**Universidad Nacional Autónoma de México**

**Facultad de Ingeniería**

Laboratorio de docencia

Laboratorios de computación

Laboratorios de computación A y B

Práctica #10 Excepciones y errores

|  |  |
| --- | --- |
| *Profesor:* | Tista García Edgar |
| *Asignatura:* | Programación Orienta a Objetos |
|  | *Grupo: 3* |
| *No de Práctica(s):* | 10 |
| *Integrante(s):* | Félix Flores Paul Jaime |
|  |  |
|  |  |
| *Semestre:* | 2019-2 |
| *Fecha de entrega:* | 16-04-2019 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

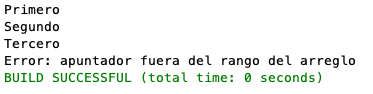
**OBJETIVO.**

Identificar bloques de ccódigo propensos a generar errores y aplicar técnicas adecuadas para el manejo de situaciones excepcionales en tiempo de ejecuciónn.

**DESARROLLO.**

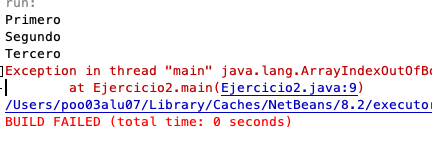
**Ejercicio 1) Manual de laboratorio.**

En este ejercicio vimos cómo manejar una excepción se utilizan las palabras reservadas try y catch. El bloque try es utilizado para definir el bloque de código en el cual una excepción pueda ocurrir. El o los bloques catch son utilizados para definir un bloque de código que maneje la excepción.



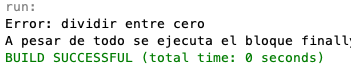
**Ejercicio 2) Manual de laboratorio.**

Si no se captura la excepción interviene un manejador por defecto que normalmente imprime información que ayuda a encontrar dónde y cómo se produjo el error, sin embargo, como la excepción no es controlada, ésta se propaga hasta llegar al método principal y termina abruptamente el programa.



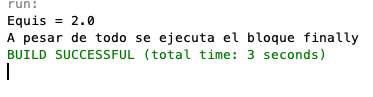
**3) Manual de laboratorio.**

La palabra reservada finally permite definir un tercer bloque de código dentro del manejador de excepciones. Este bloque le indica al programa las instrucciones a ejecutar de manera independiente de los bloques try-catch, es decir, si el código del bloque try se ejecuta de manera correcta, entra al bloque finally; si se genera un error, después de ejecutar el código del bloque catch ejecuta el código del bloque finally.



**4)Manual de laboratorio.**

Sin generar excepción:



**5)Manual de laboratorio.**

Si se invierte el orden, se capturan las excepciones de lo más general a lo más específico, todas las excepciones caerían en la primera (Exception es la más general) y no habría manera de enviar un error del tipo IOExcepction o ClassNotFoundException. Esta situación la detecta el compilador y no permite generar el bytecode, es decir, no compila.

Captura de pantalla 2019-04-11 a la(s) 11

**6) Manual de laboratorio.**

No es obligatorio tratar las excepciones dentro de un bloque manejador de excepciones, pero, en tal caso, se debe indicar explícitamente a través del método. A su vez, el método superior deberá incluir los bloques try/catch o volver a pasar la excepción. De esta forma se puede ir propagando la excepción de un método a otro hasta llegar al último método del programa, el método main.

Captura de pantalla 2019-04-11 a la(s) 11

**7) Manual de laboratorio.**

Esta sintaxis obliga que al utilizar el método se deba realizar dentro de un manejador de excepciones o dentro de un método que indique que va a arrojar la misma excepción.

Captura de pantalla 2019-04-11 a la(s) 11

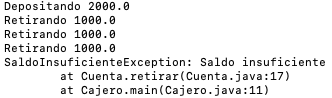
Captura de pantalla 2019-04-11 a la(s) 11

**8) Manual de laboratorio.**

Se crea la clase *Cuenta*, la cual podrá lanzar una excepción de tipo *SaldoInsuficienteException* si se intenta retirar un monto mayor al saldo de la cuenta:

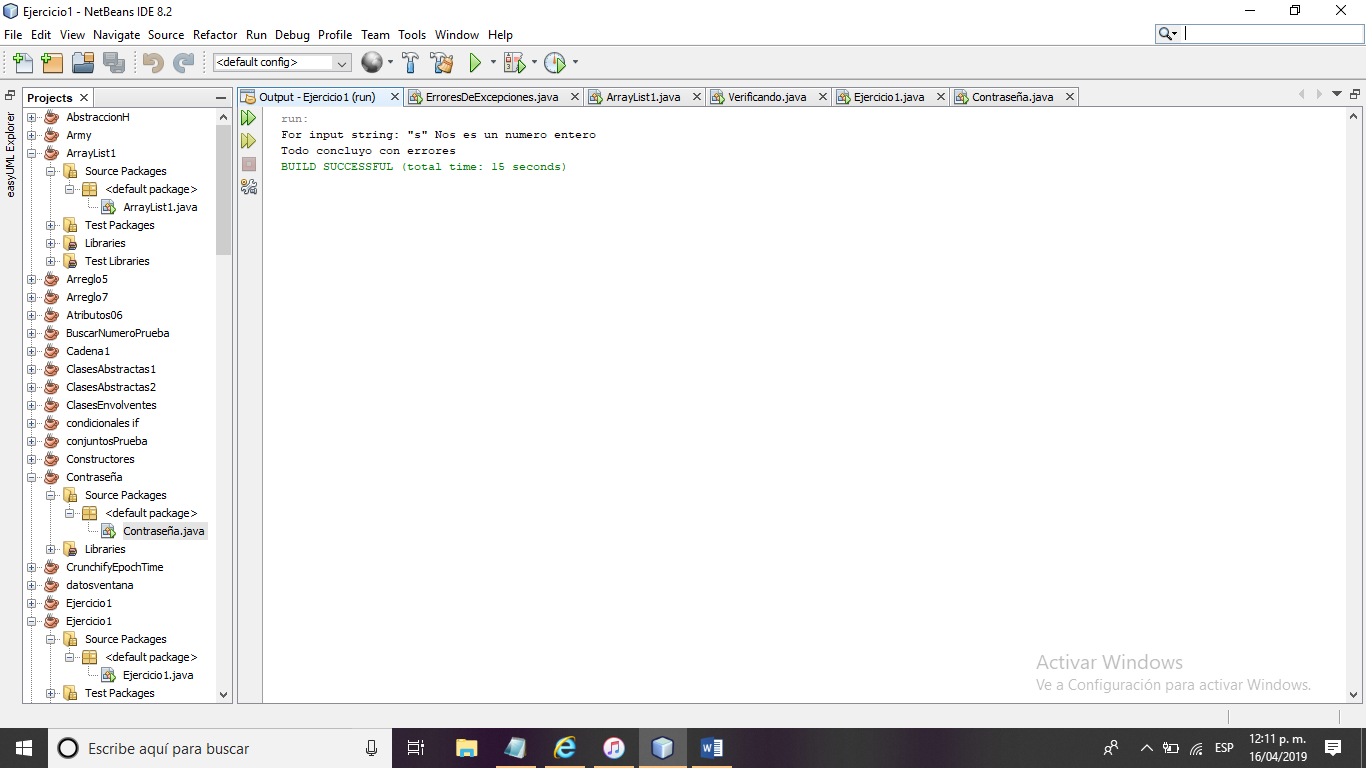
Finalmente, para probar el funcionamiento de la clase *Cuenta* y su correspondiente

excepción, se crea una cuenta *Cajero* donde se emulan depósitos y retiros a una cuenta:

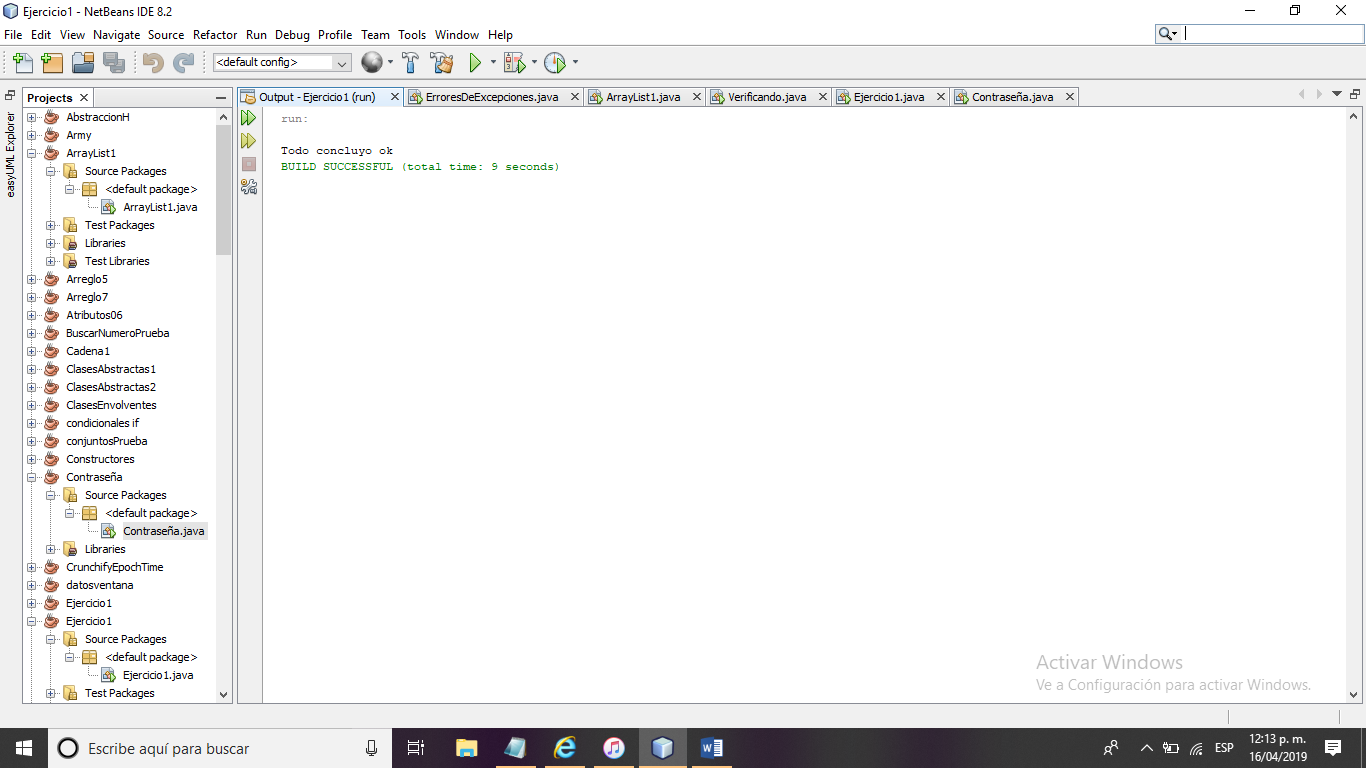
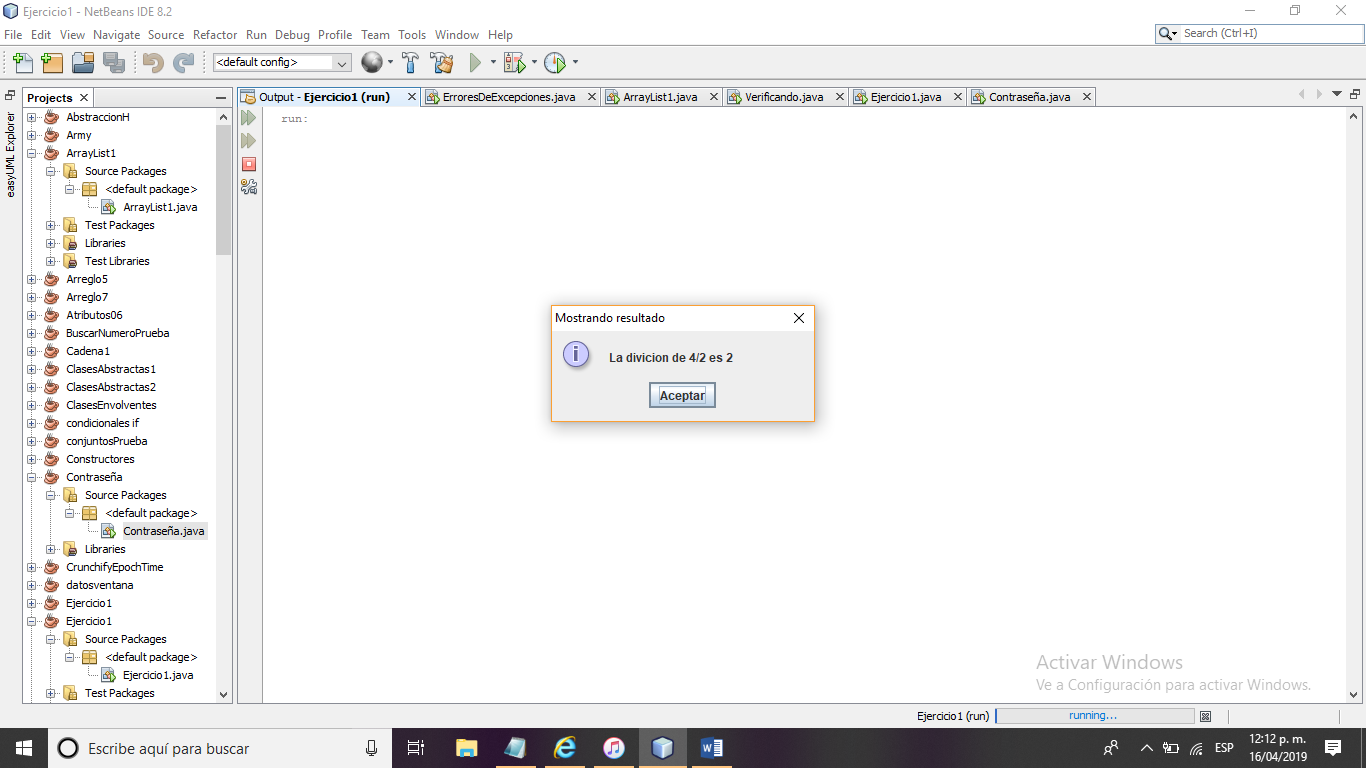


