Escuela Colombiana de Ingeniería Julio Garavito

Andrés Cardozo

Tulio Riaño

Laboratorio 02

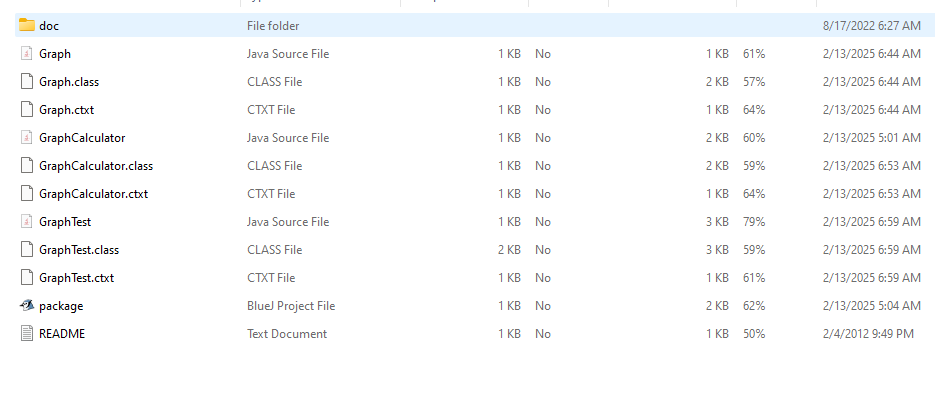
Programacion Orientada a Objetos

FECHA:

14/02/2025

**Conociendo el proyecto**

1. El proyecto “graphCalculator” contiene una construcción parcial del sistema. Revisen el directorio donde se encuentra el proyecto. Describan el contenido en términos de directorios y de las extensiones de los archivos.

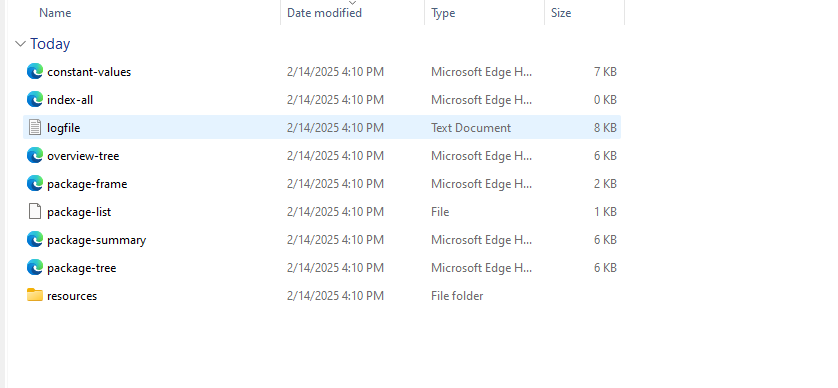


En el directorio donde se encuentra el proyecto tenemos diferente contenidos como:

* Graph.java
* Graph.class
* Graph.ctxt
* GraphCalculator.java
* GraphCalculator.class
* GraphCalculator.ctxt
* GraphTest.java
* GraphTest.class
* GraphTest.ctxt
* Package.bluej
* Readme.txt

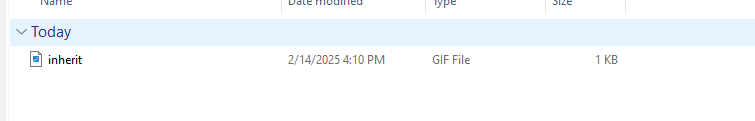
Las extensiones están clasificadas como: .java, .class, .ctxt, .bluej

En la carpeta doc

En este directorio encontramos diferentes archivos que presentan otro tipo de extensiones así como otro directorio como es resources.

* Constant-values.html
* Index-all.html
* Logfile.txt
* Overview-tree.html
* Package-frame.html
* Package-summary.html
* Package-tree.html

Por último, tenemos el directorio resources que contiene:

Una imagen con extensión .gif

2. Exploren el proyecto en BlueJ

¿Cuántas clases tiene? ¿Cuál es la relación entre ellas?

Tiene 3 clases que son: graphCalculator, Graph y graphTest.

graphTest y graphCalculator utilizan la clase de Graph.

¿Cuál es la clase principal de la aplicación? ¿Cómo la reconocen?

La clase principal es Graph, la reconocemos porque identificamos que va a ser la base para realizar la calculadora correspondiente.

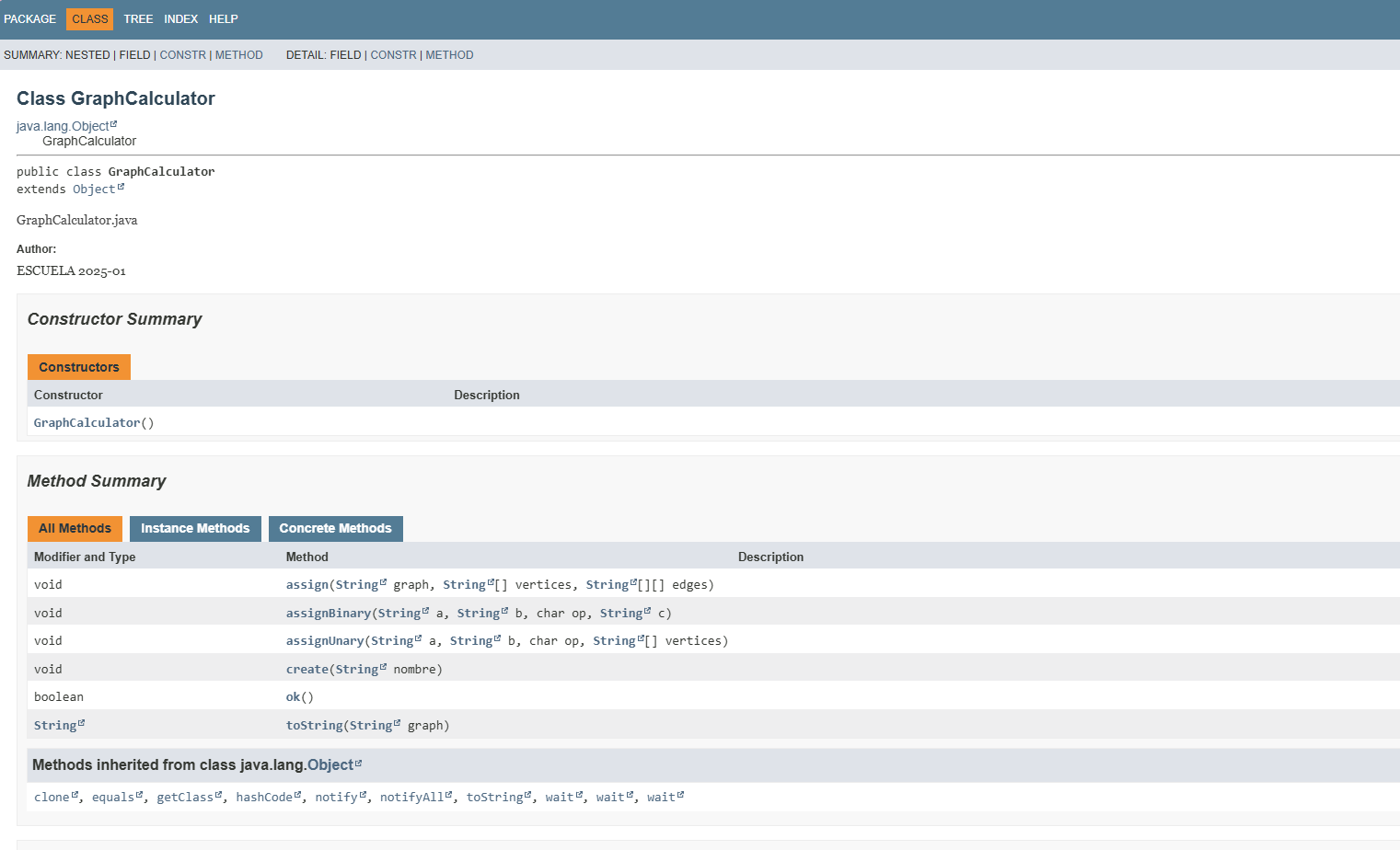
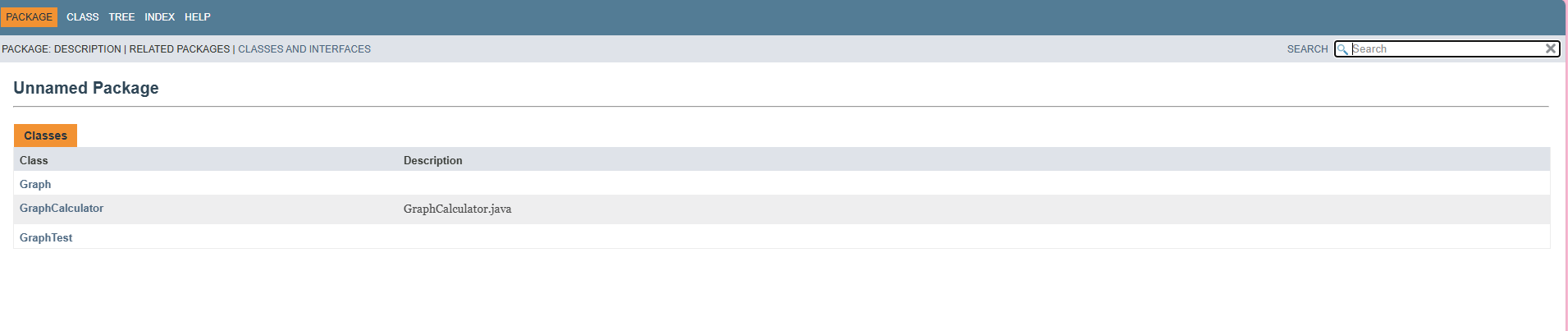
¿Cuáles son las clases “diferentes”? ¿Cuál es su propósito?

