CASE PROGRAM – TOKENIZE

@microserve //turns on microservices

package MyTokenizePackage

import case.lang.System.InputOutput

import case.lang.Systen.FileInputOutput

import case.lang.System.BufferedFileInput

Import case.lang.System.Tokenizer

Import case.lang.System.InputOutpout.ConsolePrint

namespace TokenizerSpace

{

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

TokensString (n) String

TokensString = BufferedFileInput(“inputstringfile.txt”)

Stream (t) Tokenizer

StringTokenizer (t) Tokenizer

//initialie

Token (n) String

while (StringTokenizer.hasNext())

{

Token = StringTokenizer.nextElement()

//no simple print hello is function name for pool purposes

//Token is an argument

//the brackets are the ‘functionality’

Token => SimpleTokenizer => { ConsolePrint(Token) }

}

]

#end class

}

//The architecture grows as demand increases using microservices. The //microservices use a we service style architecture. When one microservice dies or //completes, CASE replaces it. These things come and go. When a microservice is //finished, it is torn down. The process is fast enough that the user should not even //know it is done. Services are small and live and die fast, such as AWS lambda, they //are spawned for a service then die. This is automatically handled behind the scenes