CASE PROGRAMS – CONTROL STRUCTURES 2

package myentitites

import case.lang.System

import case.lang.Image

import case.lang.Spool

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 foreachloops()

Foreach (TypeClass iteratorVariableName : Collection)

]

[public void iterativeloops()  
 //**Iterative For Based Loops**

For (variable initialize; conditional test; increment)

For exasmple, f(int i=0; i<10;i++)

[public void

//**A Simpler For Statement**

Often times more information is processed then need be for a for loop.That is, a for loop reallly can be just:

for (<5) { Print “Print 5 times” }

[public void autowhile()

**//An automated While Loop**

**//**

//Typically a while appears as such:

//

//While (a<5)

//{

//}

//We can make it so it is:

While (<5)

{ Print “Printing 5 times” }

{ Print “Current Counter” wcounter }

//In this example wcounter is an automated variable: a variaable that’s built into //the language.

[public void oldfashionedswitch()

Switch(t)

{

case (t == 5.0)

{returnString = 50}

**exit**; //passes control out of the current scope

case (t == 0)

{returnString = “Wow its cold”}

**pass;** //passes control compeltely to the root scope

case (t==”Crackers”)

{returnString = “I’m hungry”}

**break**; //escapes from the switch statement

case (t==”GetFood”)

{goto buyFood}

}

]

[public void buyFood()

System.out.println(“Hungry for some food”)

System.out.println(“I nee dto go shopping”)

**Label** buyFood

System.out.println(“Going to buy Food”)

[public void casesswitch()

Instead of using the keyword using **break** we use the keyword **fall,** causing the execution to fall to the nexdt case. And as for default, we simply use default as normal. Example:

Switch(t)

{ ==5.0 } {returnString = 50, fall}

{==0 } { returnString = “Wow its cold”}

{Crackers} { returnString = “I am so hungry”, fall}

{==5} {returnString = “Where is my space ship?”, break}

{default} {returnString = “No idea”}

]

[public void branching()

//we are going to launch multiple thraeds using the Spool class

//creates an instance of the funtion along with either the id or

//name or anynonymously

Spool.createThread(casesswitch(), 1)

Spool.createThread(casesswitch(), myCasesThread)

Spool.createThread(casesswitch)

Spool.craeateThread(casesswitch(), blocking)

//blocking makes it so when the thread terminates it still hangs around

//for examination until told to terminate

Spool.kill(1)

Spool.kill(myCasesThread)

//entites allow us to access thread by variable name

EntityPool.accessThread(myCasesThread, returnString)

]