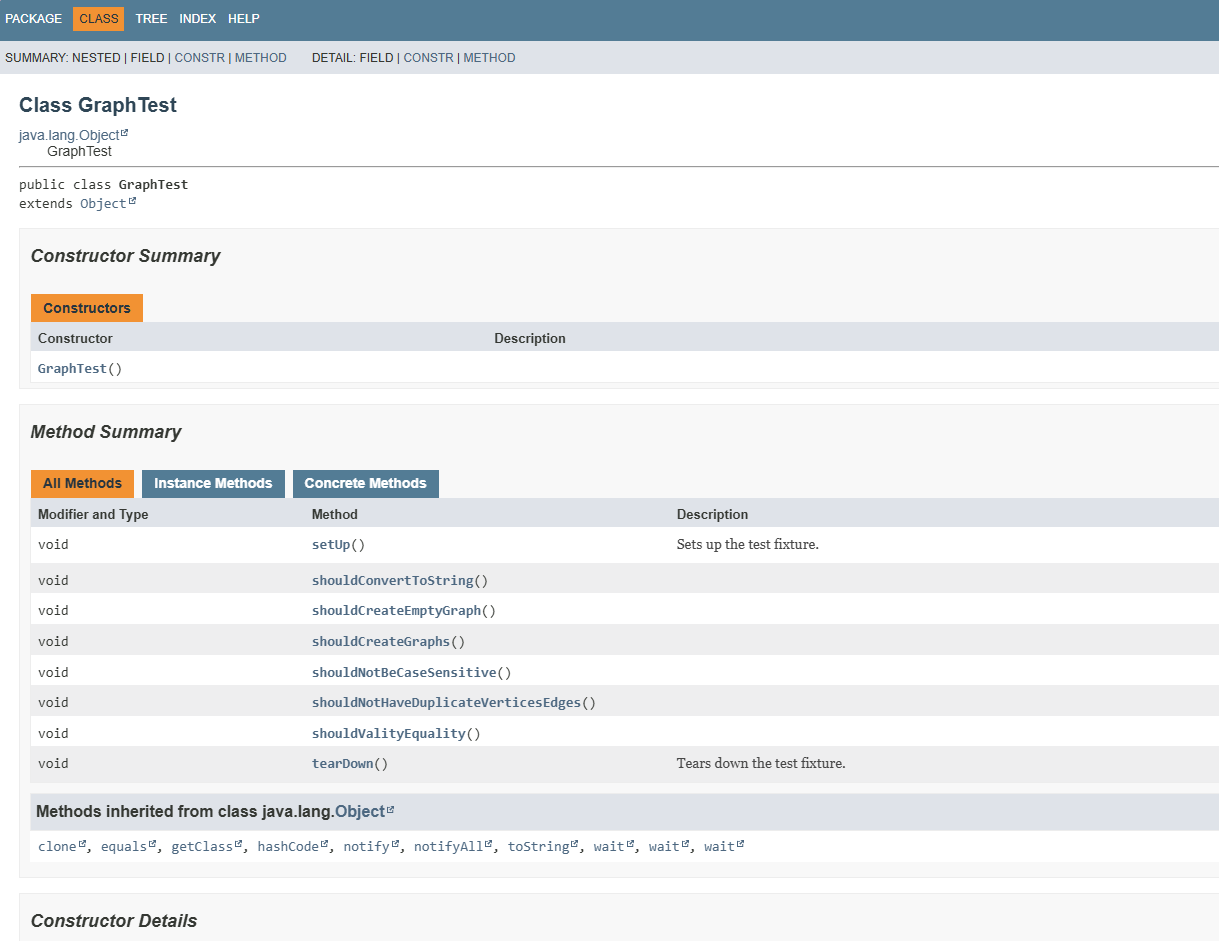
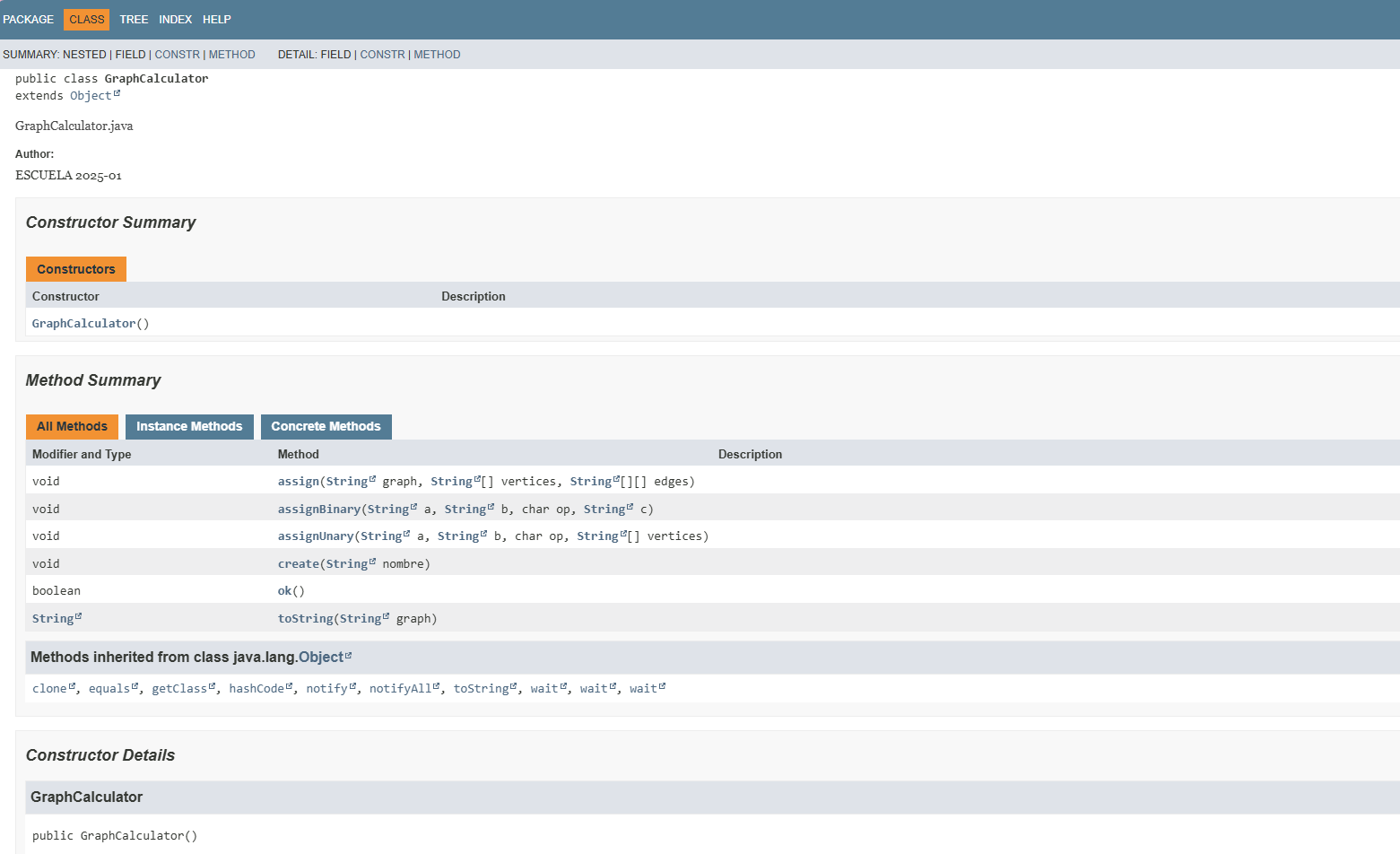
La clase graphTest es visualmente diferente ya que su proposito es proporcionar testeos y pruebas.

Para las siguientes dos preguntas sólo consideren las clases “normales”:

3. Generen y revisen la documentación del proyecto: ¿está completa la documentación de cada clase? (Detallen el estado de documentación: encabezado y métodos)

La documentación está incompleta, las clases no tienen descripción, los métodos de cada una de las clases solamente presentan su encabezado, pero no tienen documentación propia ni los atributos, así como ni descripción de que hace cada método asociado.





4. Revisen las fuentes del proyecto, ¿en qué estado está cada clase? (Detallen el estado de las fuentes considerando dos dimensiones: la primera, atributos y métodos, y la segunda, código, documentación y comentarios)

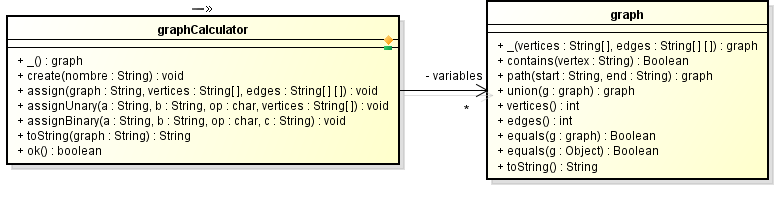
¿Qué diferencia hay entre el código, la documentación y los comentarios?

Las clases están en un estado incompleto, por ejemplo en la clase graph: no tienen descripción y hay pocos comentarios, los métodos de cada una de las clases solamente presentan su encabezado, pero no tienen documentación ni metodos completos, la clase tampoco tiene atributos.

Por otra parte, en graphCalculator podemos evidenciar una mayor cantidad de comentarios, la definición de una variable como es privada en este caso, pero los métodos están incompletos de igual manera falta una documentación detallada.

**Ingeniería reversa**

1. Realicen el diagrama de clases correspondiente al proyecto. (No incluyan la clase de pruebas)



2. ¿Cuáles contenedores están definidos? ¿Qué diferencias hay entre el nuevo contenedor, el ArrayList y el vector [] que conocemos? Consulte el API de java.

En el proyecto estan definidos: TreeMap, y vector []

ArrayList y Vector implementan la interfaz java.util.List y brindan la capacidad de almacenar y obtener objetos mediante métodos API simples. Los Vectores son una colección sincronizada y ArrayList no lo es ( Que sea sincronizada representa que son seguras para usar en entornos multihilo. Esto implica que, aunque múltiples hilos puedan acceder al Vector simultáneamente las operaciones sobre él se realizan de manera controlada). Debido a esto se considera que el Vector tiene mejor rendimiento.

Cuando un vector necesita aumentar su capacidad para agregar un elemento, el vector aumenta la capacidad en un 100 % durante el cambio de tamaño, mientras que la capacidad de ArrayList aumenta en un 50% de la capacidad existente.

Por otro lado, TreeMap es una implementación de la interfaz Map, que representa una colección de pares clave-valor, parecida a un arbol binario de busqueda. Una diferencia clave es que ArrayList y el Vector son colecciones indexadas, mientras que Map no se basa en índices, sino en claves únicas para acceder a los valores. Debido a esto tienen diferencias como en un ordenamiento diferente, como tambien que en el TreeMap no se permiten duplicados.

(referencia parafraseo de 1,2,3)

3. En el nuevo contenedor, ¿Cómo adicionamos un elemento? ¿Cómo lo consultamos? ¿Cómo lo eliminamos?

Para añadir un elemento usamos .put(clave , valor)

Para consultar se usa .get(clave) la cual devuelve el valor de dicha clase

Y para eliminar se usa .remove(clave) la cual va a eliminar el valor asociado a dicha clave.

(referencia 3)

**Conociendo Pruebas en BlueJ**

Para poder cumplir con la prácticas XP vamos a aprender a realizar las pruebas de unidad usando las herramientas apropiadas. Para eso implementaremos algunos métodos en la clase GraphTest

1. Revisen el código de la clase GraphTest. ¿cuáles etiquetas tiene (componentes con símbolo @)? ¿cuántos métodos tiene?

@before : setUp() --> 1 metodo

@test : shouldCreateEmptyGraph(), shouldCreateGraphs(), shouldNotHaveDuplicateVerticesEdges(), shouldNotBeCaseSensitive(), shouldConvertToString(), shouldValityEquality() --> 6 metodos

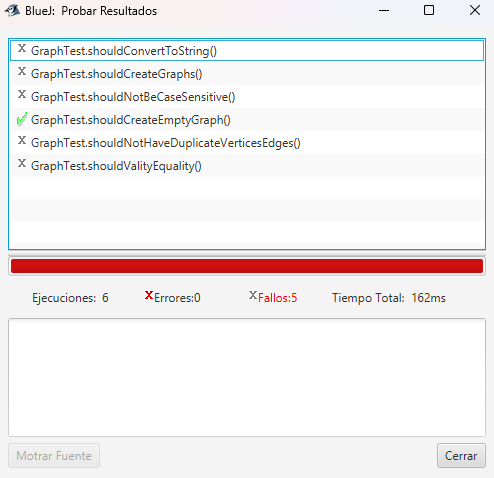
@After : tearDown(); --> 1 metodo

¿cuantos métodos son de prueba? ¿cómo los reconocen?

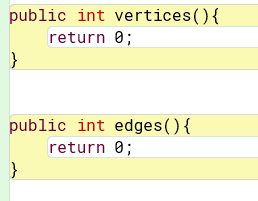
6 metodos son de prueba, esto mediante la etiqueta de test

2. Ejecuten los tests de la clase GraphTest. (click derecho sobre la clase, Test All)

¿cuántas pruebas se ejecutan? ¿cuántas pasan? ¿por qué? Capturen la pantalla.



Se ejecutan las 6 pruebas, pero solo pasa 1 prueba pasa correctamente. Los casos de prueba que estan fallando es debido a la manera en la que se esta utilizando AssertEquals este devolverá verdadero si y solo sí los dos valores son iguales ( esperado == real), lo que ocurre en el caso de shouldCreateEmptyGraph(), en los demas casos no ocurre nada ya que en la clase graph esta definido de la siguiente manera:



3. Estudie las etiquetas encontradas en 1 (marcadas con @). Expliqen en sus palabras su significado.

Las etiquetas son parte de las pruebas JUnit, estas verifican un estado antes y despues de las pruebas, por ejemplo para limpiar el entorno despues de cada caso de prueba

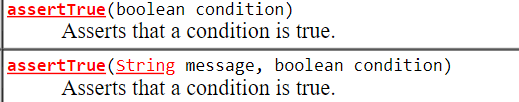
@before – antes de cada prueba, se usa setUp y configura el dispositivo de prueba.

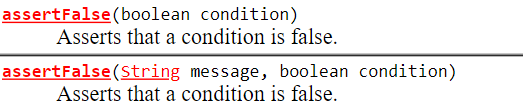
@test -- ya son los metodos que pertenecen a las pruebas que queremos realizar

@After -- despues de cada prueba, se usa tearDown y derriba el dispositivo de prueba.

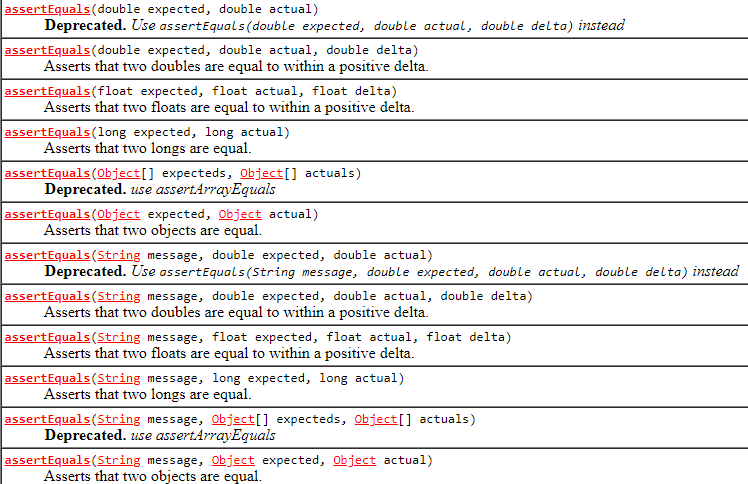
4. Estudie los métodos assertTrue, assertFalse, assertEquals, assertNull y fail de

la clase Assert del API JUnit. Explique en sus palabras que hace cada uno de ellos.

