Taller # 01 | Generador de Números Pseudo Aleatorios y Pruebas de Bondad

Caicedo Hidalgo Geraldine [1527691] [geraldine.caicedo@correounivalle.edu.co]

Abstracto – Este documento presenta el análisis y explicación del taller número 1 de simulación computacional, el cual consiste en una implementación de un generador de números pseudo aleatorios GEM y comparar con el generador propio del lenguaje de programación usado empleando las pruebas de bondad, Chi Cuadrado, Poker 2 y 3.

Índice de Términos – pseudo aleatorios, generador, pruebas de bondad, GEM, chi cuadrado, poker.

# GENERACION DE NUMEROS PSEUDO ALEATORIOS

Para implementar el generador de números pseudo aleatorios se utilizó el lenguaje Java, utilizando el generador visto en clase (GEM) y el generador de números de aleatorios provisto por Java.

Se garantiza en el codigo que el generador tenga una semilla diferente de 0.

# PRUEBAS DE BONDAD

Se realizarán 3 pruebas diferentes a cada generador, Chi Cuadrado, Poker con dos decimales y con tres decimales.

Las pruebas se harán para dos cantidades de datos, 1.000 y 10.000.

La prueba de **Chi Cuadrado** se utiliza para comparar el comportamiento de los datos generados con la distribución uniforme .

En esta prueba se calcula el número de clases, los grados de libertad los cuales para las cantidades de datos es:

Para 1.000:

Para 10.000:

Se clasifican los datos en clases de frecuencia obtenida, se calcula la frecuencia esperada y el chi cuadrado calculado para cada clase, como es uniforme, la frecuencia esperada es igual para todos.

Que para las dos cantidades de datos seria:

Para calcular chi cuadrado calculado se utiliza la siguiente formula:

Teniendo en cuanta que el valor de confianza dado para calcula chi cuadrado es se tiene el valor del chi cuadrado critico según el número de datos.

Las pruebas de **Poker** se utilizan para analizar si los datos generados se comportan como datos verdaderamente aleatorios.

Para esta prueba se toman dígitos decimales de un número pseudo aleatorio como una mano de k cartas de un juego de cartas, que están numeradas de 0 a 9.

Si estos números son independientes, deben ocurrir los mismos patrones, y con las mismas probabilidades que en un juego de poker:

* Todas las cartas iguales
* Cartas en orden creciente o decreciente

Se buscan patrones que sucedan en secuencias aleatorias de números del 0 al 9, agrupados en sub secuencias de tamaño (las manos). Los patrones deben ser mutuamente disjuntos y exhaustivos.

Se calculan las probabilidades teóricas de cada patrón .

Se aplica la prueba chi cuadrado donde las calases corresponde a los patrones, las frecuencias observadas se determinan con las manos y las frecuencias esperadas de un patrón corresponde a

Para la prueba de **2** decimales las frecuencias observadas son:

* 2 números iguales
* 2 números diferentes

Para la prueba de **3** decimales las frecuencias observadas son:

* 3 números iguales
* 2 iguales y 1 diferente
* 3 números diferentes

Irene

## GENERADOR GEM

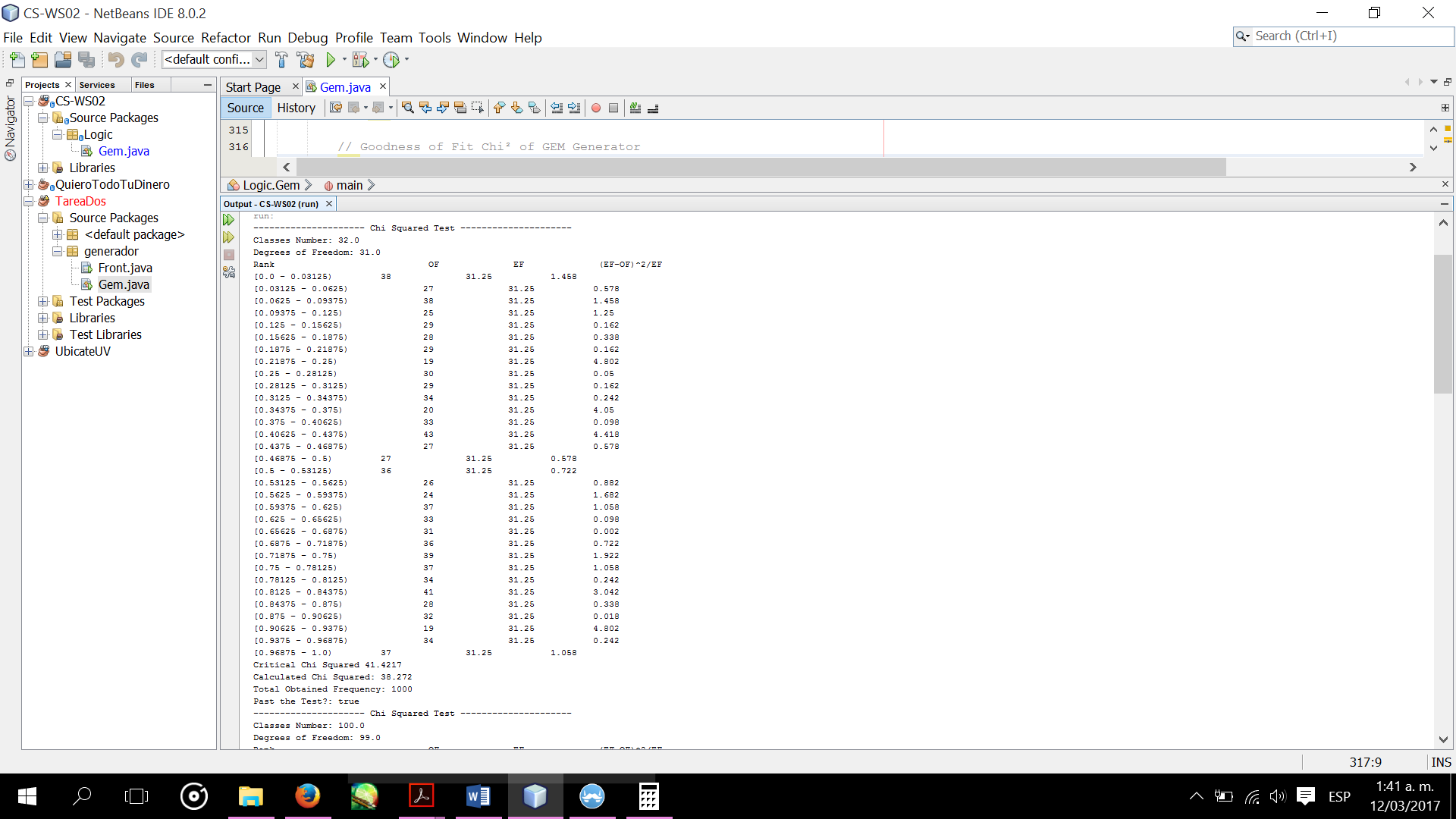
El generador GEM se implementa con la relación de recurrencia:

Donde es el multiplicador; es el incremento y es el modulo.