**Ejercicio 1)**

Para este ejercicio me salió, bien al momento de que el usuario ingresara una letra el programa nos iba a mandar

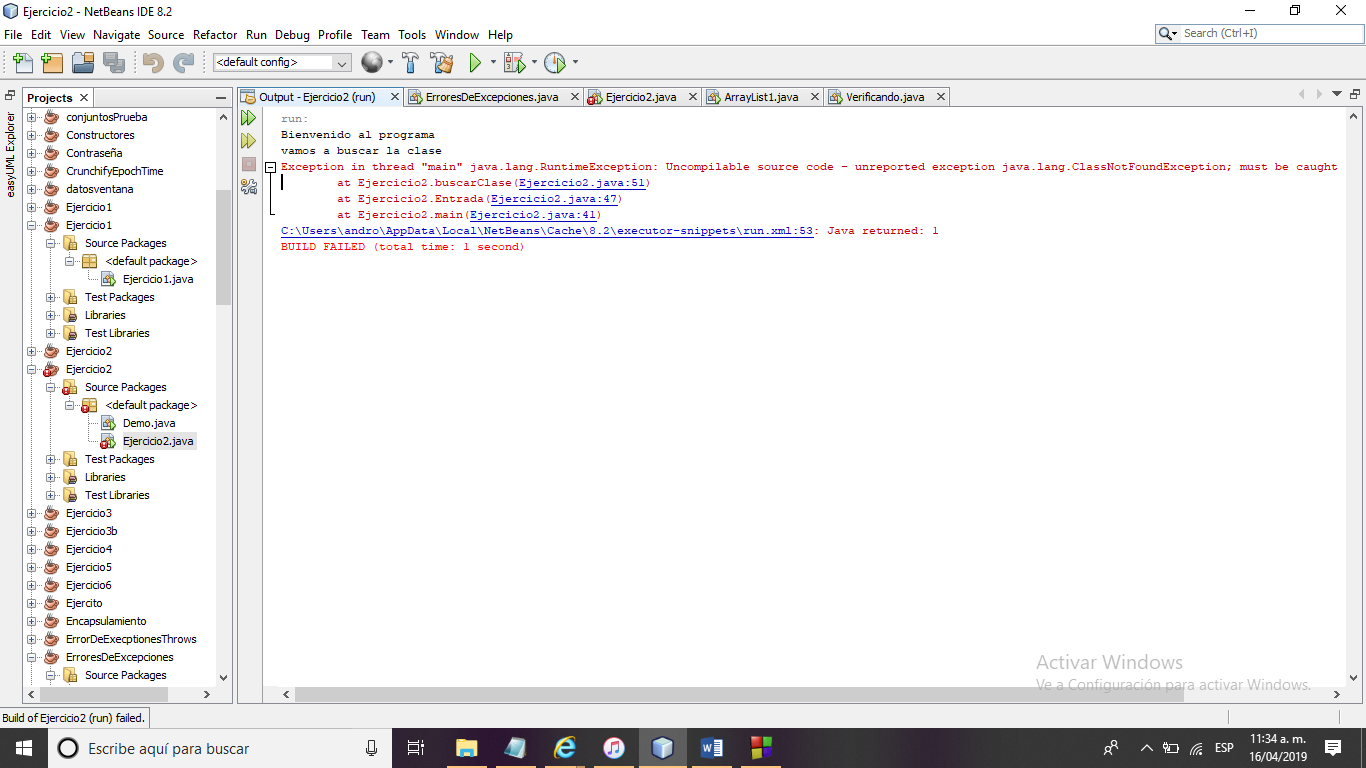


En caso de que todo resulte bien