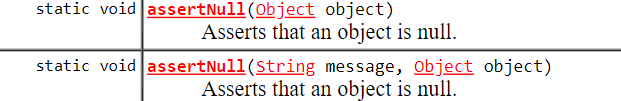
AssertTrue determina si una condición es verdadera, si esta es falsa, la prueba fallara.

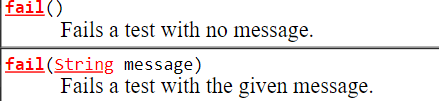


AssertFalse determina si una condicion es falsa, si esta es verdadera, la prueba fallara.



AssertEquials determina si dos valores, uno esperado y otro la prueba que es el real, son iguales, si es falso, la prueba fallara.

AssertNull verifica si un objeto es nullo, si no es nulo, la prueba fallara.



Fail provoca que inmediatamente la prueba falle.

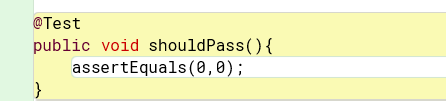
(Referenciado 5)

5. Investiguen y expliquen la diferencia entre un fallo y un error en Junit. Escriba código, usando los métodos del punto 4, para codificar los siguientes tres casos de prueba y lograr que se comporten como lo prometen shouldPass, shouldFail, shouldErr.

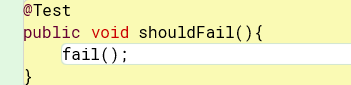
“Los fallos son when your test cases fail– es decir, sus afirmaciones son incorrectas. Los errores se producen when unexpected errors/exceptions occur, por ejemplo, cuando se intenta ejecutar la prueba y se lanza una excepción inesperada como FileNotFound, etc.” (6)

En resumidas cuentas, un fallo va más de la mano de las propias pruebas que realizamos con Junit, mientras que los errores son excepciones o errores inesperados tales como división por 0, o índices fuera de rango, etc.

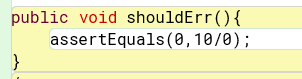
**Should Pass**

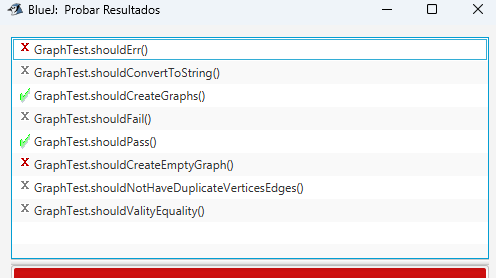


**Should Fail**



**Should Err**



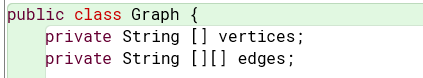


**Prácticando Pruebas en BlueJ**

Ahora vamos escribir el código necesario para que las pruebas de GraphTest pasen.

1. Determinen los atributos de la clase Graph. Justifique la selección.

Para los atributos de la clase Graph definimos dos específicamente:



2. Determinen el invariante de la clase Graph. Justifique la decisión.

- No pueden haber vertices ni aristas duplicadas

- Una arista es una pareja ordenada (no puede haber mas de un elemento)

- La arista referencie elementos del grafo que existan

3. Implementen los métodos de Graph necesarios para pasar todas las pruebas definidas. ¿Cuáles métodos implementaron?

Hicimos los siguientes metodos:

- removeDuplicates() para quitar elementos repetidos como aristas y vertices

- upperCase() para pasar todo String a mayusculas

- reorganizeEdges() y reorganizeVertexGraph() para organizar alfabeticamente los vertices y aristas

- verifyVertexExist() para verificar si una arista referencia vertices que existen

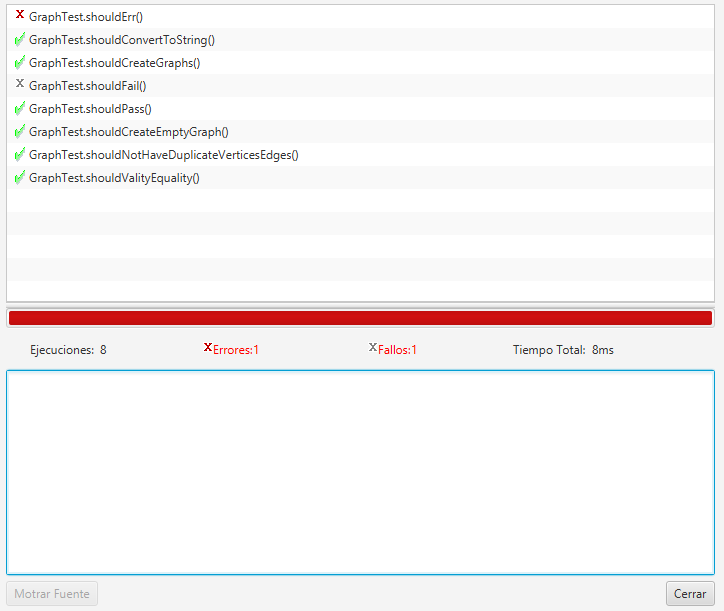
- toString() para obtener la cadena de texto en vez de la referencia del objeto para devolver las aristas con las especificaciones

- equals(Graph g) para verificar si dos grafos son iguales

- vertices() y edges() para devolver el valor entero de la cantidad de vertices y aristas

- contains(String vertex) verifica si pertenece un vertice al grafo

4. Capturen los resultados de las pruebas de unidad.



**Desarrollando GraphCalculator**

Para desarrollar esta aplicación vamos a considerar algunos ciclos. En cada ciclo deben realizar los pasos definidos a continuación.

1. Definir los métodos base de correspondientes al mini-ciclo actual.

2. Definir y programar los casos de prueba de esos métodos

Piensen en los deberia y los noDeberia (should and shouldNot)

3. Diseñar los métodos

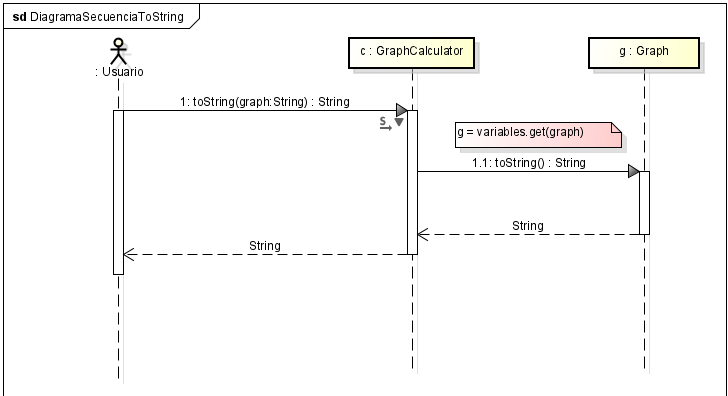
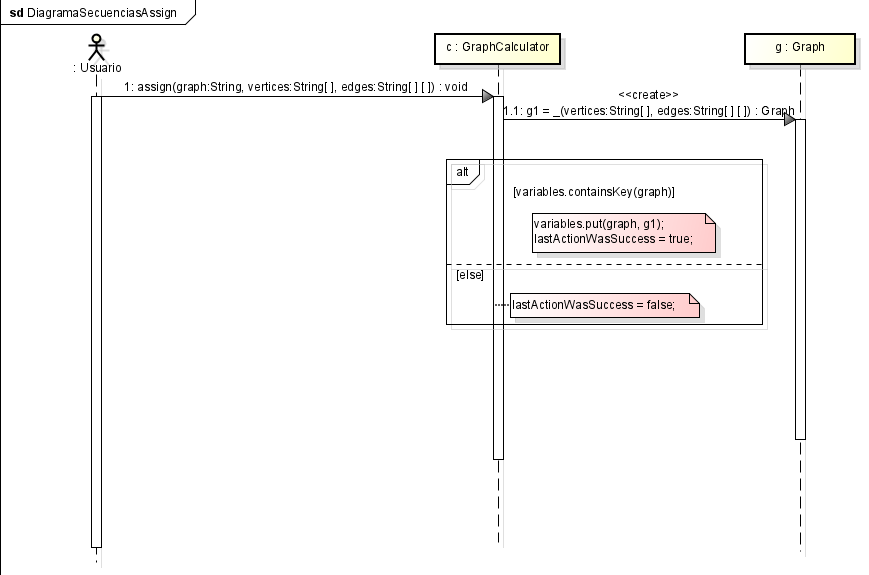
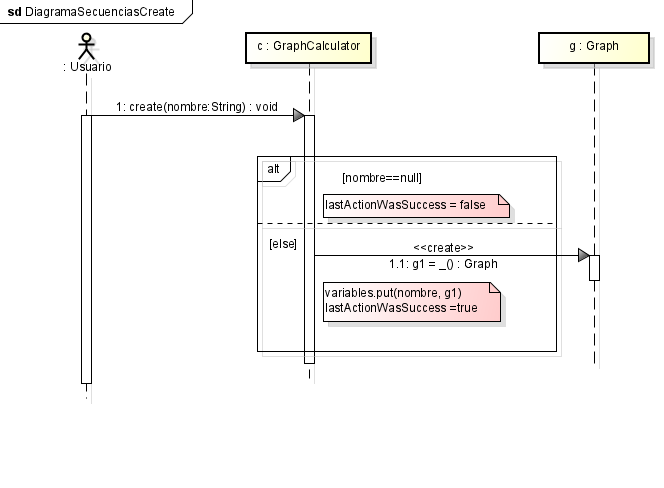
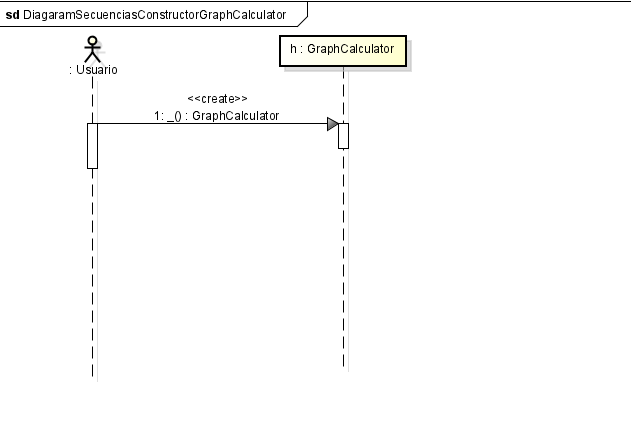
Usen diagramas de secuencia. En astah, creen el diagrama sobre el método correspondiente.