Se tiene , por eso .

### Chi cuadrado

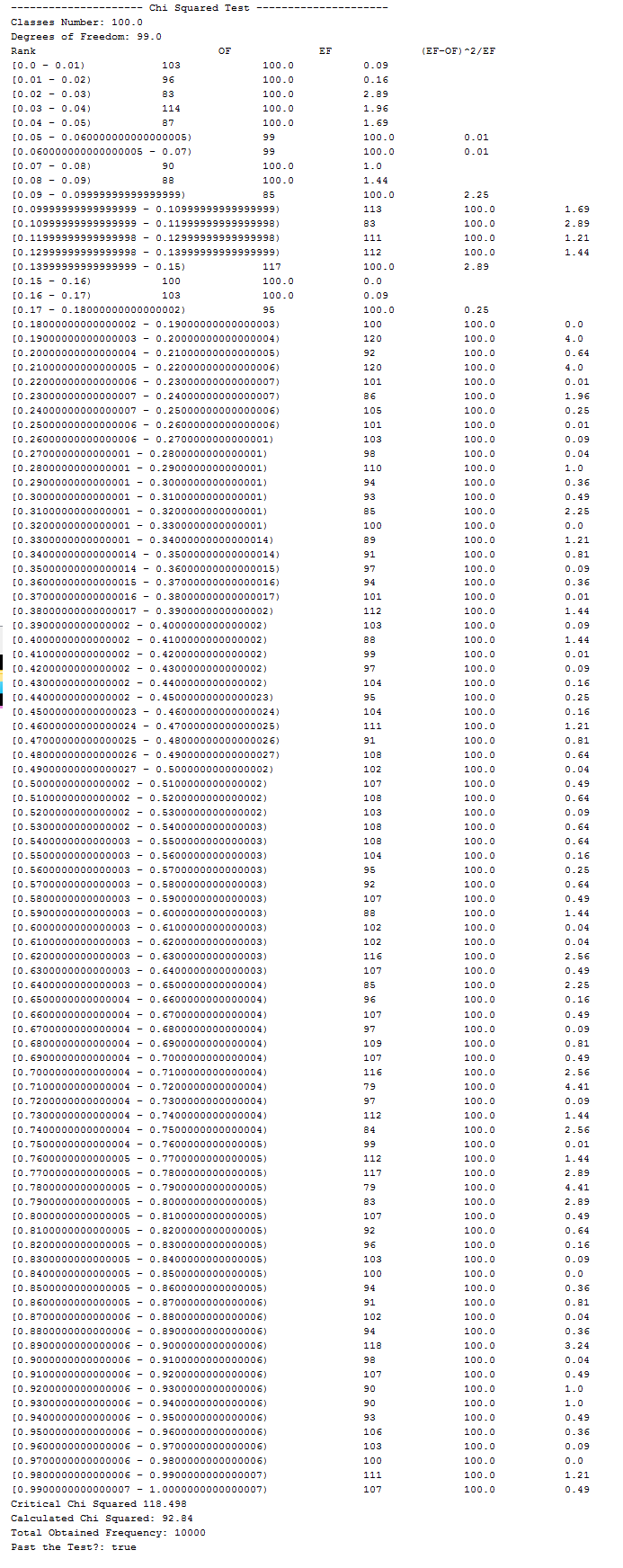
En la *Imagen 1* se muestra el resultado del test Chi Cuadrado para 1.000 números pseudo aleatorios con el generador GEM.



**Imagen 1.** Resultados Chi Cuadrado de 1.000 números con Generador GEM

En este caso se puede ver que el generador paso la prueba, ya que el chi cuadrado critico es mayor que el chi cuadrado calculado.

En la *Imagen 2* se muestra el resultado del test Chi Cuadrado para 10.000 números pseudo aleatorios con el generador GEM.