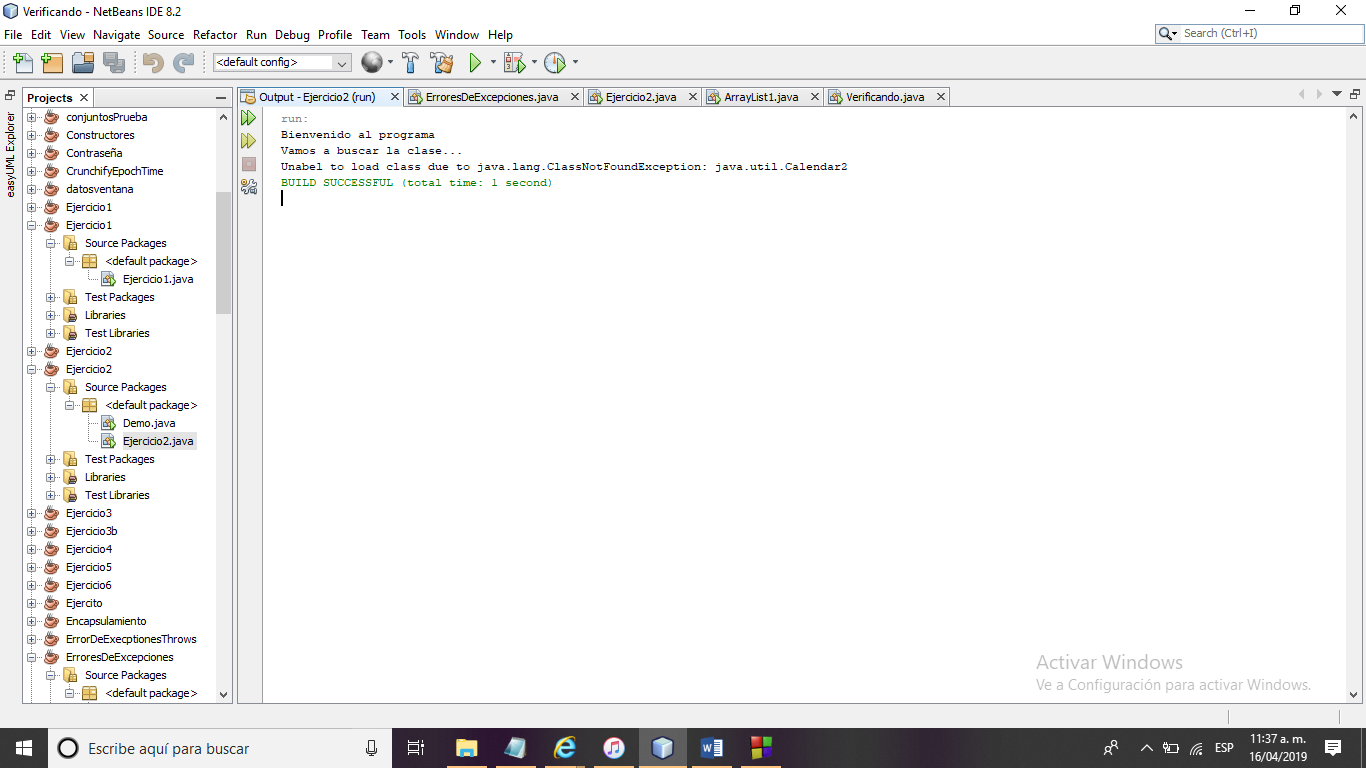


**Ejercicio 2)**

Para este ejercicio lo corregimos mediante el Try –catch, por que al momento de compilar nos mandaba los siguientes errores en pantalla.