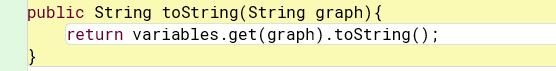
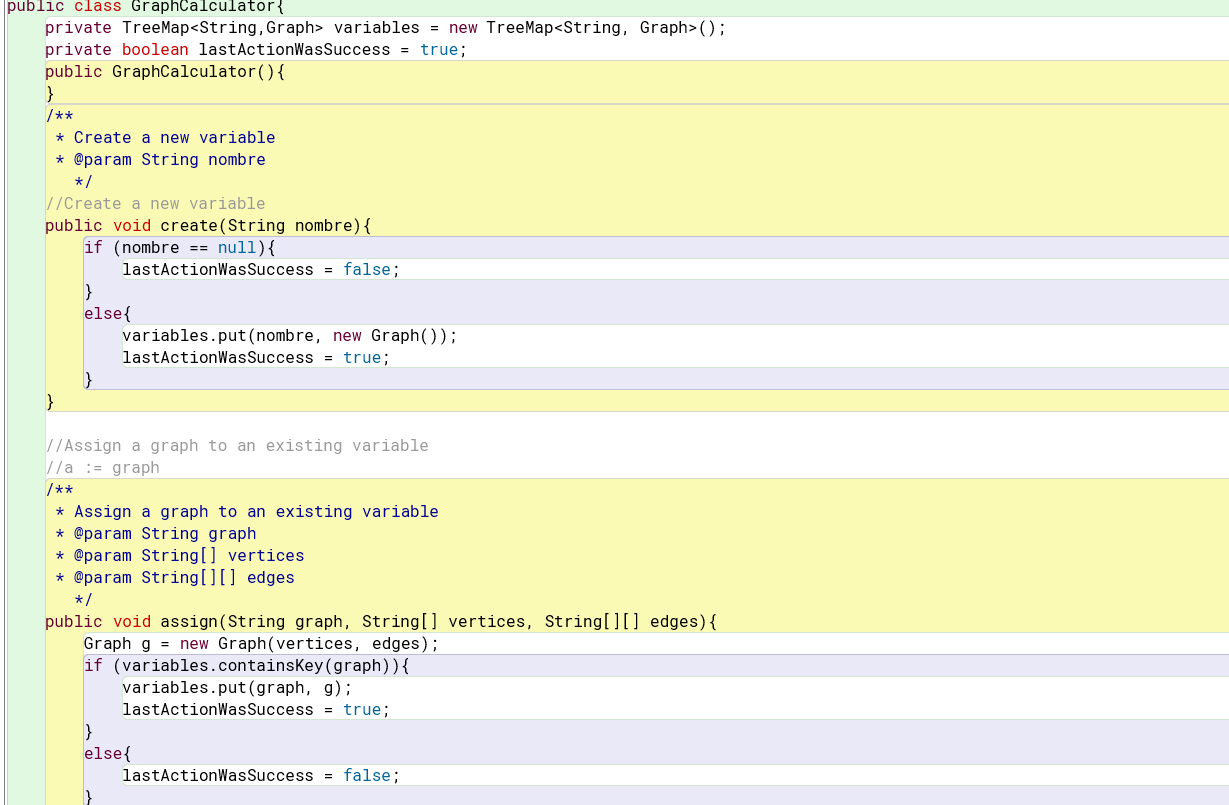
4. Escribir el código correspondiente (no olvide la documentación)

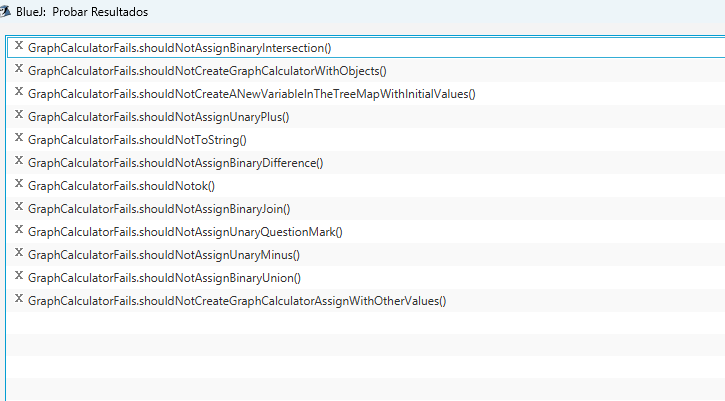
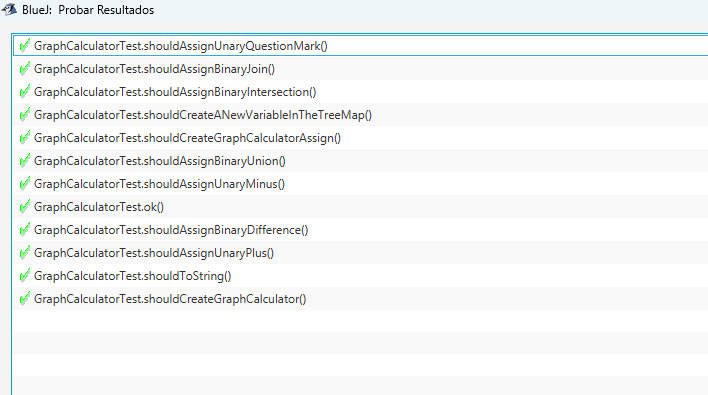
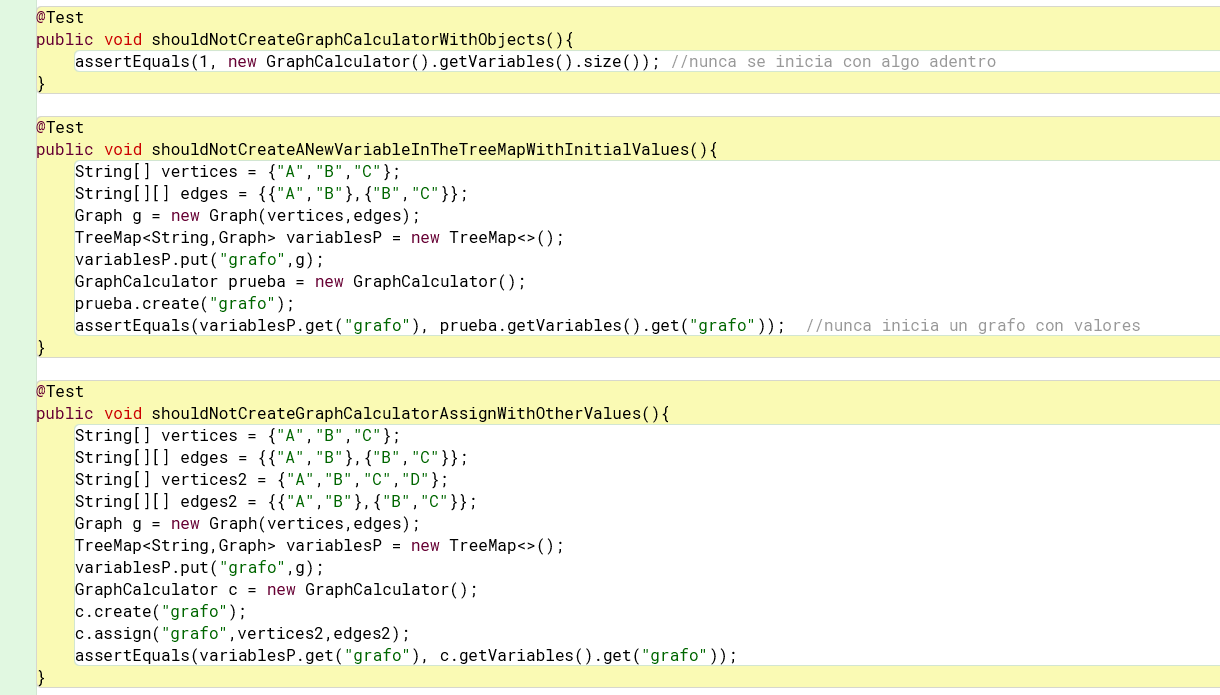
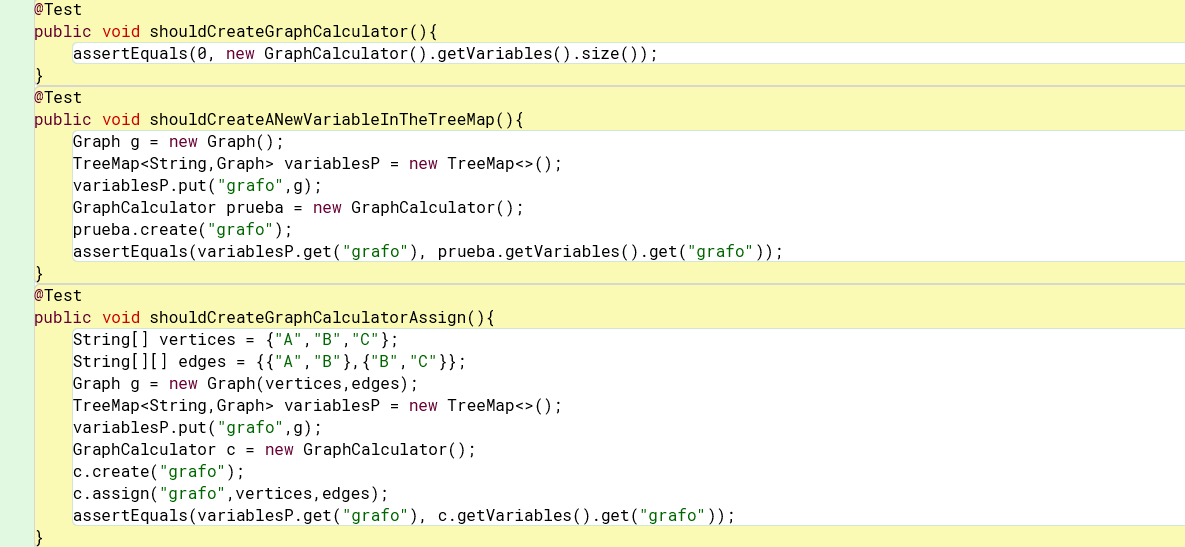
5. Ejecutar las pruebas de unidad (vuelva a 3 (a veces a 2), si no están en verde)

6. Completar la tabla de clases y métodos. (Al final del documento)

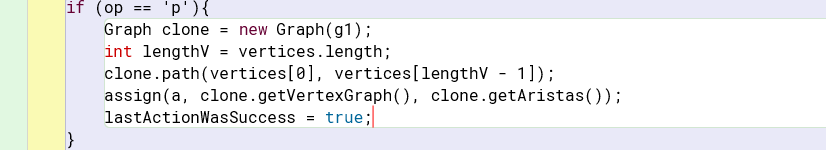
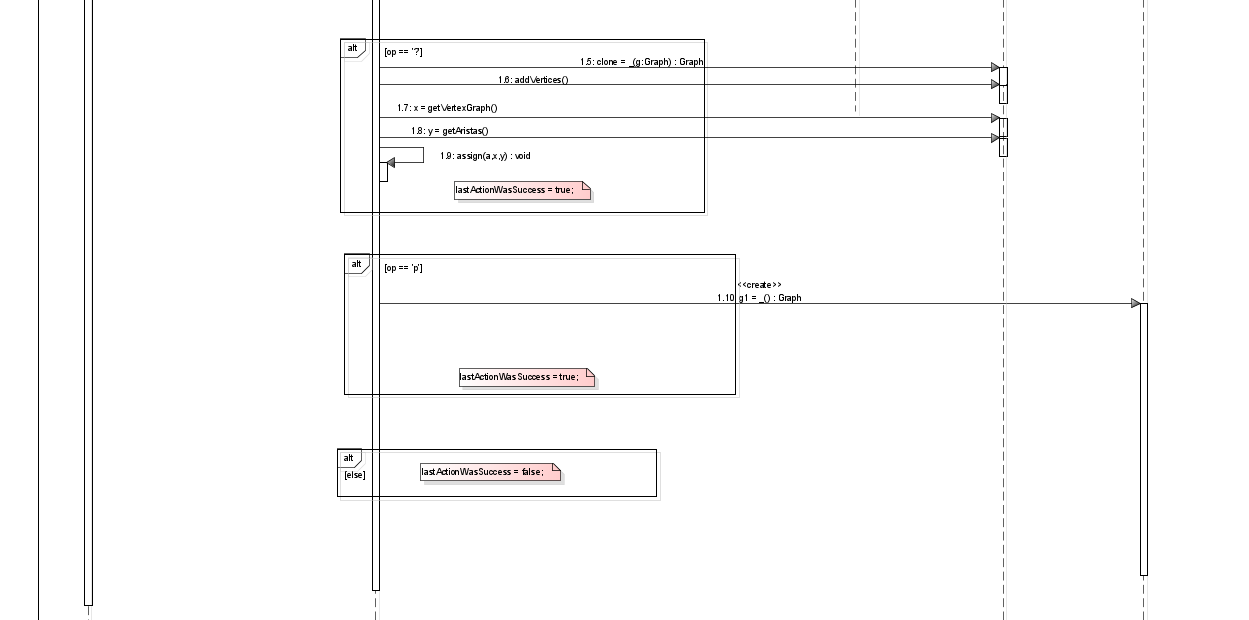
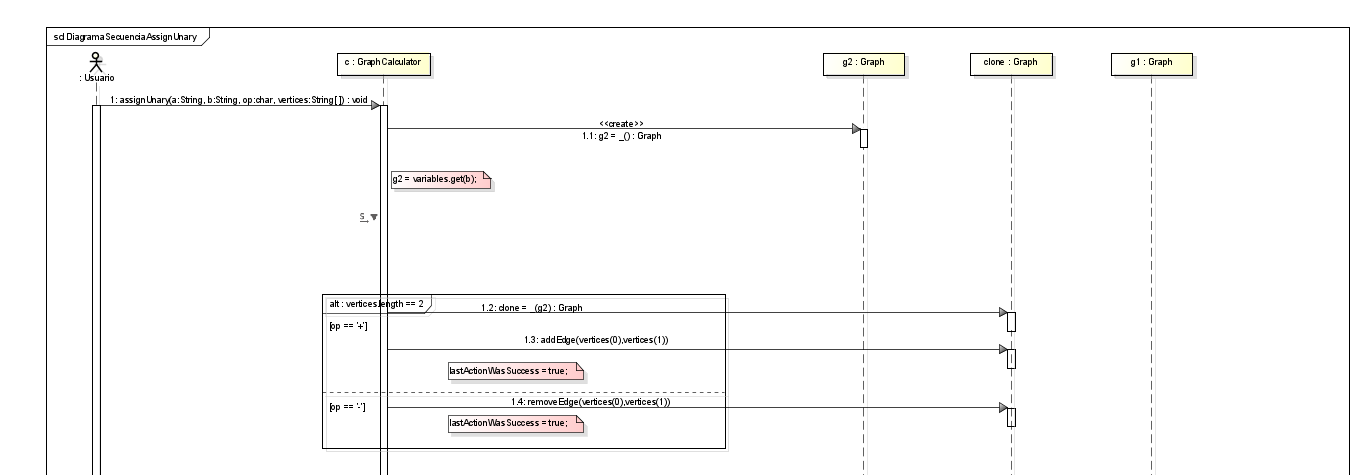
Ciclo 1 : Operaciones básicas de la calculadora: crear una calculadora y asignar y consultar un grafo.