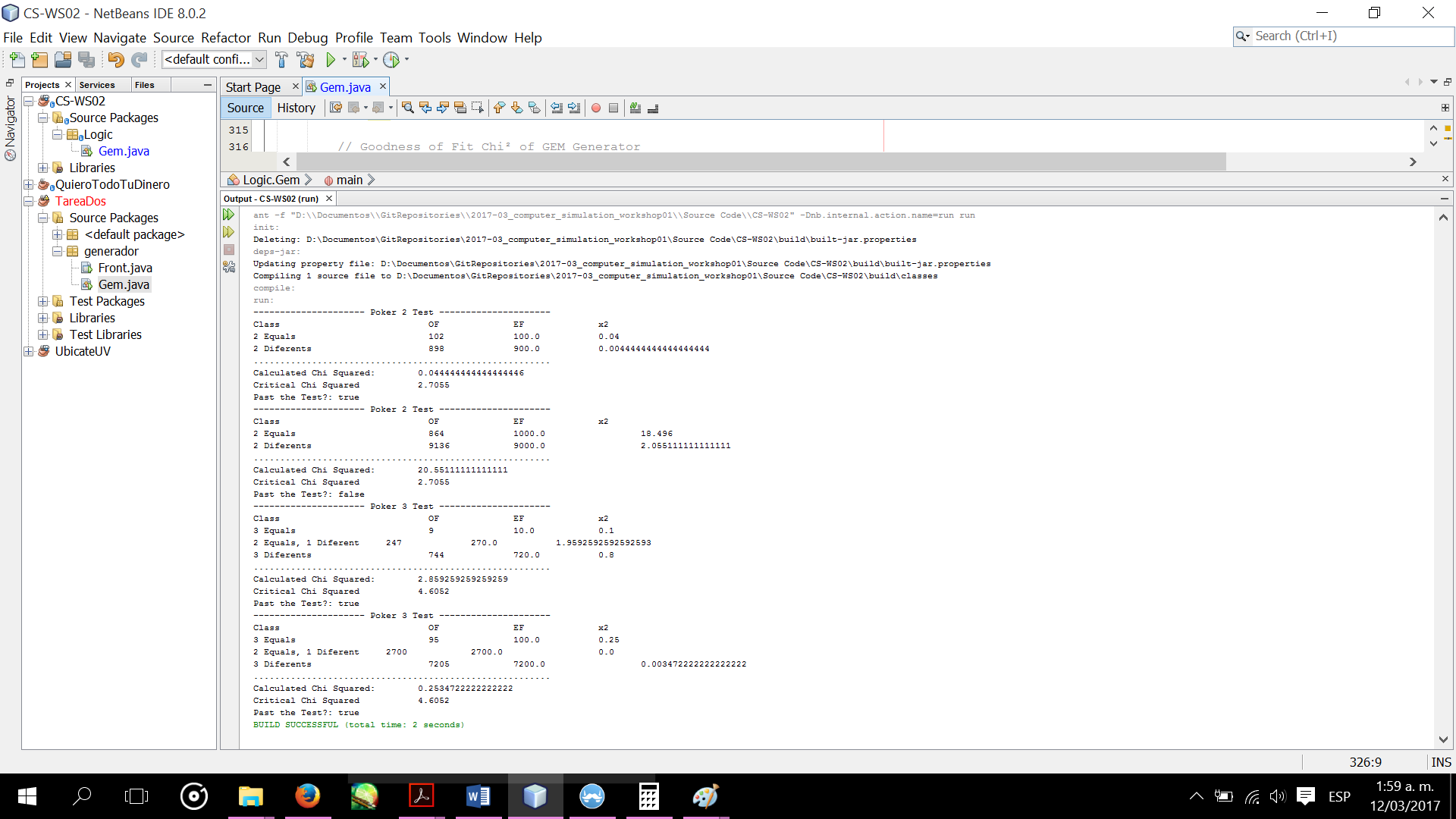


**Imagen 2.** Resultados Chi Cuadrado de 10.000 números con Generador GEM

También aquí se puede ver que el generador paso la prueba, ya que el chi cuadrado critico es mayor que el chi cuadrado calculado.

### Poker 2

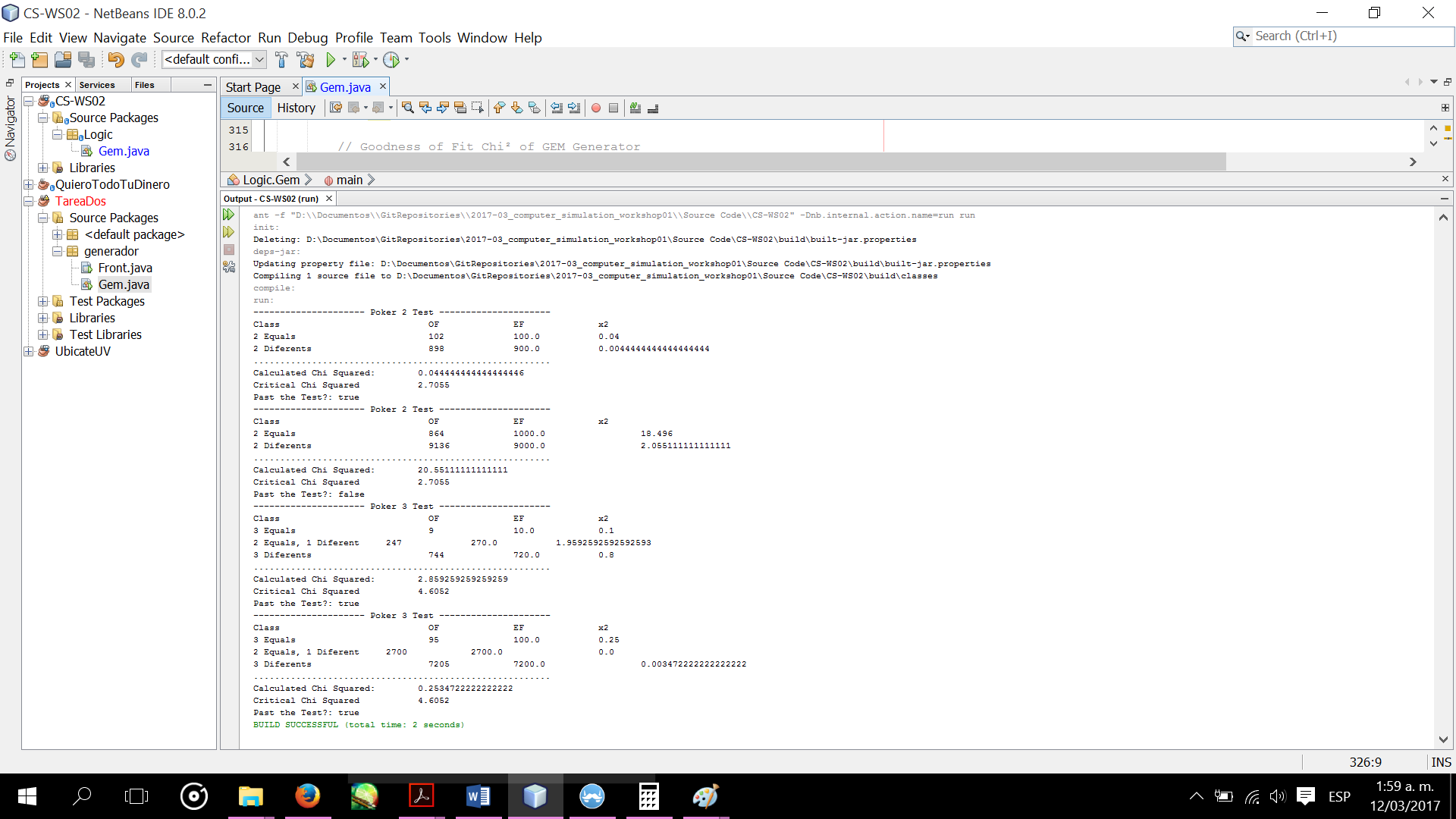
En la *Imagen 3* se muestra el resultado del test Poker con 2 decimales para 1.000 números pseudo aleatorios con el generador GEM.



**Imagen 3.** Resultados Poker 2 decimales de 1.000 números con Generador GEM

Aqui se puede ver que el generador paso la prueba, ya que el chi cuadrado critico es mayor que el chi cuadrado calculado.

En la *Imagen 4* se muestra el resultado del test Poker con 2 decimales para 10.000 números pseudo aleatorios con el generador GEM.