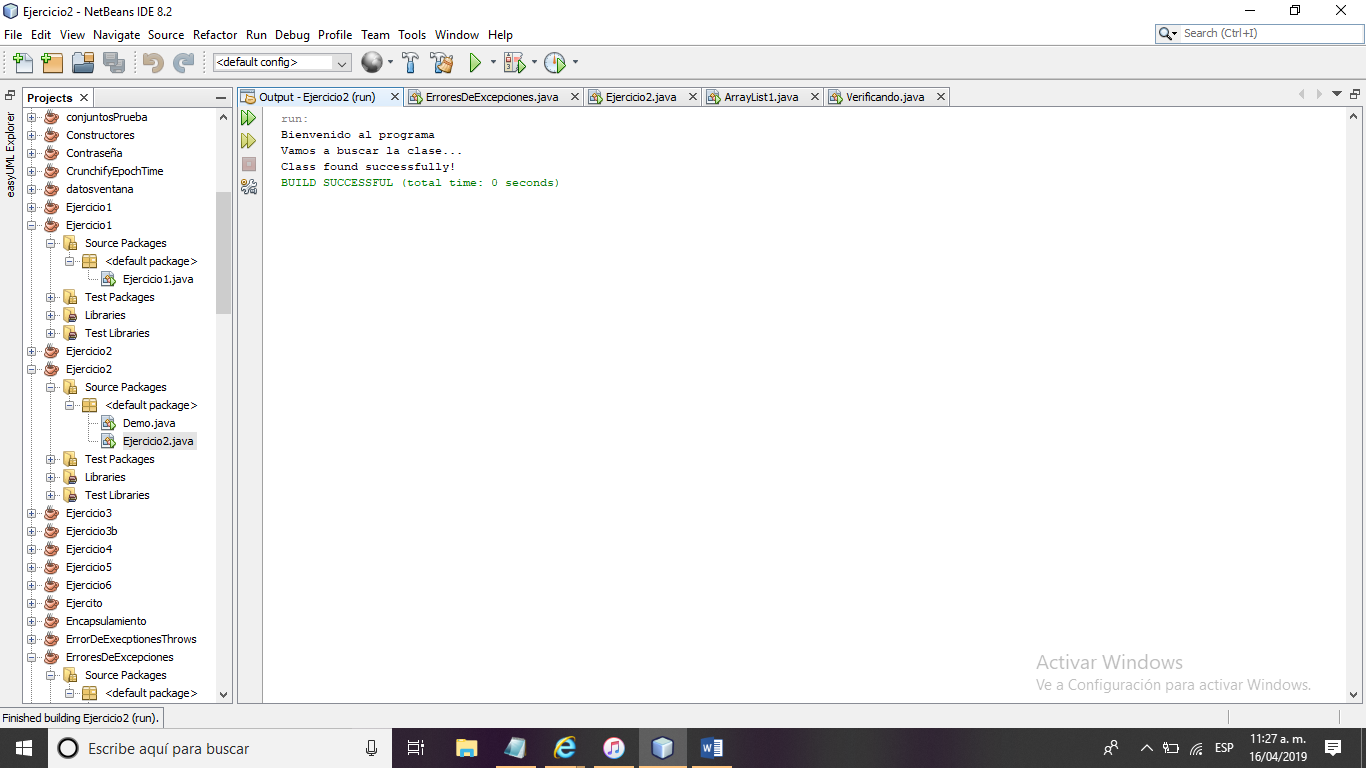


Una vez corregido con el try catch nos mandaba esto, donde la clase no había sido encontrada. Y la razón no era porque el programa no funcionara ,sino porque la clase no estaba.



