ESCUELA COLOMBIANA DE INGENIERÍA

PROGRAMACIÓN ORIENTADA A OBJETOS

2020-1

Laboratorio 2/6

Integrantes:

* Juan Andrés Pico
* Ana Gabriela Silva

**Conociendo el proyecto**

1. En este proyecto encontramos diferentes clases, una de ellas representa una calculadora de matrices de fraccionarios. La clase principal es llamada “fraccionario” e implementa el tipo de dato fraccionario; es decir, un número racional que se puede escribir de la forma p/q, donde p y q son enteros, con q <> 0.
2. El proyecto tiene 4 clases, (“Calmatfra”, “Matriz”, “Fraccionario”, “FraccionarioTest”), la relación entre ellas es que todas son un complemento para realizar la calculadora de fraccionarios, ya que una de ella pide los números, otra los organiza y por último se operan.

La clase principal reconocida es “Calmatfra” ya que es la encargada del comportamiento del software. Las clases diferentes serian: Matriz, guarda los fraccionarios a operar; Fraccionario, implementa el tipo de dato fraccionario; FraccionarioTest, se encarga de probara la funcionalidad de las diferentes clases que existen en el proyecto

1. Falta la documentación de la clase “Calmatfra”, “Matriz”, “FraccionarioTest”. Igualmente, de la clase Calmatfra ninguno de los métodos está documentados. En la clase Matriz, los métodos dimensión y get no están documentados.
2. Calmatfra, falta codificación de los métodos.

Matriz, falta codificación de los métodos.

Fraccionario, falta codificación de los métodos.

FraccionarioTest, completo.

**Ingeniería reversa**

**2.** Los contenedores que hay en las colecciones son:

* HashMap, la clase HashMap es equivalente a una tabla de hash a diferencia de que esta es desincronizada y permite nulls.