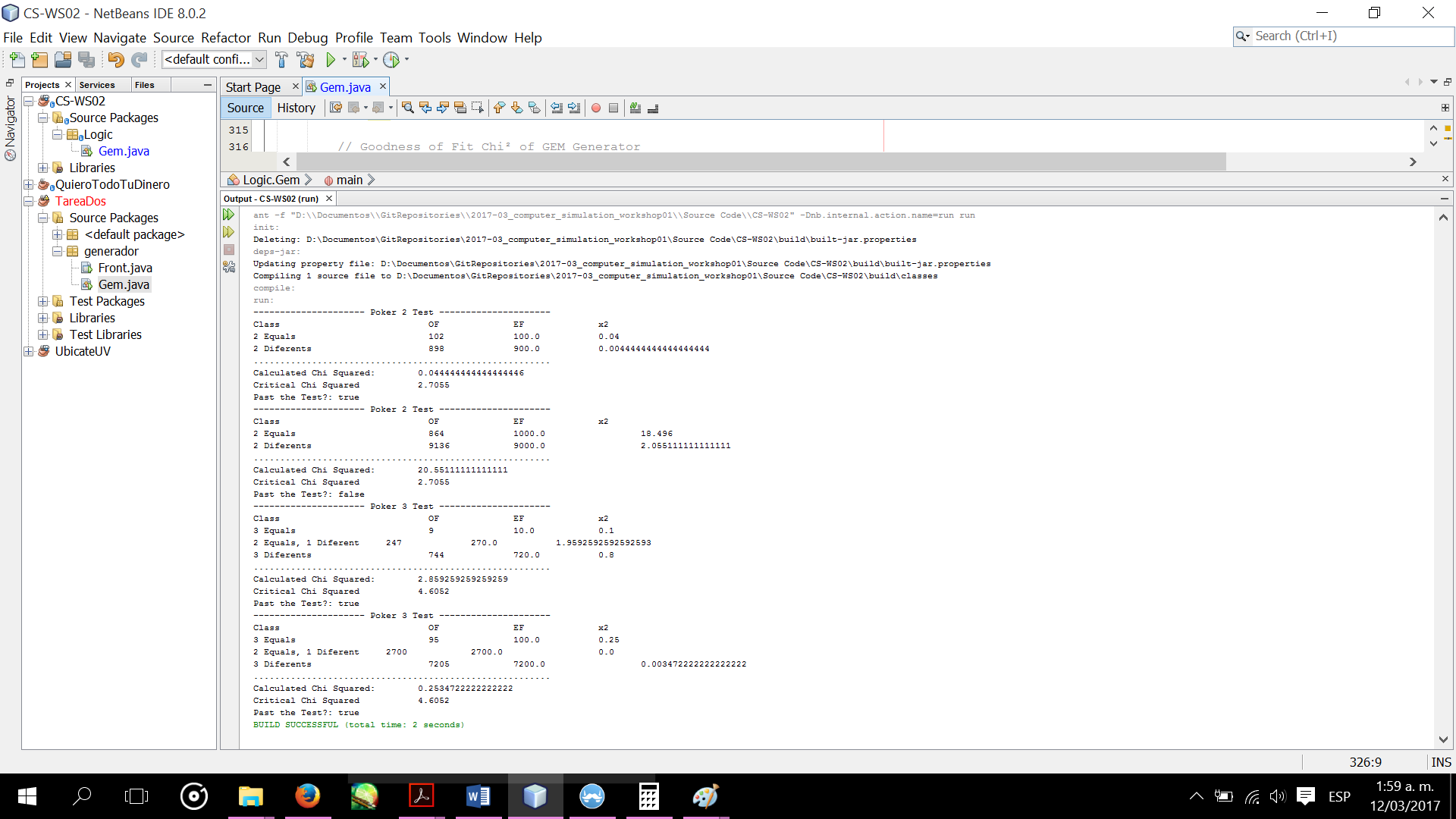


**Imagen 4.** Resultados Poker 2 decimales de 10.000 números con Generador GEM

En este caso se puede ver que el generador no paso la prueba, ya que el chi cuadrado critico es menor que el chi cuadrado calculado.

### Poker 3

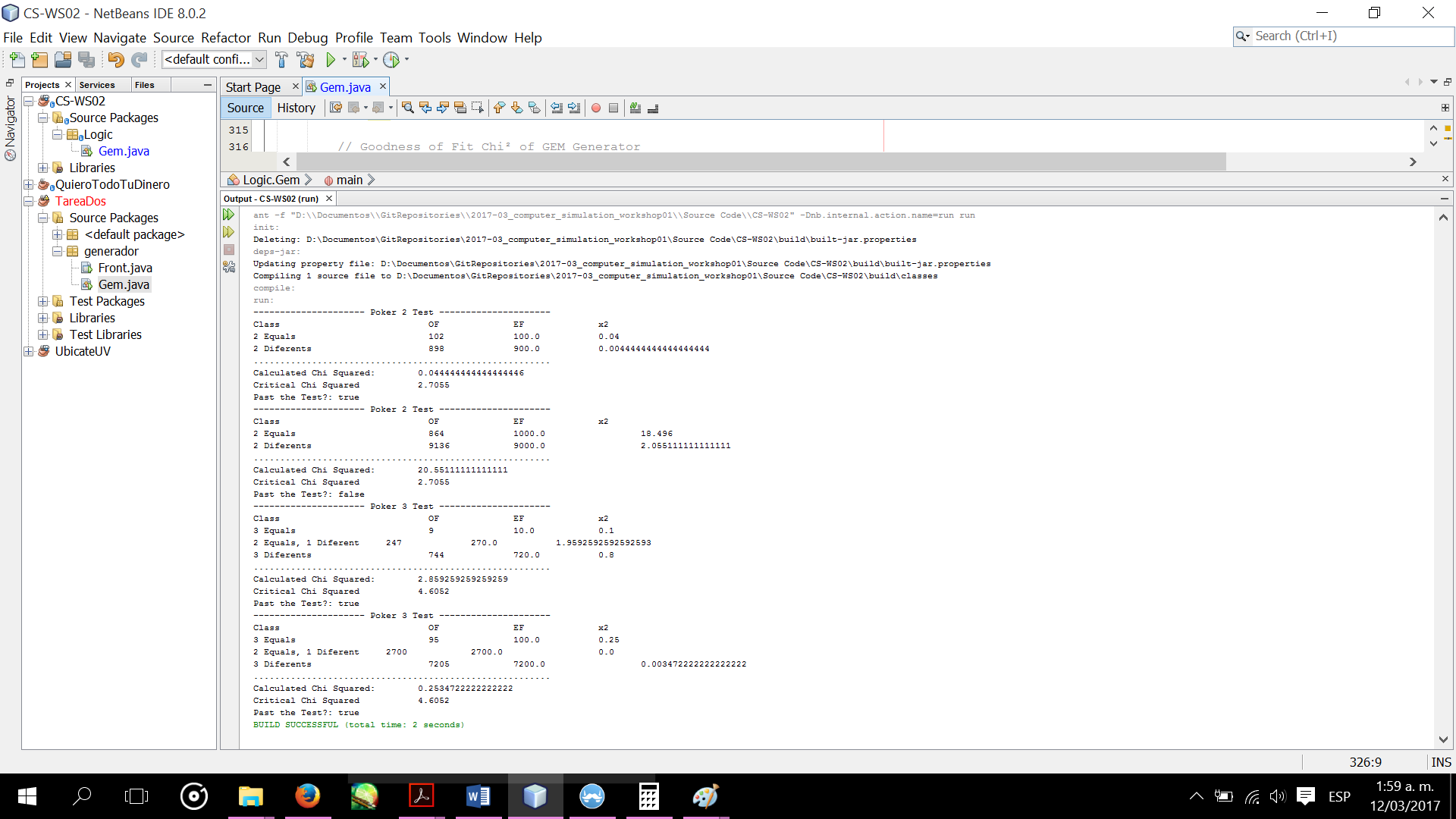
En la *Imagen 5* se muestra el resultado del test Poker con 3 decimales para 1.000 números pseudo aleatorios con el generador GEM.