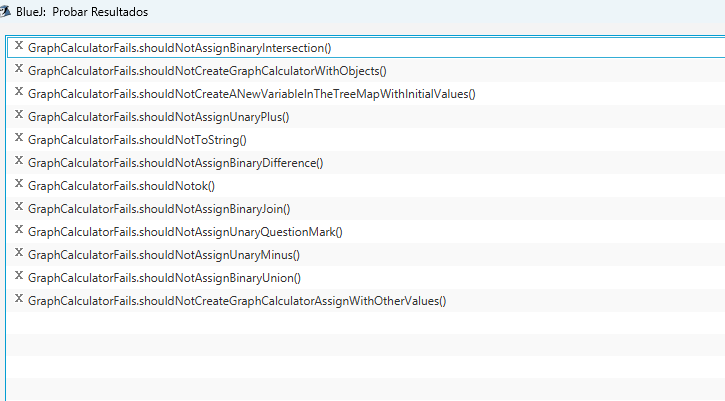
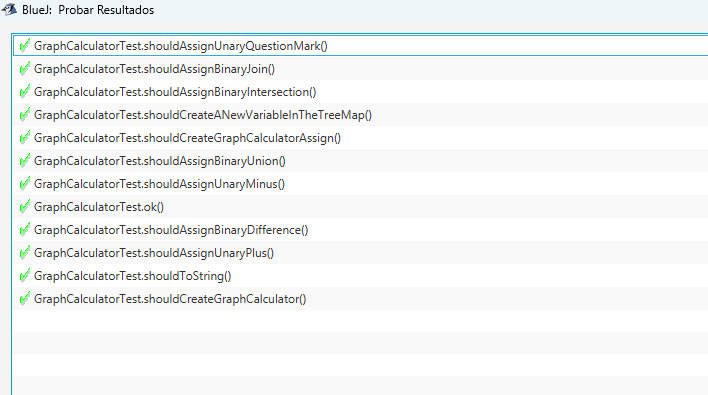
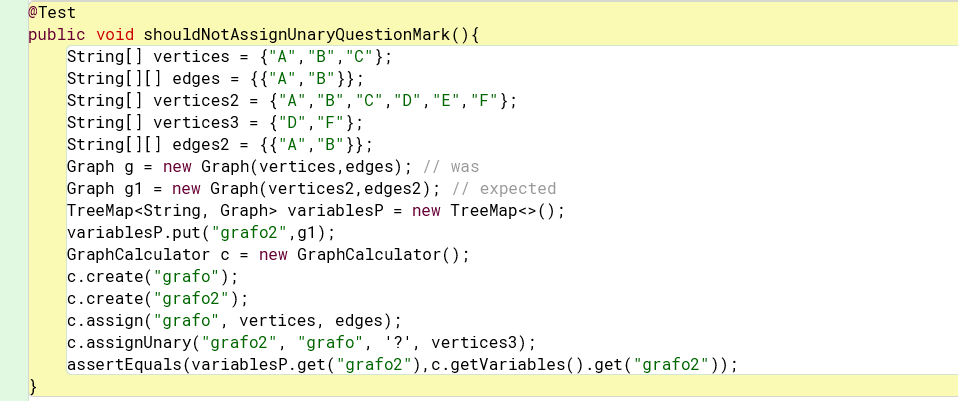
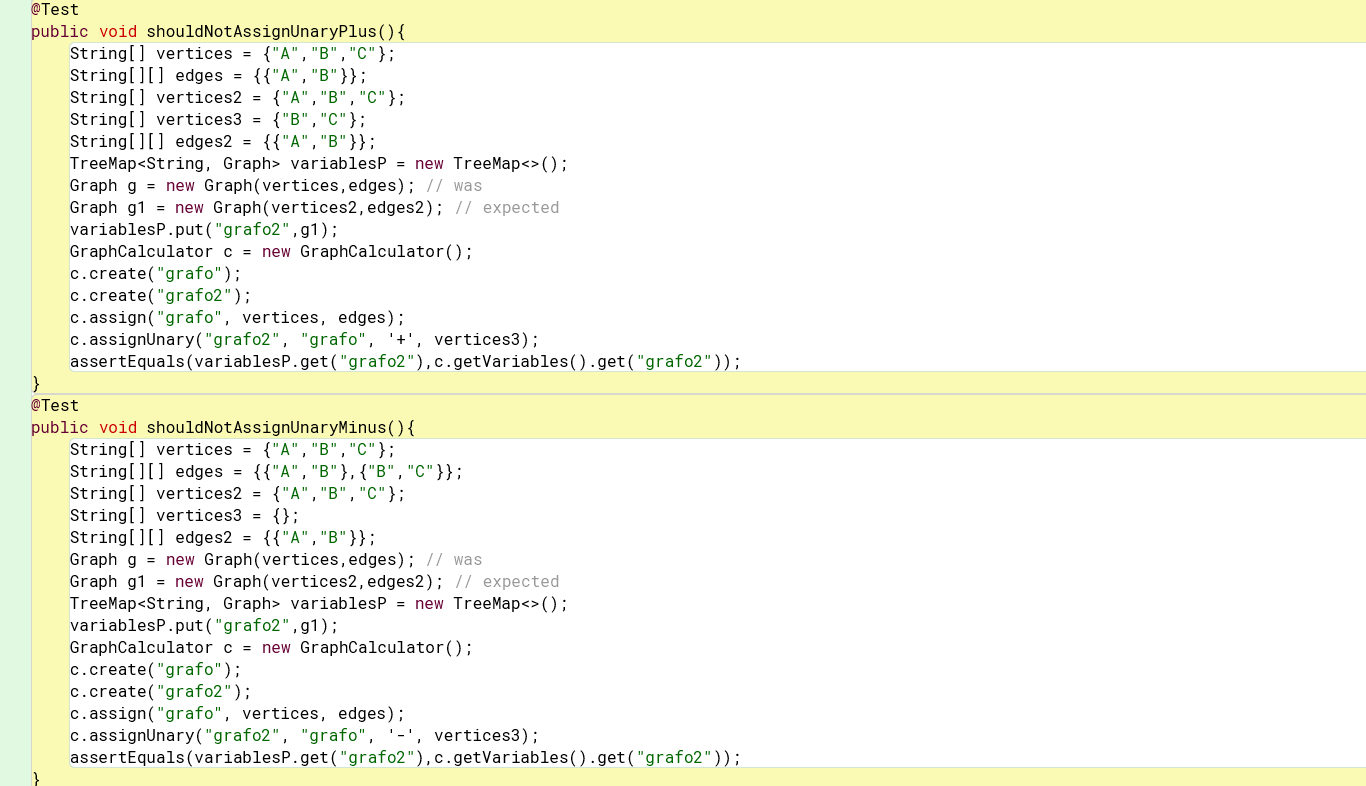
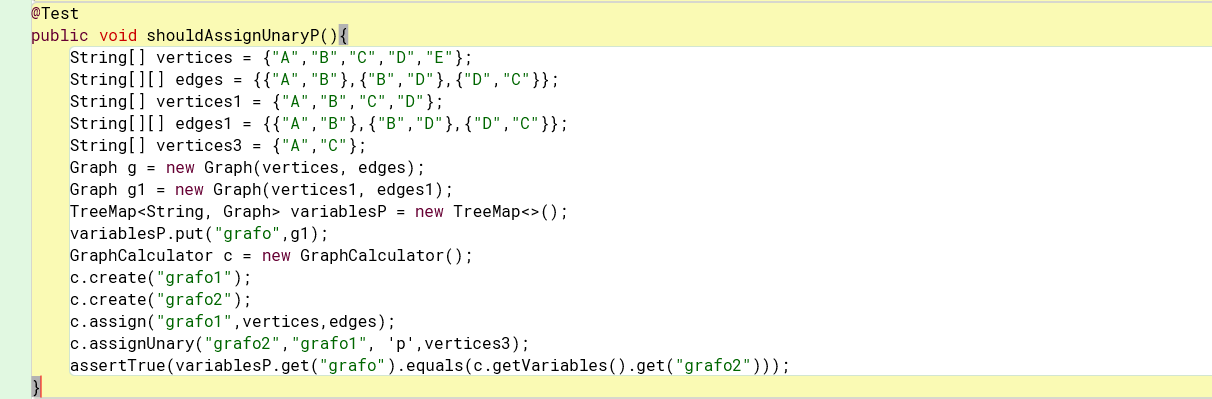
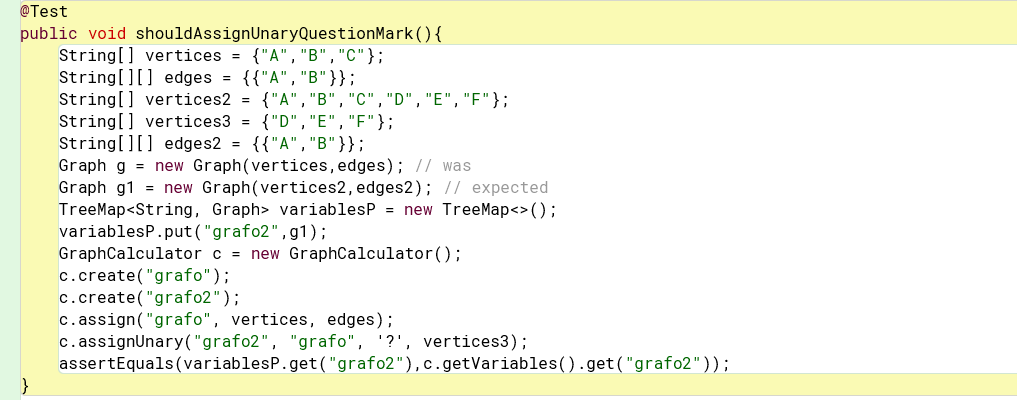
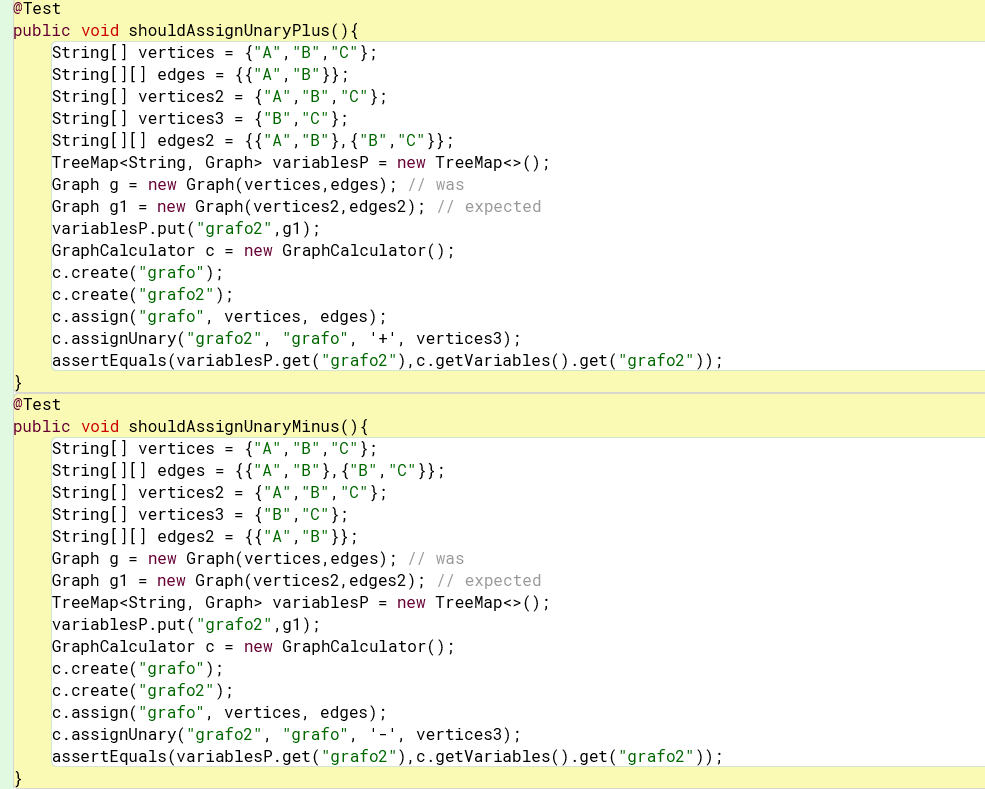
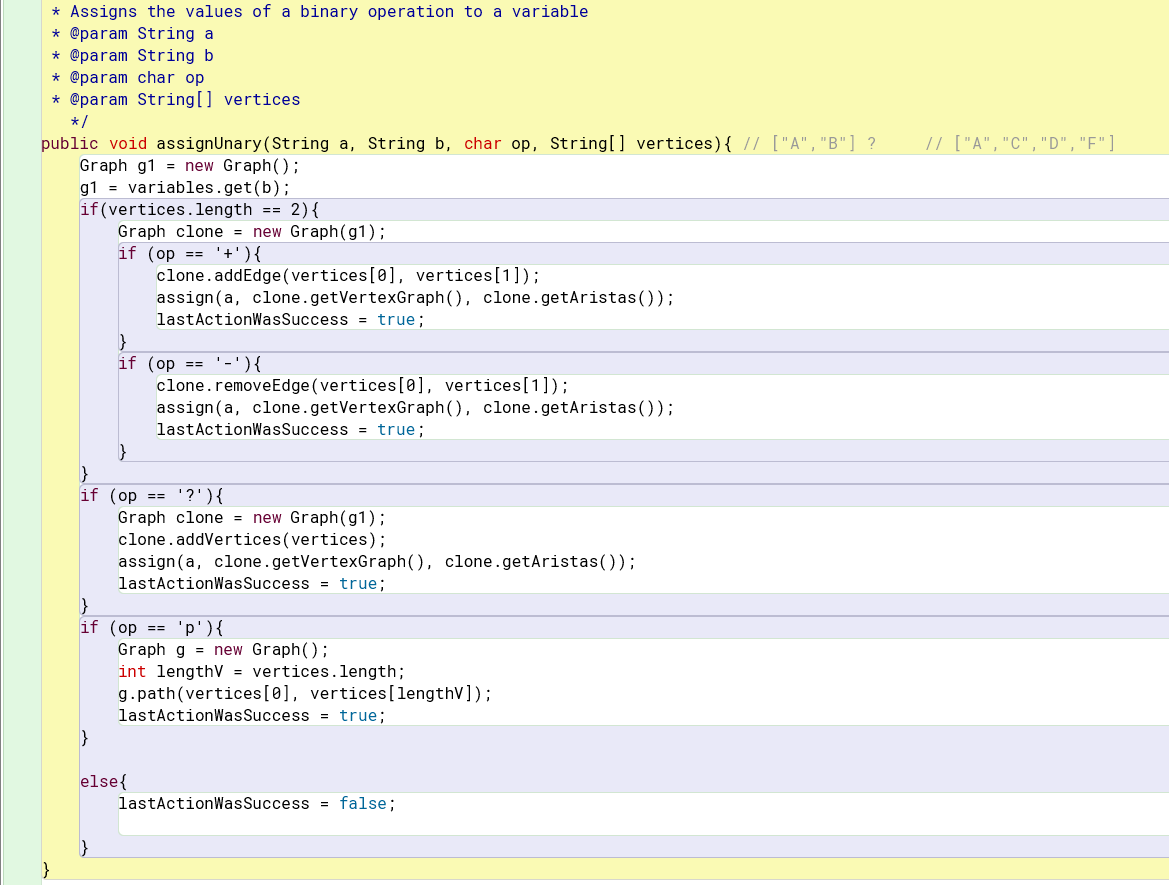