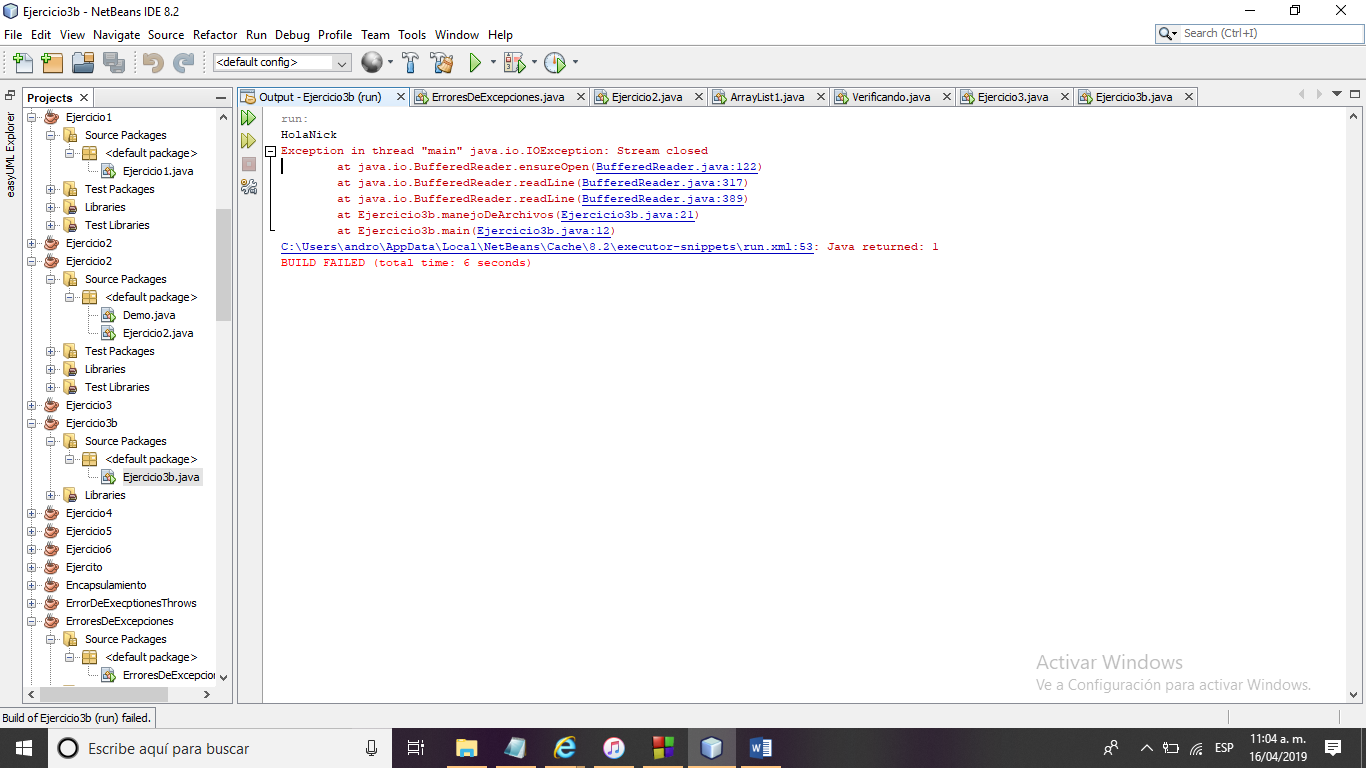
Para el ejercicio hice otra clase llamada “Demo” a la cual iba a buscar mediante el programa corregido.

Una vez encontrado nuestra clase nos mandaría el siguiente aviso donde si encontró a la clase llamada “Demo “si cambiamos a la clase buscar “Demos” el programa nos decía que no se encontraba, así pude corroborar los datos.