**Imagen 5.** Resultados Poker 3 decimales de 1.000 números con Generador GEM

En esta prueba se puede ver que el generador paso la prueba, ya que el chi cuadrado critico es mayor que el chi cuadrado calculado.

En la *Imagen 6* se muestra el resultado del test Poker con 3 decimales para 10.000 números pseudo aleatorios con el generador GEM.



**Imagen 6.** Resultados Poker 3 decimales de 10.000 números con Generador GEM

En este caso también se puede ver que el generador paso la prueba, ya que el chi cuadrado critico es mayor que el chi cuadrado calculado.

## GENERADOR JAVA

Al elegir el lenguaje de programación Java se utilizó como generador de números pseudo aleatorios alternativo Util.Random para la comparación con el generador GEM.

Este generador utiliza un generador lineal congruente y sus parámetros son tomados del generador UNIX rand48 con una ligera variación en la función de la semilla.

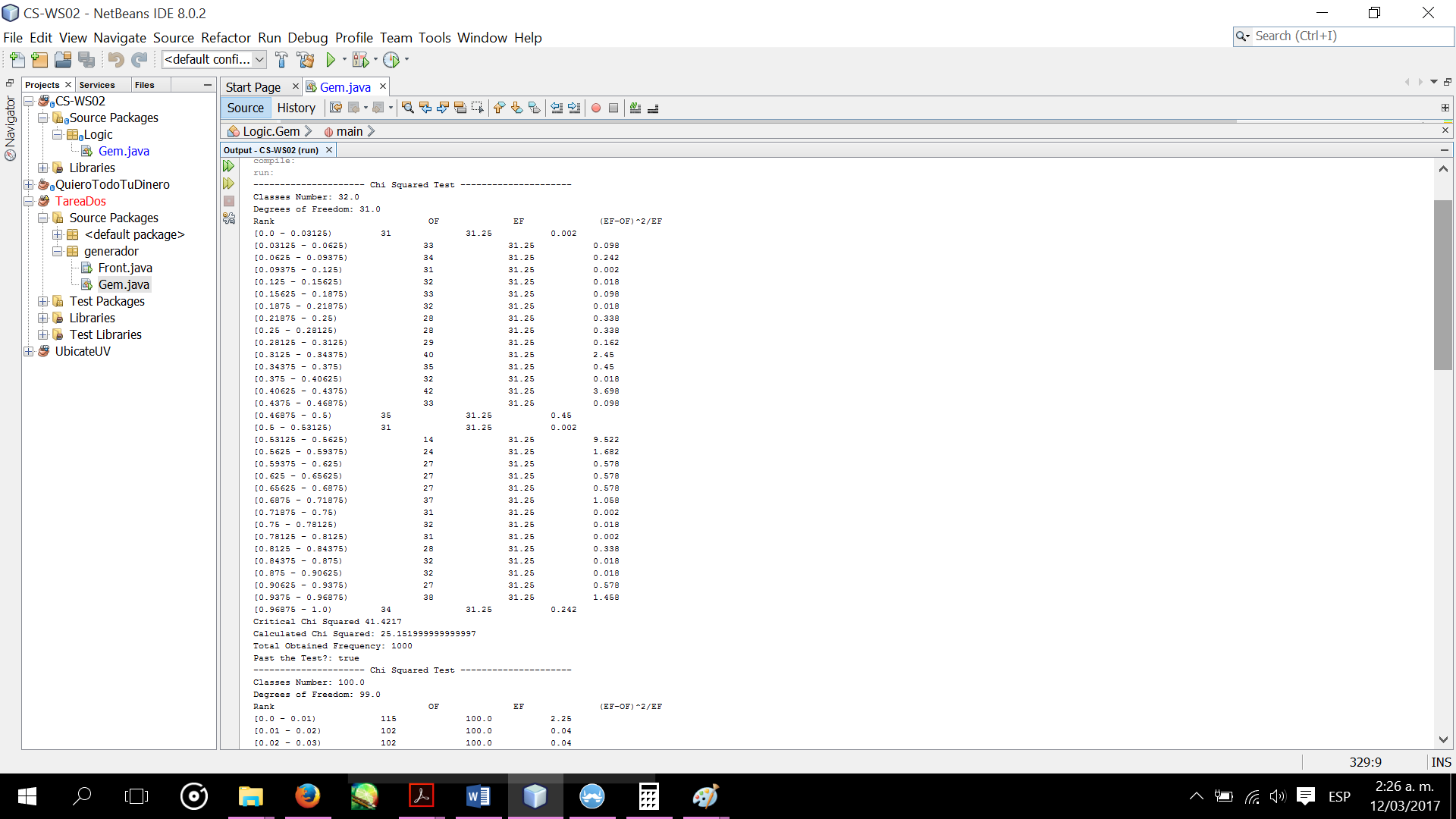
El periodo de este generador es , y no puede producir todas las combinaciones de pares de 2 enteros posibles, y sus bits no son equitativamente aleatorios.

### Chi Cuadrado

En la *Imagen 7* se muestra el resultado del test Chi Cuadrado para 1.000 números pseudo aleatorios con el generador Java.

Alli se puede ver que el generador paso la prueba, ya que el chi cuadrado critico es mayor que el chi cuadrado calculado.

En la *Imagen 9* se muestra el resultado del test Chi Cuadrado para 10.000 números pseudo aleatorios con el generador Java.