**Conociendo Pruebas en BlueJ**

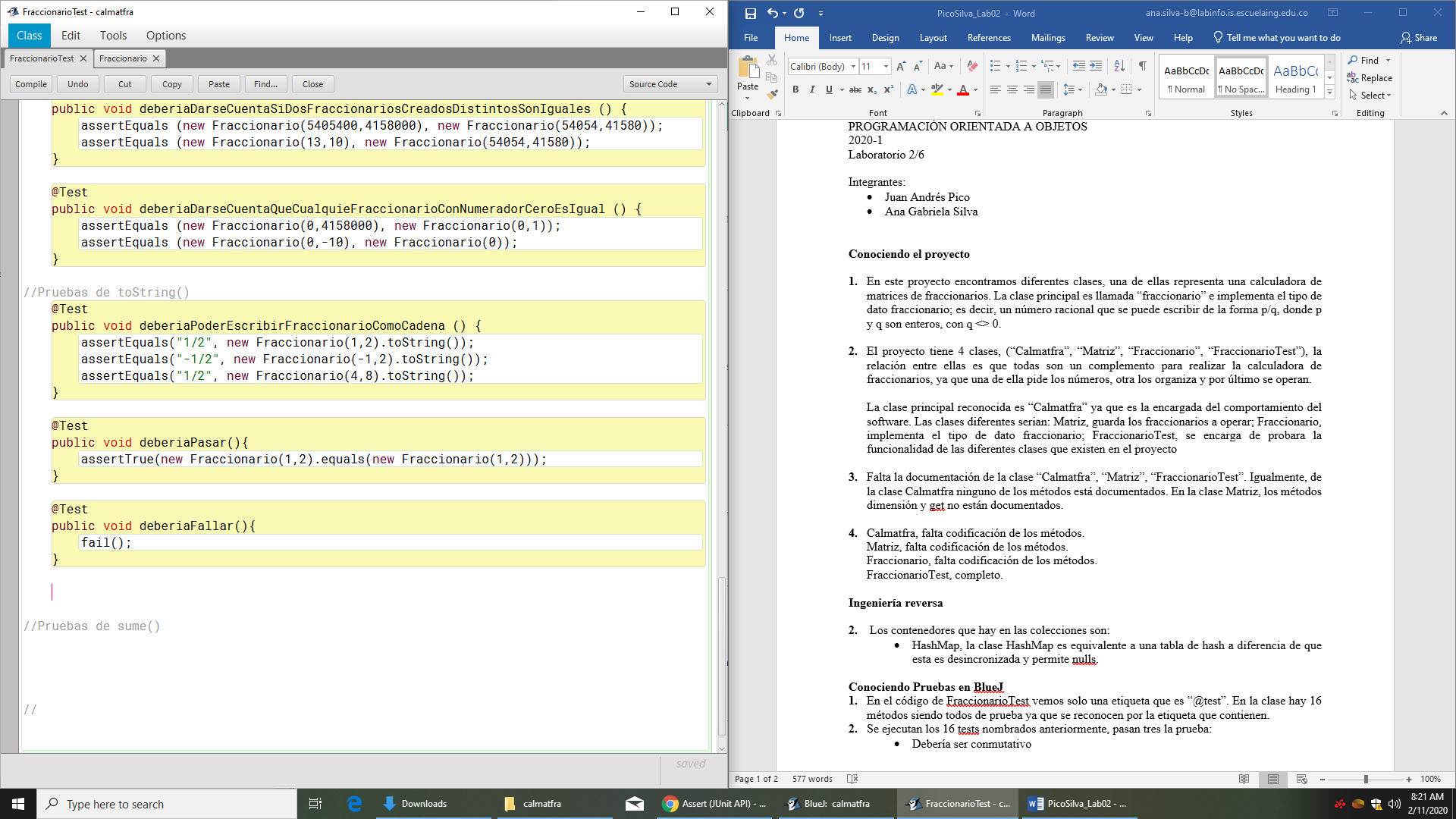
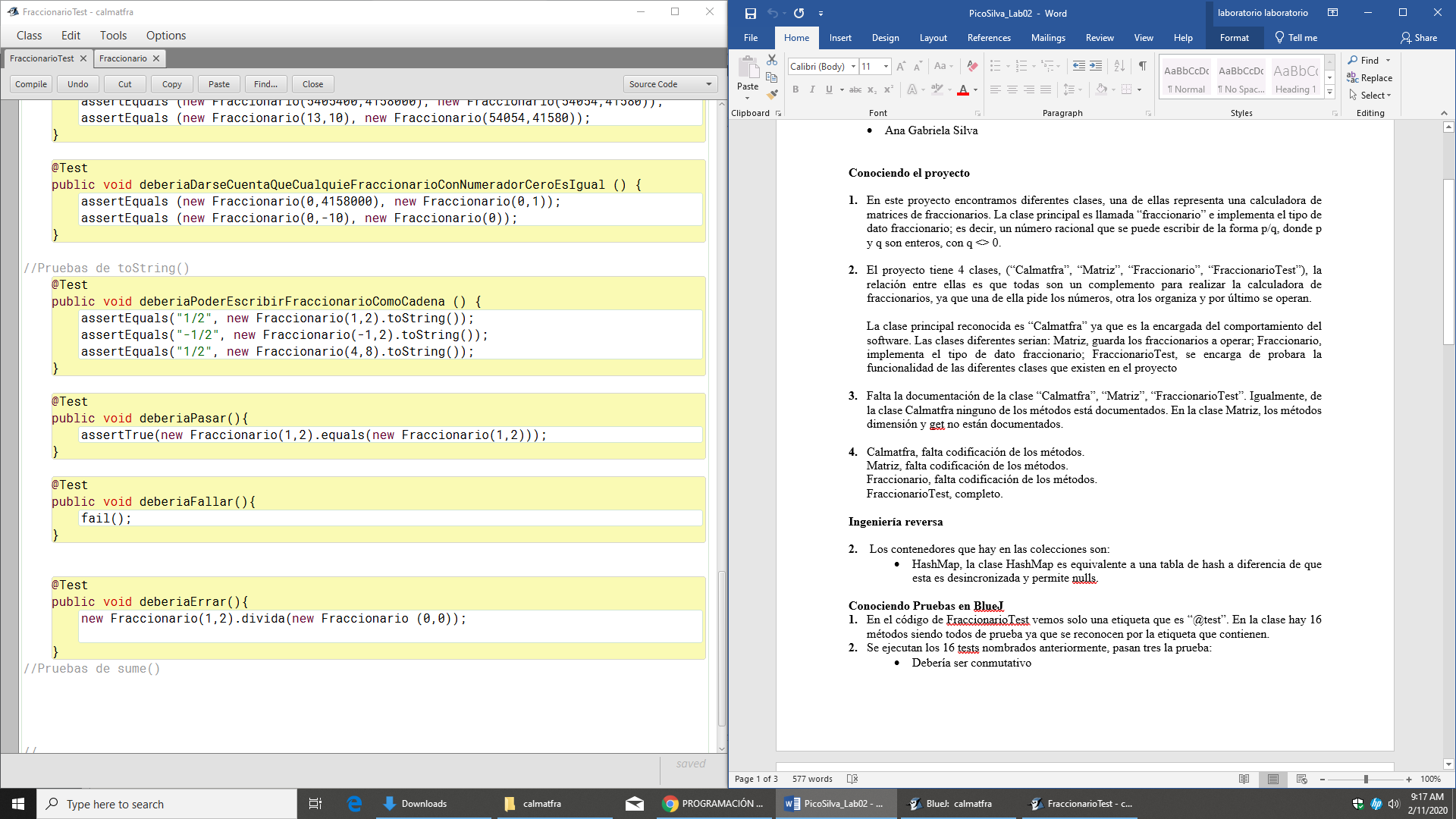
1. En el código de FraccionarioTest vemos solo una etiqueta que es “@test”. En la clase hay 16 métodos siendo todos de prueba ya que se reconocen por la etiqueta que contienen.
2. Se ejecutan los 16 tests nombrados anteriormente, pasan tres la prueba:

* Debería ser conmutativo
* Debería ser el mismo para negativos
* Debería poder crear fraccionarios numerador cero

1. La etiqueta encontrada en el inciso uno, fue “@Test” que da referencia a que el método que se ejecutará será para probar algo puntual del código.
2. **Class Assert**

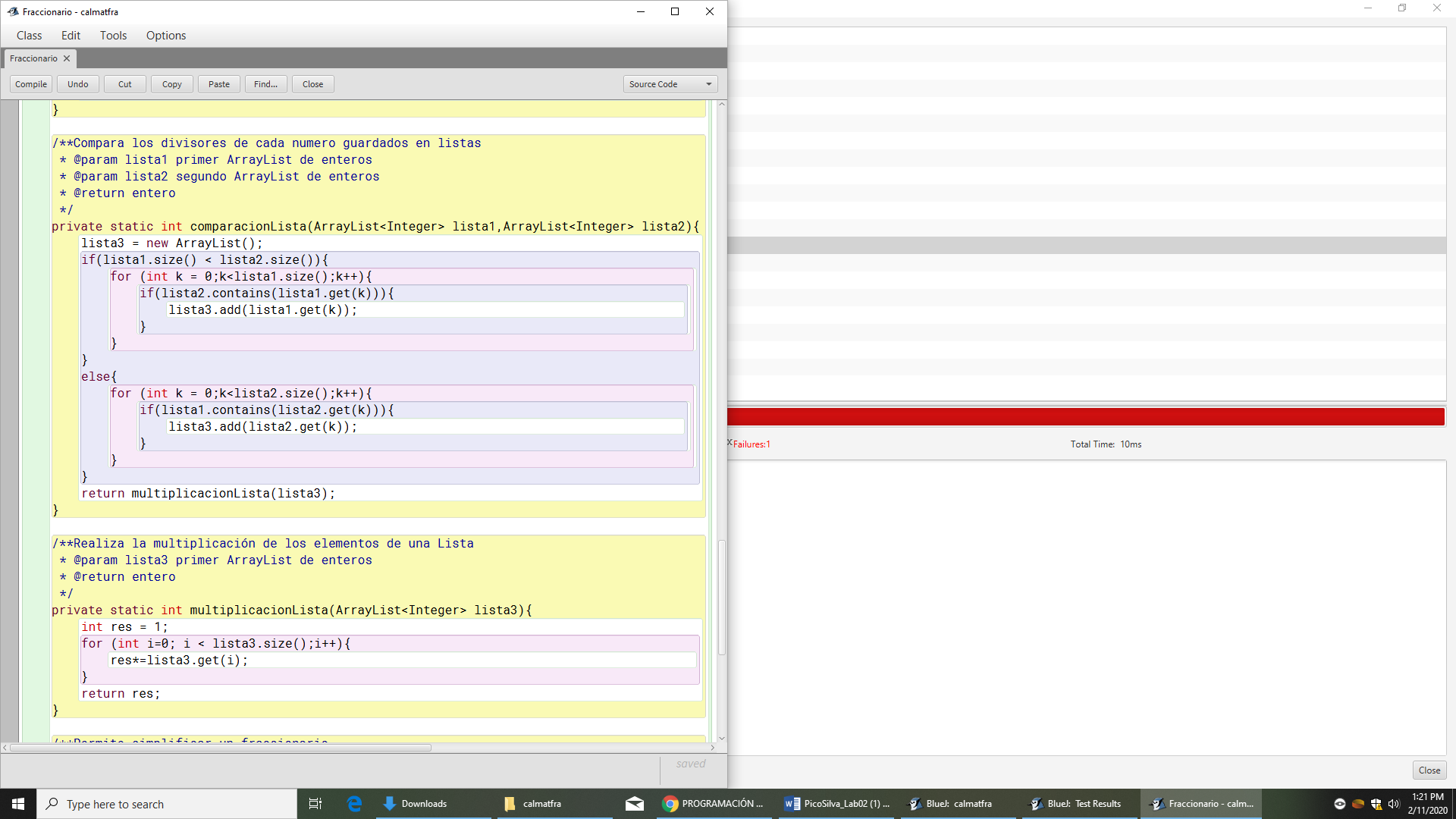
* **assertTrue:** afirma que una condición es verdadera.
* **assertFalse:** afirma que una condición es falsa.
* **assertEquals:** afirma que dos objetos son iguales. Si no lo son, se arroja un “AssertionError” con el mensaje dado. Si lo esperado y lo real son nulos, se consideran iguales.
* **assertNull:** afirma que un objeto no es nulo.
* **fail:** marca una prueba que está incompleta, por lo que falla y te advierte hasta que se pueda terminar.