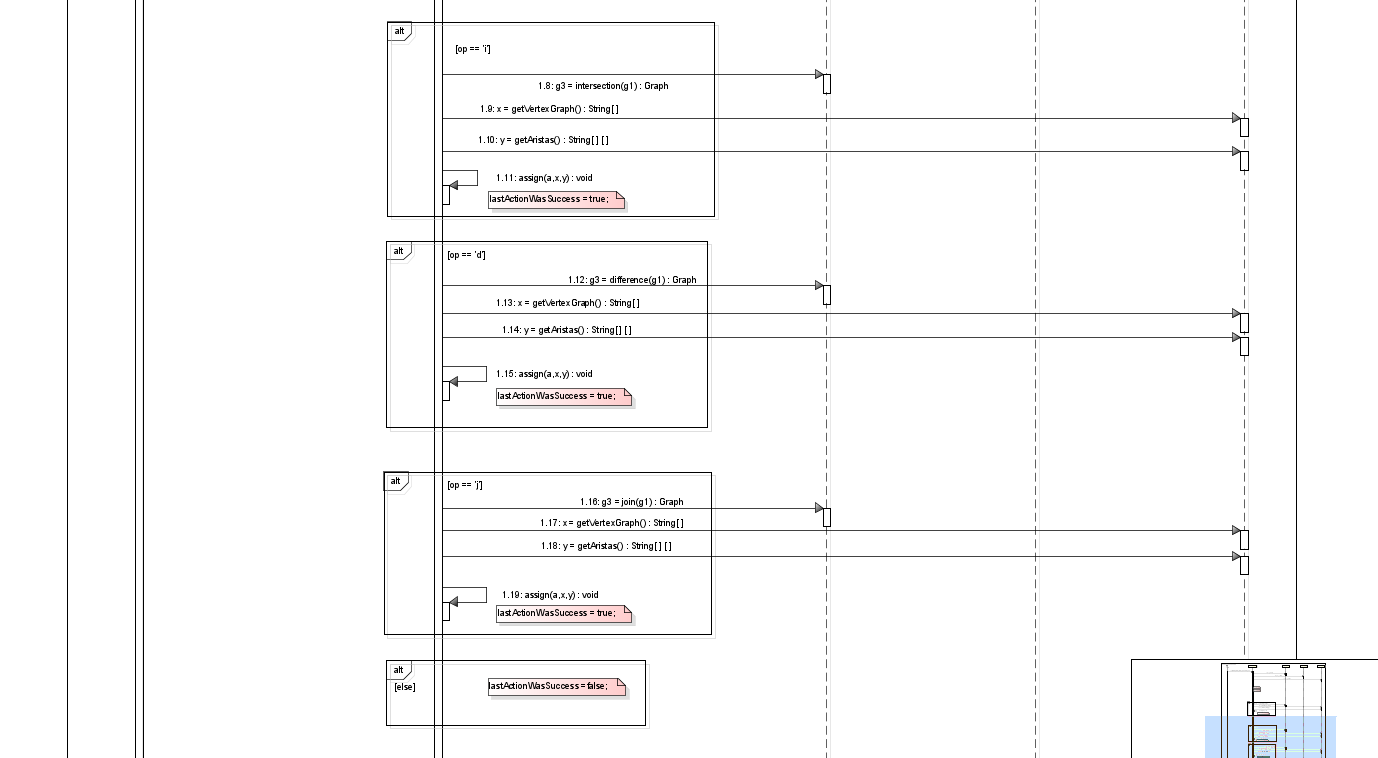
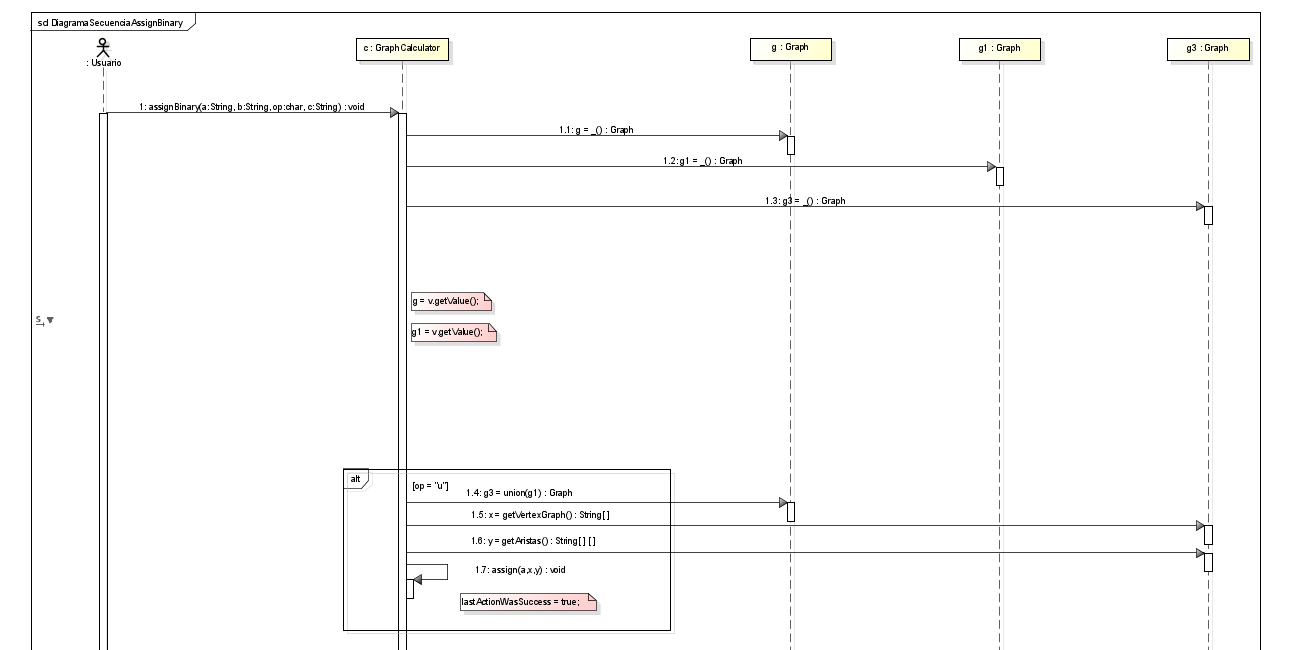