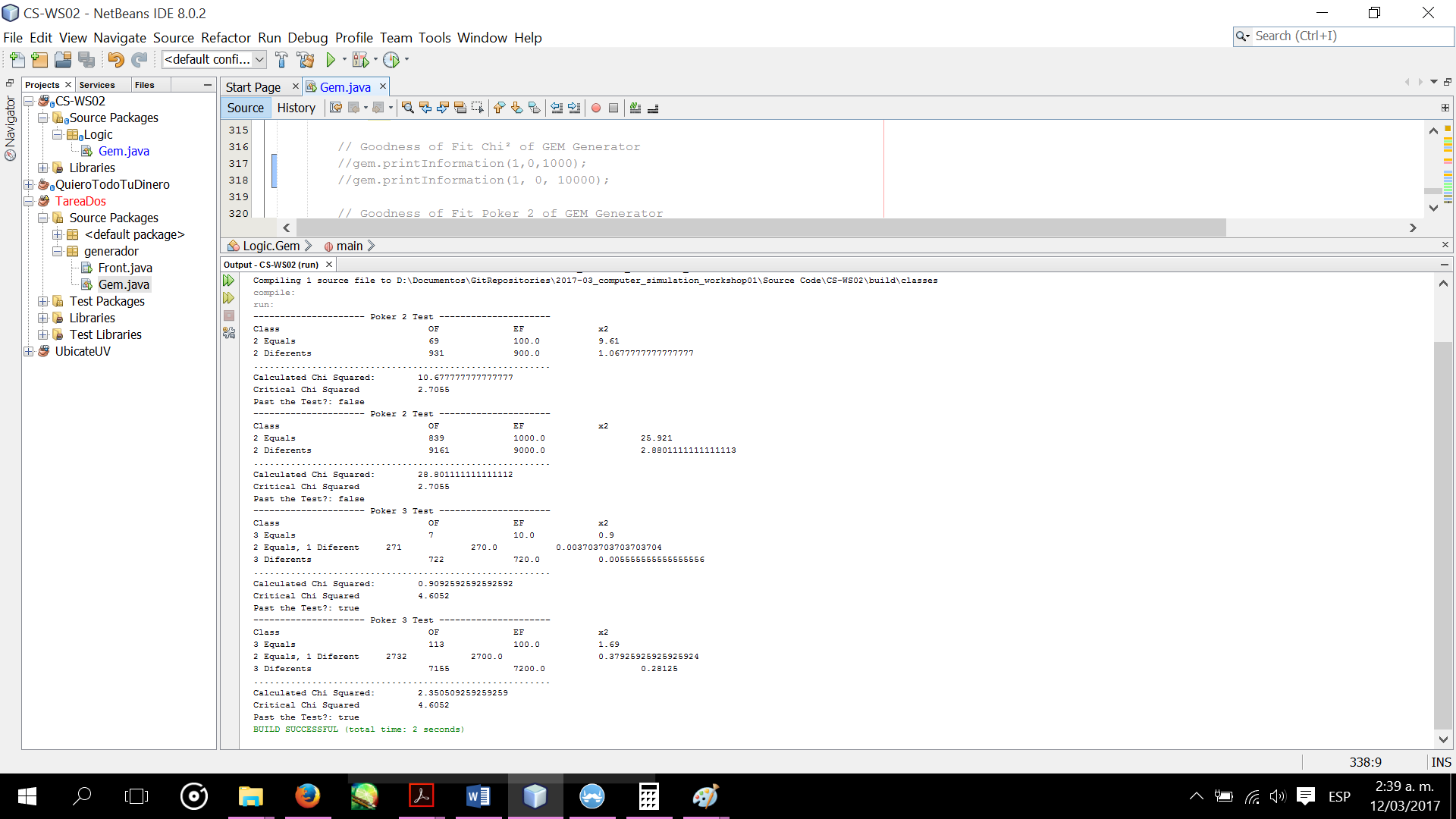


**Imagen 7.** Resultados Chi Cuadros de 1.000 números con Generador Java

En la *Imagen 9* se puede ver que el generador paso la prueba, ya que el chi cuadrado critico es mayor que el chi cuadrado calculado.

### Poker 2

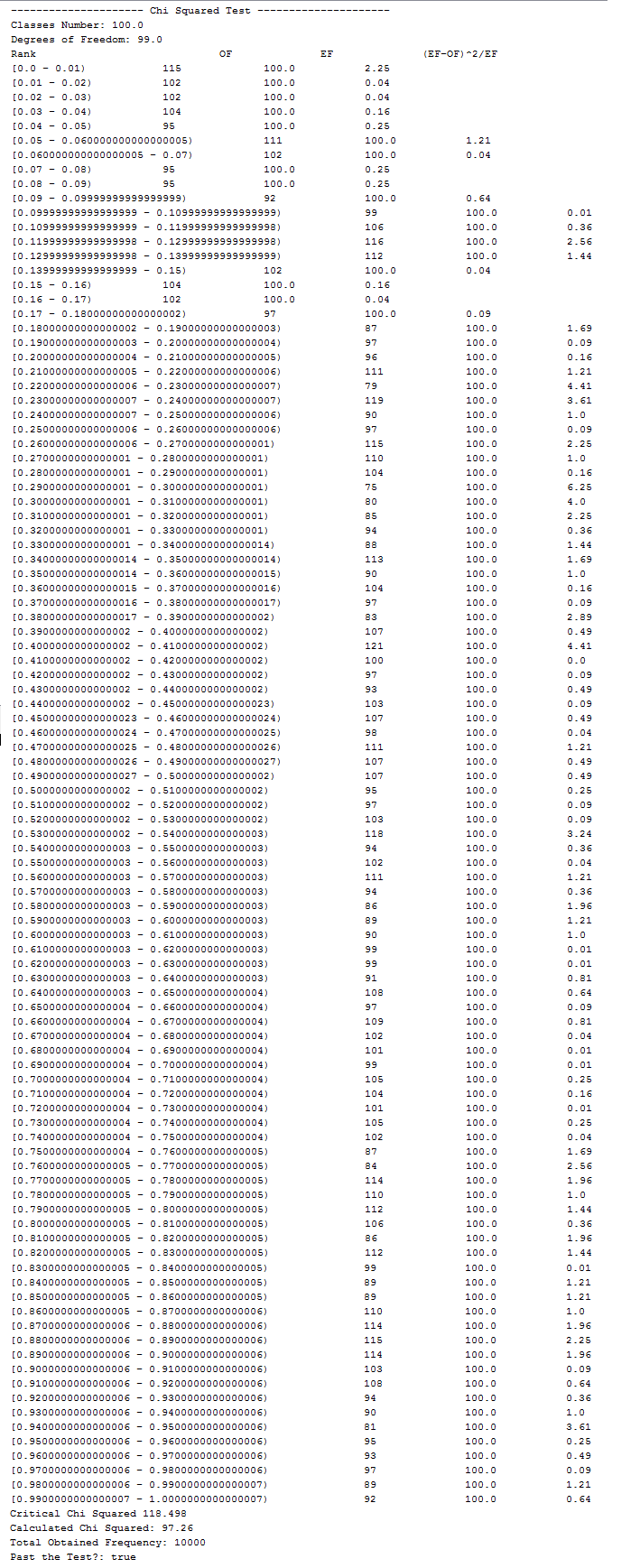
En la *Imagen 8* se muestra el resultado del test Poker con 2 decimales para 1.000 números pseudo aleatorios con el generador Java.