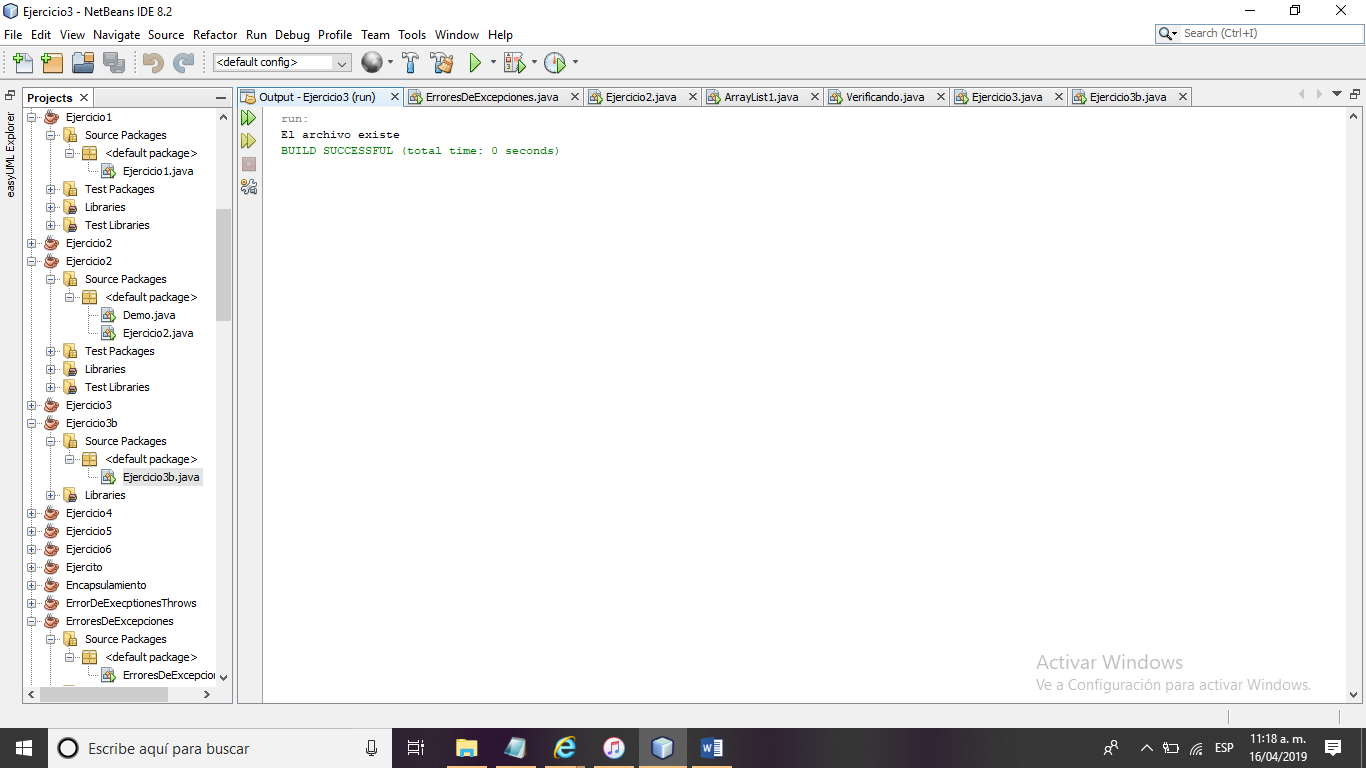


**Ejercicio 3)**

The program doesn't compile, because the function main() uses FileReader() and FileReader() throws a checked exception FileNotFoundException .El programa no se compila, porque la función main () usa FileReader () y FileReader () lanza una excepción comprobada FileNotFoundException . It also uses readLine() and close() methods, and these methods also throw checked exception IOException También usa los métodos readLine () y close (), y estos métodos también arrojan una excepción comprobada IOException



Para arreglar el programa anterior, necesitamos especificar la lista de excepciones usando throws, o usar el bloque try-catch. We have used throws in the below program. Usaremos throws en el siguiente programa. Since FileNotFoundException is a subclass of IOException , we can just specify IOException in the throws list and make the above program compiler-error-free. Dado que FileNotFoundException es una subclase de IOException ,asi podemos especificar IOException en la lista de lanzamientos y hacer que el programa anterior esté libre de errores.



**Conclusiones.**

En esta práctica se logró repasar el tema de manejo de excepciones, así como las variaciones de este tema, es interesante ver cómo tratarlas y que hacer para que nuestros programas funcionen de la manera más correctamente posible.

Esta práctica me costó en cuanto hacer cosas sencillas como al hacer el bucle para tres oportunidades, lo intente hacer mediante la lógica de los programas de contraseñas, pero como que se atrofiaron mis conocimientos en esa parte, no creo que sea algo grave, solo repasar temas anteriores, porque me salió el programa con excepciones.

Creo que lo más complicado fue el razonamiento del ejercicio 2 y 3, ya que debíamos razonar con el try catch de cómo funcionaba

Esta práctica me ayudo a reforzar mis conocimientos sobre excepciones y el cómo tratarlas, así como reforzar los conocimientos vistos en clase y con los ejercicios hechos en ella.