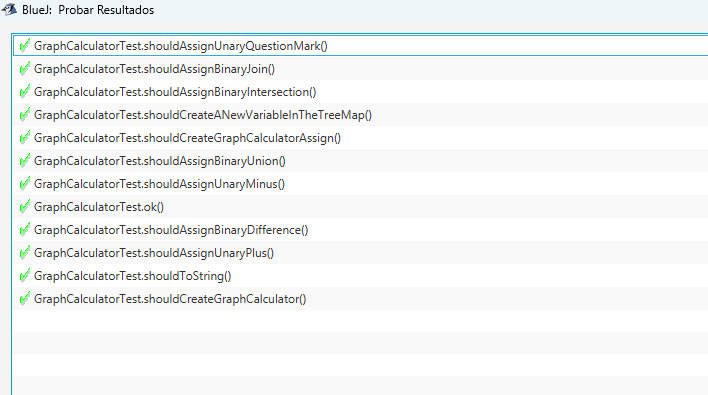
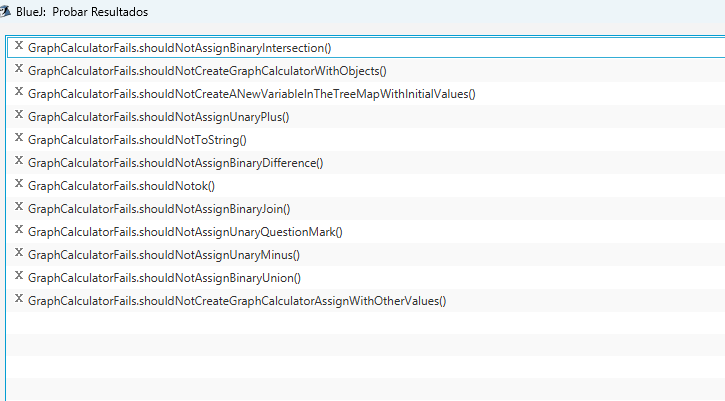
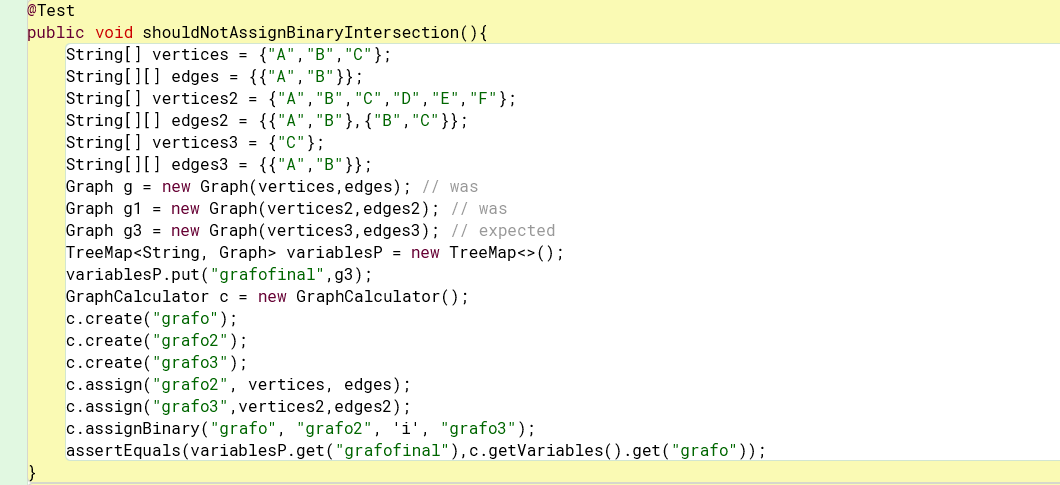
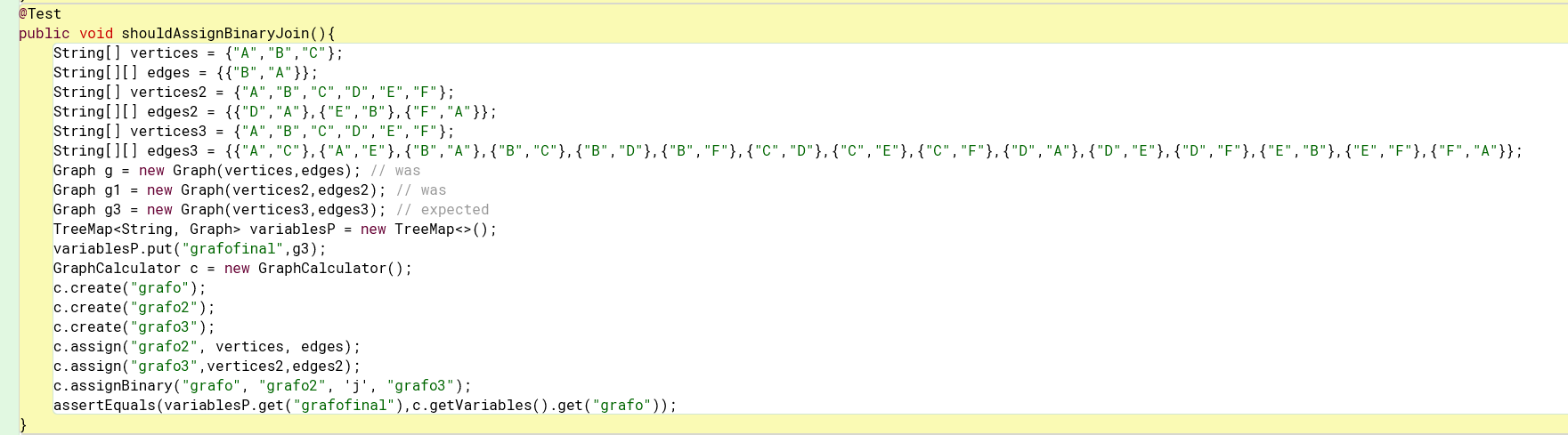
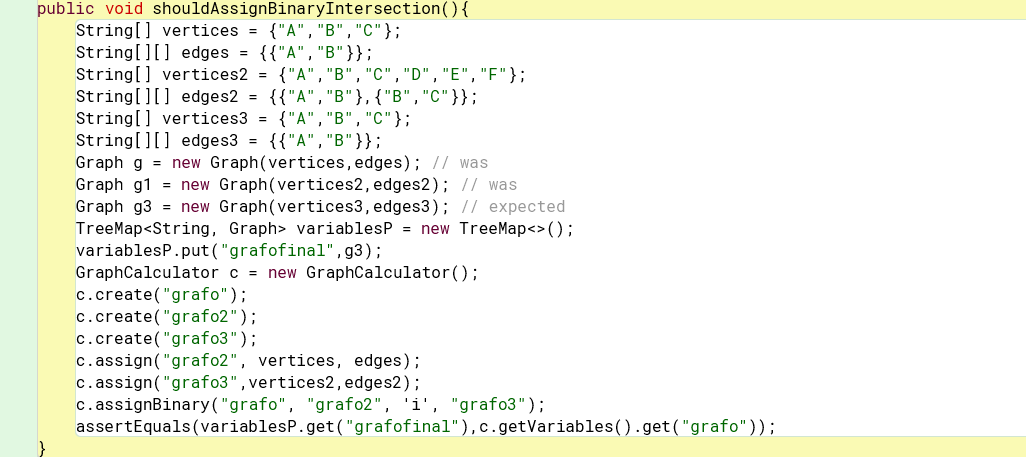
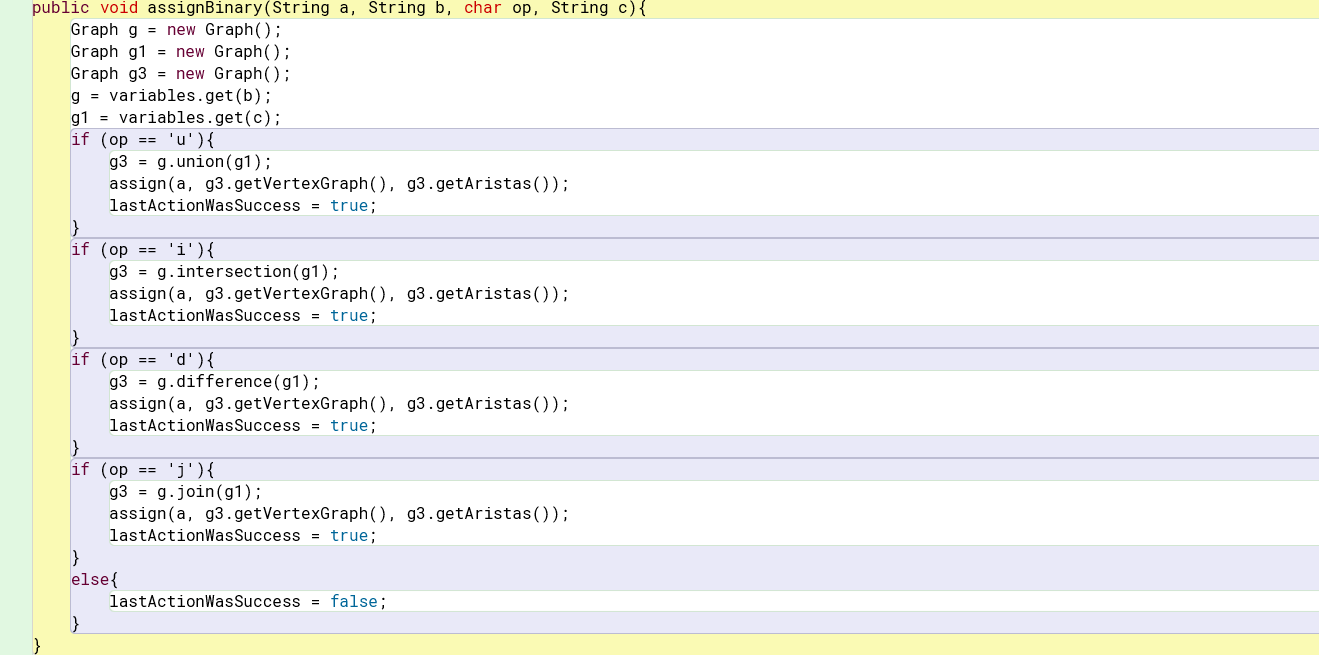
Ciclo 2 : Operaciones unarias: insertar y eliminar arcos; consultar si un conjunto de vertices pertenece al graph y retornar el camino que pasa por un conjunto de vértices



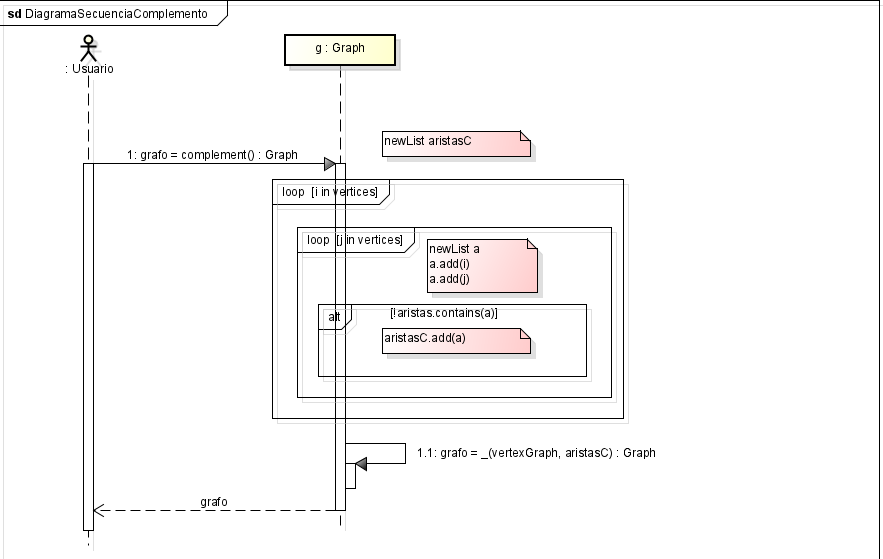


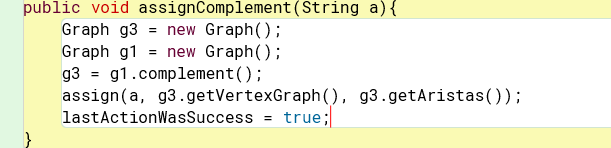
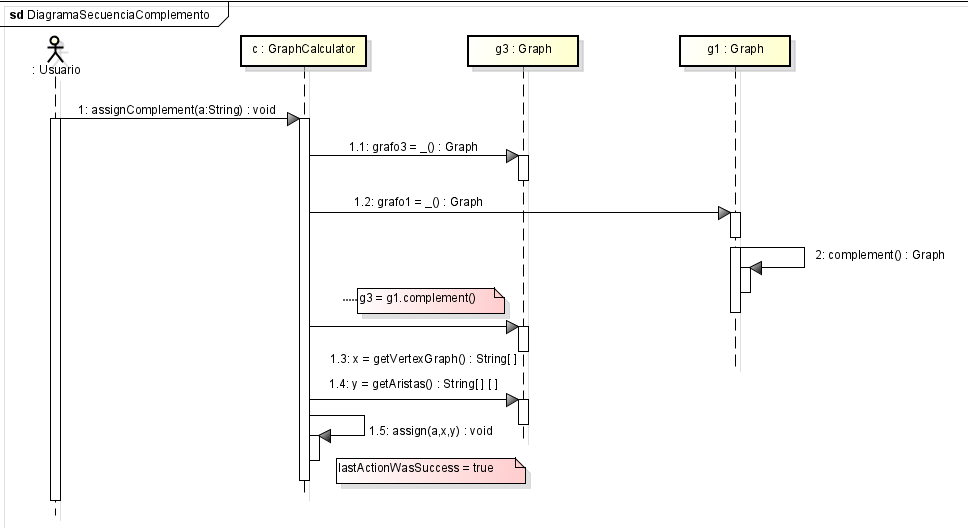
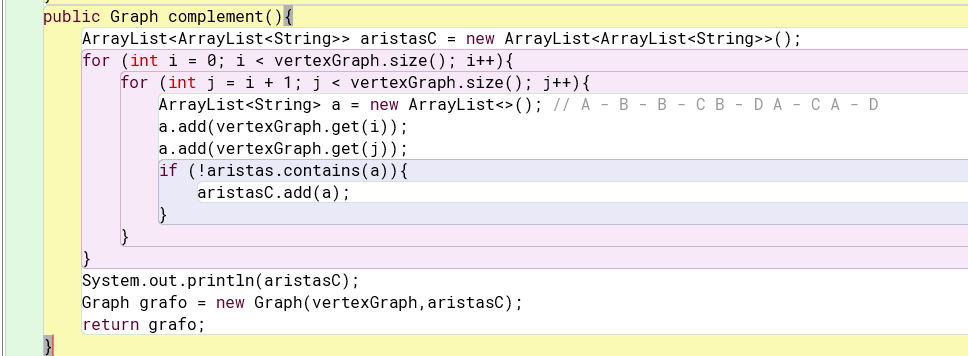
Ciclo 3 : Operaciones binarias: union, intersección, diferencia y junta