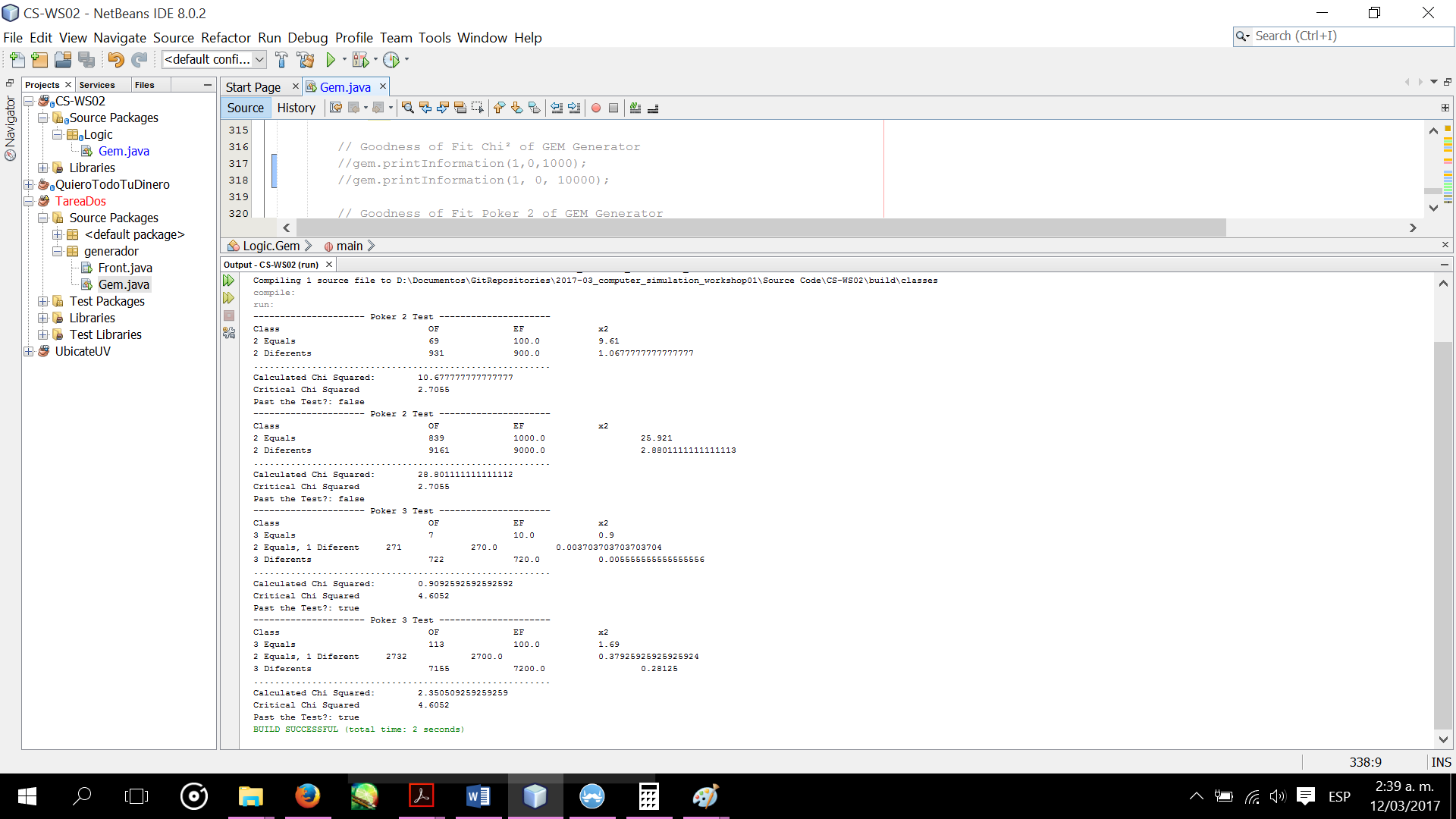
**Imagen 8.** Resultados Poker con 2 Decimales de 1.000 números con Generador Java

Este test muestra que el generador no paso la prueba, ya que el chi cuadrado critico es menor que el chi cuadrado calculado.

En la *Imagen 10* se muestra el resultado del test Poker con 2 decimales para 10.000 números pseudo aleatorios con el generador Java.



**Imagen 9.** Resultados Chi Cuadros de 10.000 números con Generador Java

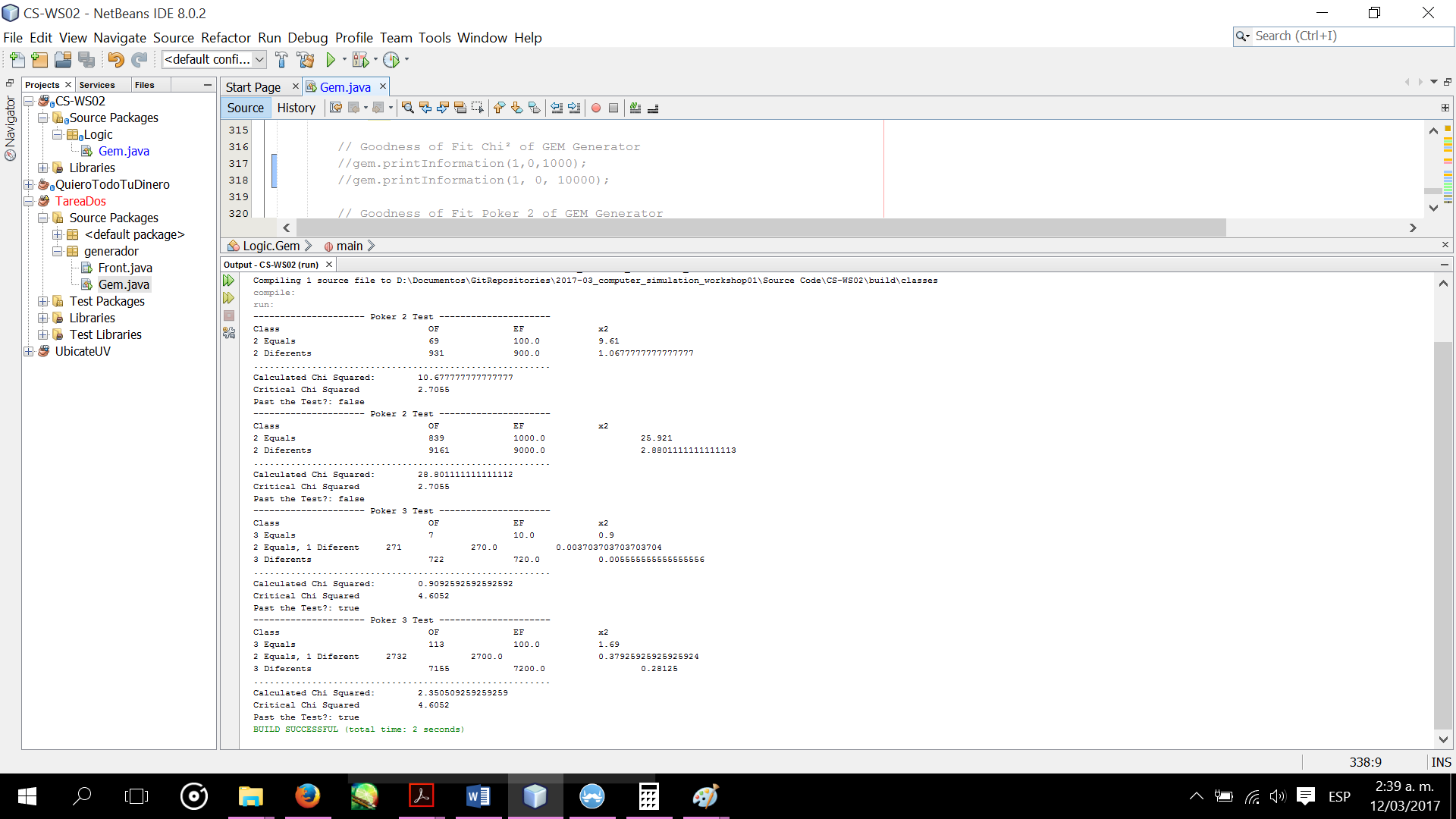


**Imagen 10.** Resultados Poker con 2 Decimales de 10.000 números con Generador Java

Aquí se muestra que el generador no paso la prueba, ya que el chi cuadrado critico es menor que el chi cuadrado calculado.

