CASE PROGRAM – FOR LOOPS

**Requires**

This keyword in a source file means that anyone who wants to

Use the source code via an import must specifically declare it using the keyword **satisfies** in the using source code.

import case.lang.System

namespace HelloWorld {

//String->Object->Main is a type constructor

String->Object->Main

#public class Program

@requires

[public Program(String [] args)

System.out.println(@texts:“Hello World”)

//type one of the for loop

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

{

…dosomething…

}

// type two of the for loop

{

//Residue

//Most iterative loops use a counter. The value “residue” //contains a loop variable’s value when the loop ends. For //example:

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

[myROcket.getLaunchDate()]

residue myRocketheight;

}

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

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

//Type 3

**Resolve**

When using a for each loop often it is handy to see the iterative side, that is, the index side of the for each loop, instead of the blind iterator.

For (String s : myStrings)

{  
 currentCtr = **resolve;**

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

}

//Type 4

**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:

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

{  
 Print “Hello World”

Print “I don’t like strawberry jam”  
 **leap** jelly.remove(Strawberry)

}

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);

}

]

//Type 5

**Iterator**

**Foreach Based Loops**

Foreach (TypeClass iteratorVariableName : Collection)

**//Short form of Foreach - Type6**

Foreach (Collection)

{ Print arrayListOfStrings(itervar) }

//The short from of foreach uses default iterator variable called itervar.

//The typelass is inferre from the Collection type.

//Type7

**Iterative For Based Loops**

For (variable initialize; conditional test; increment)

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

**List Comprehensions and Generator Expresions:**

[(i,j) for i in range(3) for j in range(i) ]

((i,j) for i in range(4) for j in range(i) )

**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” }

Where the parathesis is the boolean condition and the braces are the execution.

**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.

}