CASE – TRYCATCHMECHANISMS

Dynamic Annotations Example

//enable dynamic: type checking done at runtime

//

package trycatch

namespace trycatch {

@dynamic

import case.lang.System

import case.lang.System.IO

import case.lang.System.IO.IOException

import case.lang.System.ArrayIndexOutOfBounds

import case.lang.System.IO.PrintWriter

import case.lang.legacy.Print

import case.lang.IOStreamException

import case.lang.FileInputStreamException

import case.lang.ArrayIndexStream

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

myString (n) String

Try {

myString = someExceptionThrowingFunction;

}

Catch(IOException)

{  
StackTrace;

}

print(myString);

So instead why don’t we just:

**tag** String myString;

myString = someExceptionThrowingFunction;

Taggign the variable does several things. First it tells the compiler that this variable is going to be tracked by an execption system. So if there is an exception that happens, it will throw an exception. Secondmost, and foremost, it even keeps the scope level. So instead of the mess of nulls, your FileStreamReader for example, will be able to be accessed in any of the new lines.

//Another new type catching figure is

**ChainCatch**

Or you can chain exceptions….so an alternative:

**tag ( IOStreamException)**

**tag ( FileInputStreamException)**

**tag(ArrayIndexStream)**

String myString;

myString = someExceptionThrowingFunction;

The three tags are applied to the myString variable;

]

#end class

}