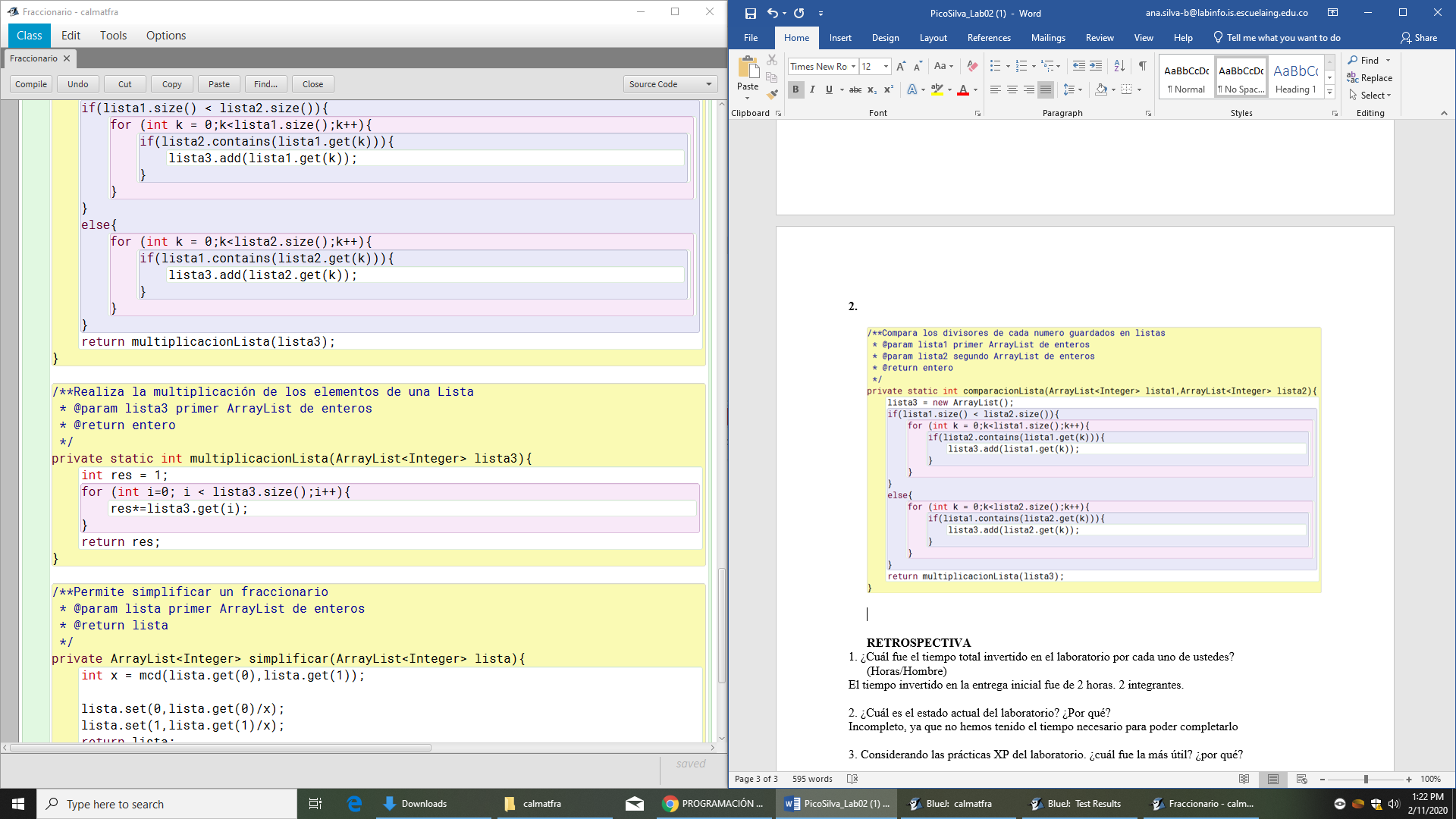
1. La diferencia entre un fallo y error, es que los fallos aparecen si alguna prueba codificada no es cierta, al contrario, un error aparece cuando es un “fallo” en código que hace que no se ejecute de buena manera.

