for (String s : myStrings)

{  
 currentCtr = **resolve;**

System.out.println(“Current counter is “ (c) **resolve** (c) “);

}

]

**Leap**

Typically when you want to modify a control structure, say, you want to alter or remove an item jelly(i) in the loop for (int i=0;i<jelly.size();i++) . Typically you can’t becauase altering jelly(i) will change the state of the loop. For example, if by altering jelly(i) you might skiup over a whole block of jelly since you removed an item, and then the next time through you removed another item. Anyway, the solution to this is the **leap** keyword. Returning to our jelly example, say we identified a can of jelly numbered 5, and we want to remove it from our list of jelly jars. Simply do this:

[public void leapfunction()

for (int i=0;i<jelly.size();i++)

{  
 Print “Hello World”

Print “I don’t like strawberry jam”

**leap** jelly.remove(Strawberry) //removing this in java would cause //problems

}

This is equivalent to C:

Int remove = 0;

For (int i=0; i<jelly.size();i++)  
{  
 cout << “Hello World” << endl;

cout << “I don’t like strawberry jam;

if (I == Strawberry)

{

remove = I;

}

}

jelly.remove(remove);

[public void simpleForLoop()

for (<5) { Print “Print 5 times” }

//automated while loop

//Typically a while appears as such:

While (a<5)

{

}

We can make the syntax so it is:

While (<5)

{ Print “Printing 5 times” }

{ Print “Current Counter” wcounter }

]

#end class

}