### Poker 3

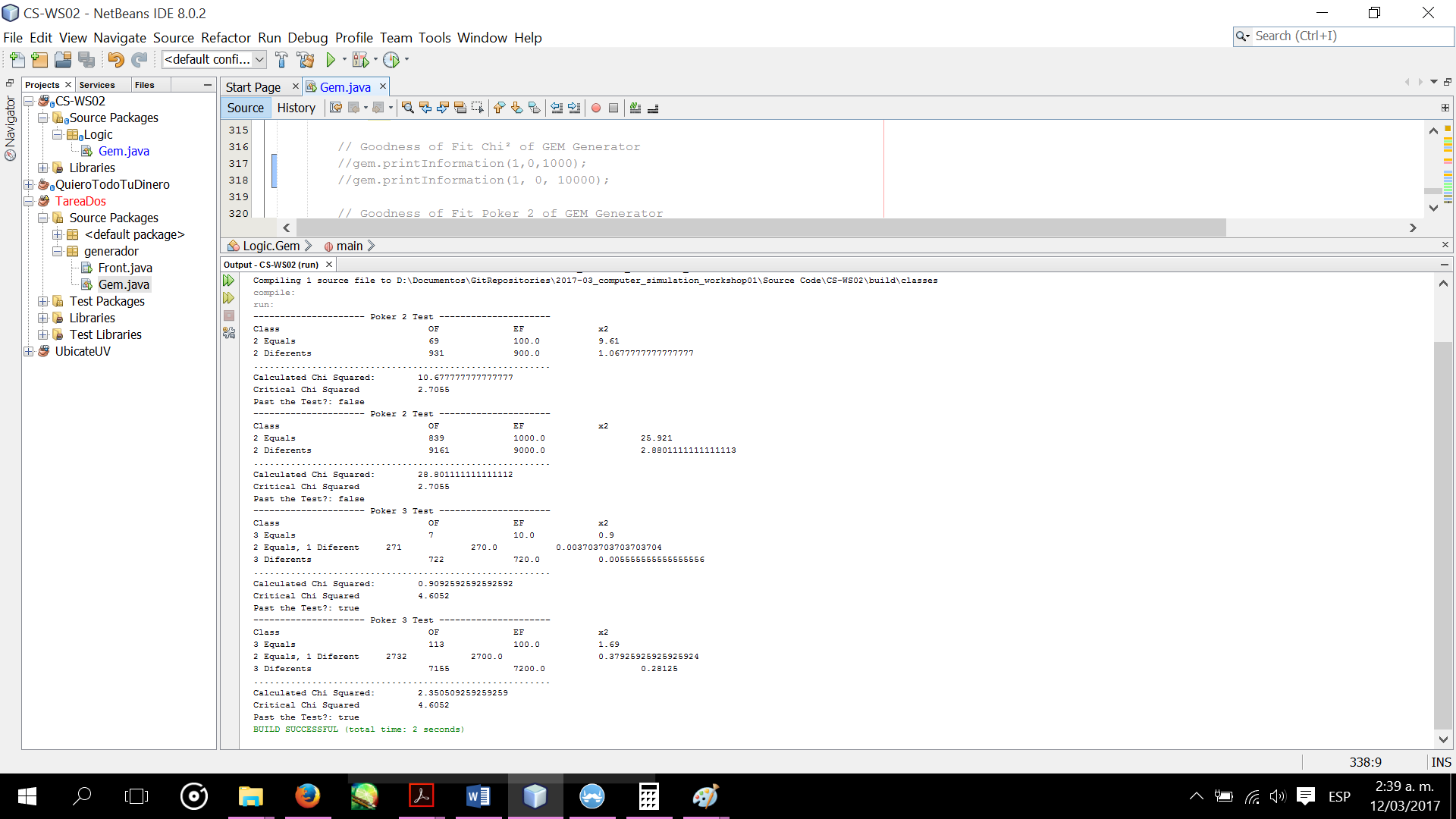
En la *Imagen 11* se muestra el resultado del test Poker con 3 decimales para 1.000 números pseudo aleatorios con el generador Java.



**Imagen 11.** Resultados Poker con 3 Decimales de 1.000 números con Generador Java

En esta imagen se muestra que el generador paso la prueba, ya que el chi cuadrado critico es mayor que el chi cuadrado calculado.

En la *Imagen 12* se muestra el resultado del test Poker con 3 decimales para 10.000 números pseudo aleatorios con el generador Java.



**Imagen 12.** Resultados Poker con 3 Decimales de 10.000 números con Generador Java

Aquí se muestra que el generador paso la prueba, ya que el chi cuadrado critico es mayor que el chi cuadrado calculado.

## ANALISIS DE LOS RESULTADOS

Según los resultados arrojados por las pruebas, el generador GEM implementado solo falla para las pruebas **de Poker con dos decimales** con una cantidad de números generados de 10.000, y el generador propio del lenguaje de programación Java, falla en las pruebas de independencia, **Poker con dos decimales** en las dos cantidades de números probadas, 1.000 y 1000.

Es decir que en las pruebas de uniformidad los dos generadores son aceptables, pero en las pruebas de independencia, pasan solo con la prueba de **Poker con tres decimales**.

## BIBLIOGRAFIA

* Diapositivas de Clase – Simulación Computacional [Marzo 2017]
* Randomness of bits with LCGs – Neil Coffey - UK - [2013]

[http://www.javamex.com/tutorials/random\_numbers/lcg\_bit\_positions.shtml]

* Java.lang.Random falls “mainly in the planes” – Neil Coffey - UK - [2013] [http://www.javamex.com/tutorials/random\_numbers/lcg\_planes.shtml]
* How does java.util.Random work and how good is it – Neil Coffey - UK - [2013] [http://www.javamex.com/tutorials/random\_numbers/java\_util\_random\_algorithm.shtml#.WMS6W39yyvB]