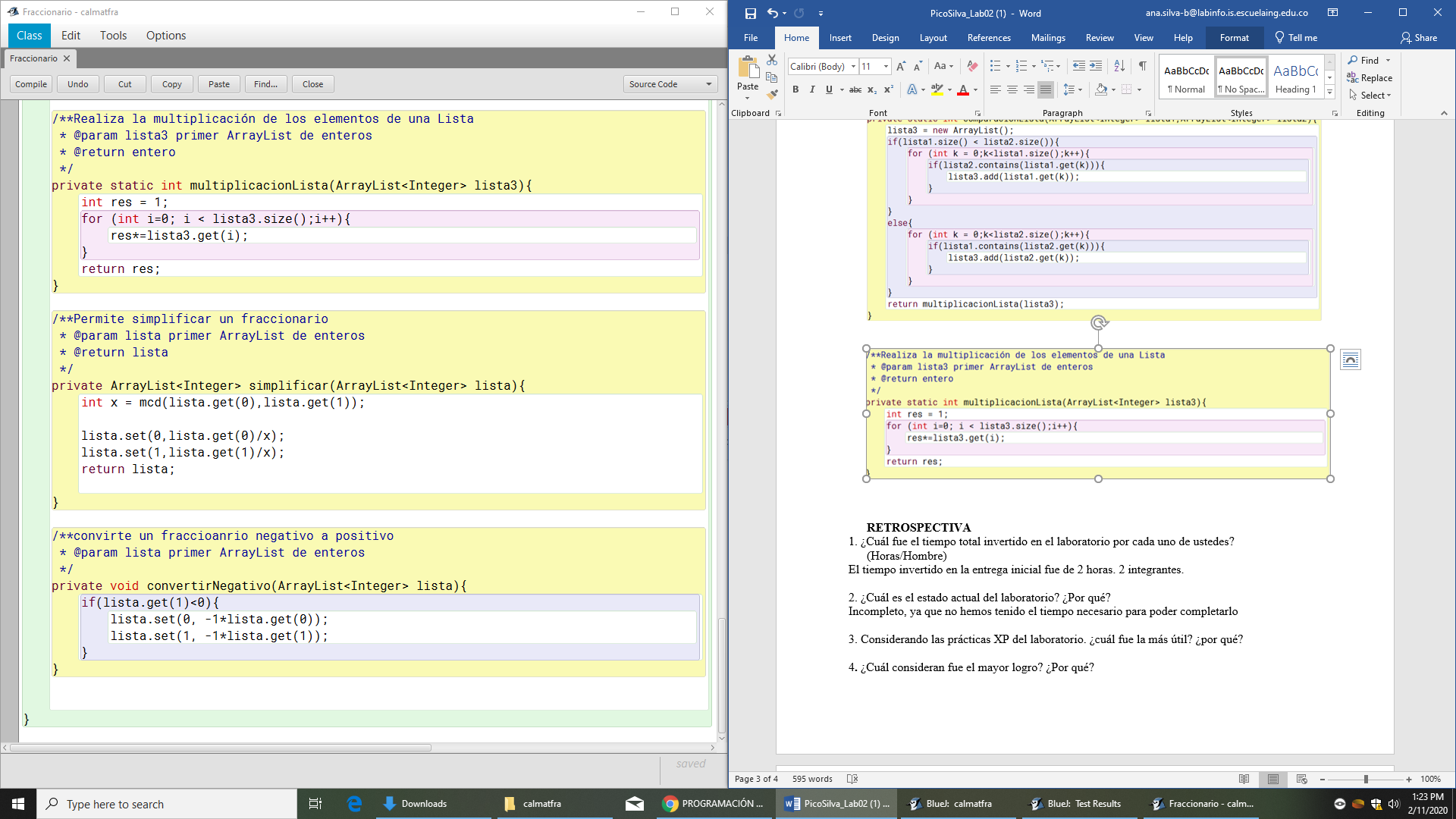


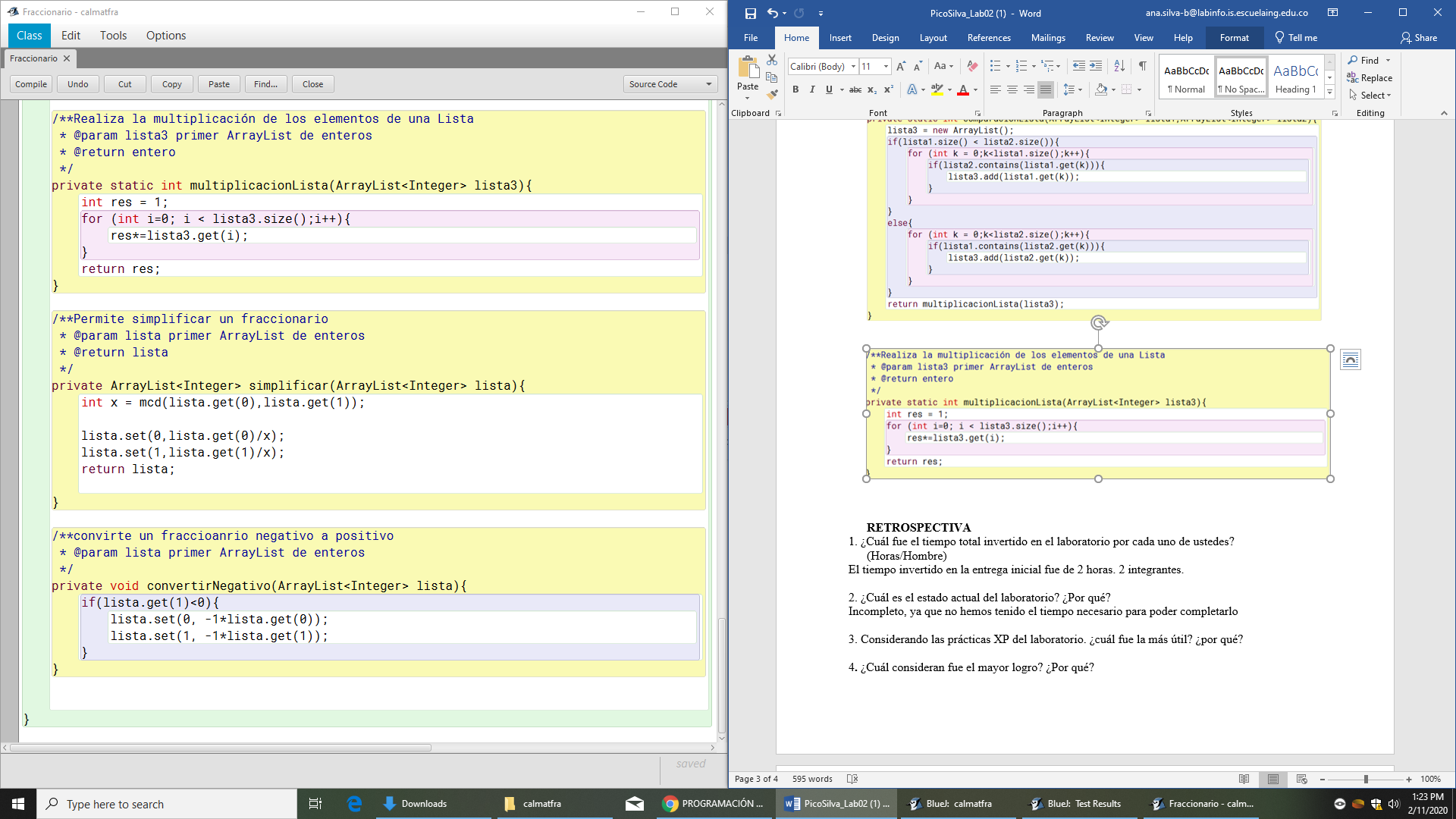
**Practicando Pruebas en BlueJ**

1. Escogemos como estructura de datos un ArrayList, ya que podemos visualizar el denominador y numerador en posiciones.









**Desarrollando**

1. **Ciclo 1 (Asigne y consulte)**

* **Métodos base**

El asigne y consulte se puede realizar con los métodos que nos dan en el blueJ, no se necesitarán más métodos para poder lograr su objetivo.

* **Casos de prueba**

Para asigne, tendremos comprobar que las matrices no tengan posiciones vacías.

Para consulte, que una llave ya ingresada esté en el HashMap.

1. **Ciclo 2 (sume y reste)**

Este ciclo lo alcanzamos a realizar en la clase fraccionario.

* **Métodos base**

Para implementar el método de sume y reste en la clase fraccionario, simplemente se utilizó el constructor de la clase Fraccionario, y los métodos sume () y reste() de esta misma.

* **Casos de prueba**

Para este ciclo los casos de prueba que se tendrán serán los mismo que para FraccionarioTest ya que los objetos que se utilizan son de tipo Fraccionario.

1. **Ciclo 3 (multiplique)**

Este ciclo lo alcanzamos a realizar en la clase fraccionario.

* **Métodos base**

Para implementar el método de sume y reste en la clase fraccionario, simplemente se utilizó el constructor de la clase Fraccionario.

**RETROSPECTIVA**

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes?

(Horas/Hombre)

El tiempo invertido en la entrega fue de 24 horas. 2 integrantes.

2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?

Incompleto, ya que no se alcanzó a realizar los últimos ciclos, con todas las clases. Por ejemplo las operaciones fueron realizadas en la clase fraccionario.

1. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?

La práctica XP más útil es la de All code must have [unit tests](http://www.extremeprogramming.org/rules/unittests.html). Porque creemos que son muy necesarias para poder lograr el objetivo del ejercicio.

4**.** ¿Cuál consideran fue el mayor logro? ¿Por qué?

El mayor logro fue el de poder identificar nuevos términos y así mismo poder aprender a utilizar y codificar pruebas unitarias.

5. Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?

Poder realizar los testing de una nueva clase.

6.¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los

resultados?

Como equipo trabajamos con responsabilidad y compromiso.