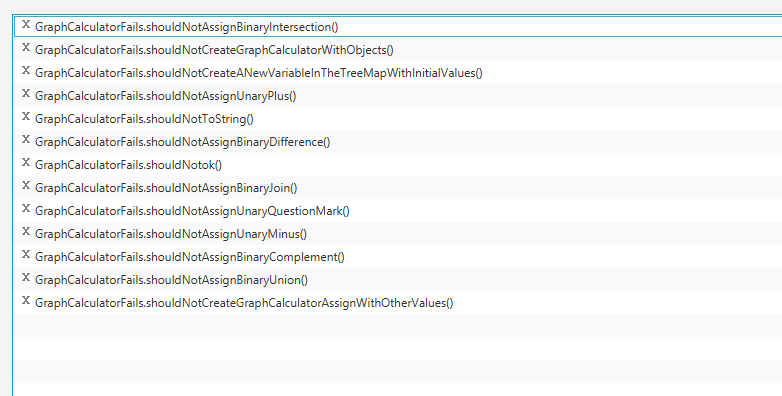
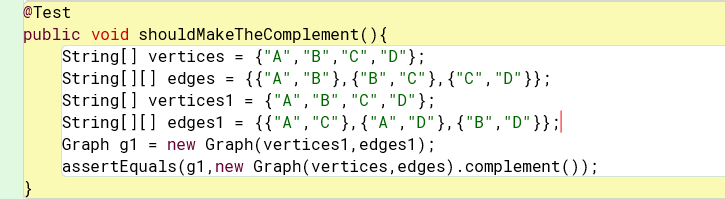


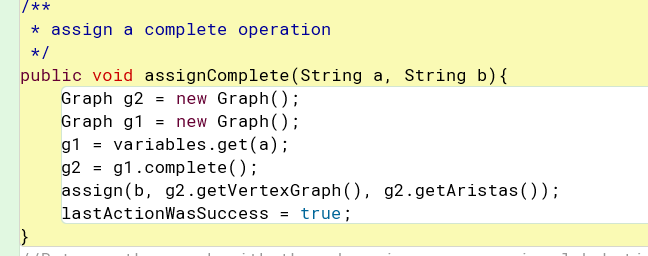
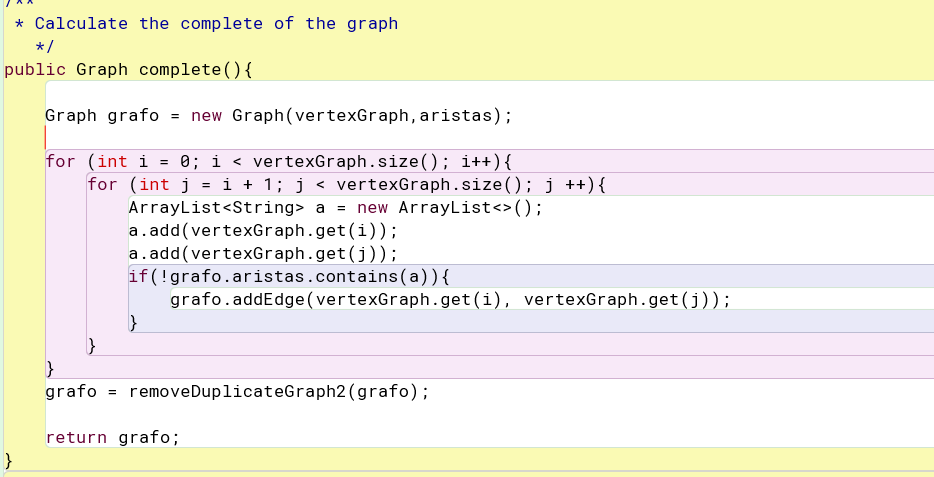
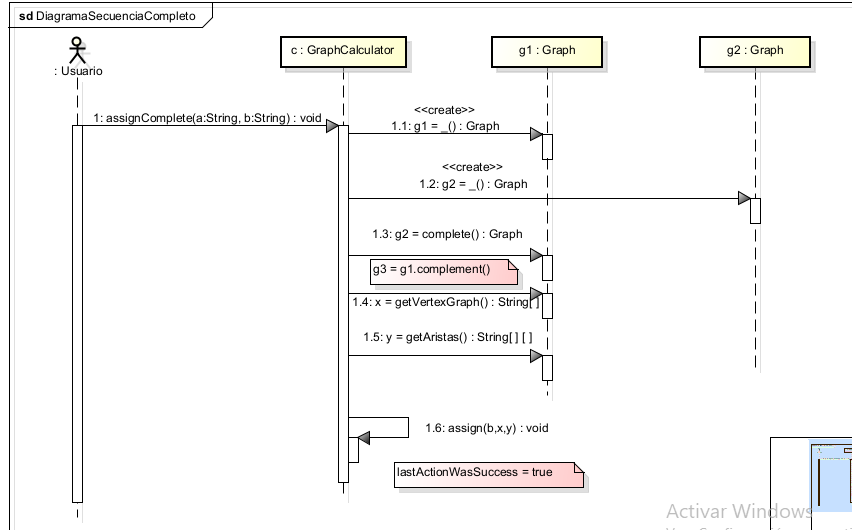


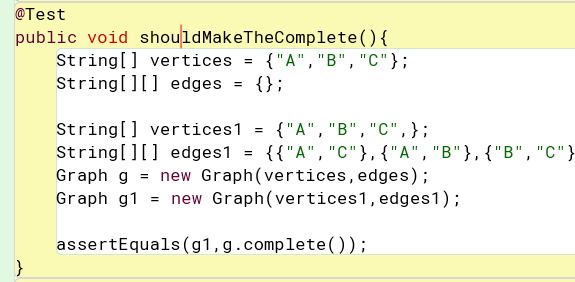
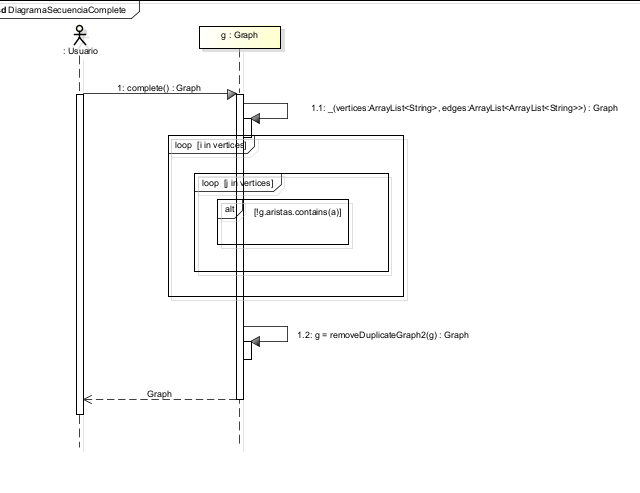
BONO Ciclo 4 : Defina dos nuevas operaciones

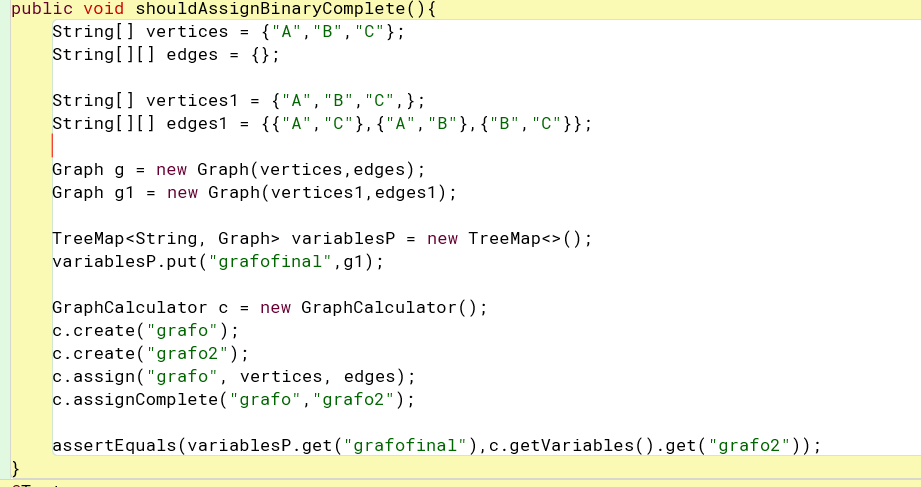
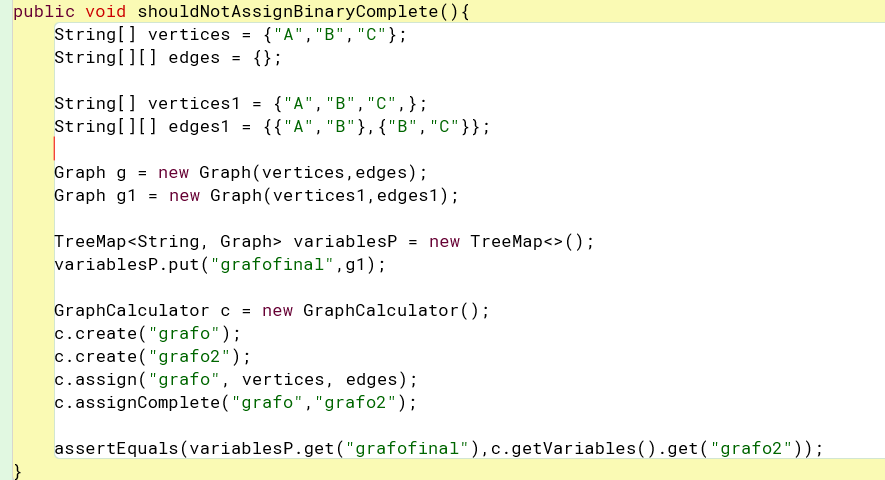












|  |  |  |
| --- | --- | --- |
| Ciclo | GraphCalculator | GraphCalculatorTest |
| Ciclo 1 | GraphCalculator(), create(String nombre), toString(String graph) | ShouldCreateGraphCalculator, shouldNotCreateGraphCalculatorWithObjects, shouldCreateANewVariableInTheTreeMap, shouldNotCreateANewVariableInTheTreeMapWithInitialValues, shouldToString, shouldNotToString |
| Ciclo 2 | AssignUnary, addEdge, removeEdge | ShouldAssignUnaryPlus, shouldAssignUnaryMinus, shouldAssignUnaryQuestionMark, shouldNotAssignUnaryPlus, shouldNotAssignUnaryMinus, shouldNotAssignUnaryQuestionMark |
| Ciclo 3 | AssignBinary, union, intersection, difference, join, Assign. | shouldAssignBinaryUnion, shouldAssignBinaryIntersection, shouldAssignBinaryDifference, shouldAssignBinaryJoin, shouldNotAssignBinaryUnion, shouldNotAssignBinaryIntersection, shouldNotAssignBinaryDifference, shouldNotAssignBinaryJoin |
| Ciclo 4 | Complement, complete | ShouldAssignBinaryComplement, shouldNotAssignBinaryComplement, ShouldMakeTheComplete, ShouldNotMakeTheComplete |

RETROSPECTIVA

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

Nombre)

Tulio Riaño Sánchez: 22 horas aprox

Andrés Cardozo: 22 horas aprox

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

El estado actual del laboratorio es completo, resaltando la comuniación y el compromiso se logró llevar a este estado.

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

Las dos prácticas XP fueron demasiado utiles ya que nos permitía definir como realizar el respectivo laboratorio, pero encontramos la más relevante all code must have unit test ya que nos permitio observar los resultados de cada uno de los métodos que realizabamos.

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

Aprender la manera de hacer pruebas en un grafo y una calculadora de este.

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

Realizar de manera correcta todas las operaciones de los grafos, así como poder pasarlo a Graph Calculator en TreeMap, la manera en que lo resolvimos fue mediante la bibliografía que explicaba la manera de hacer operaciones entre grafos.

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

Resultados?

De nuevo resaltamos la comunicación y disposición para desarrollar el laboratorio, así mismo mantenemos nuestro compromiso por la ayuda mutua que se evidencia en el desempeño.

7. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares

Adecuados.

Se adjuntan las respectivas citas bibliográficas, todas fueron demasiadas utiles ya que al tratar con TreeMap tuvimos que investigar en el API, así como en páginas que nos dieran facilidades para utilizar sus respectivos métodos.

REFERENCIAS BIBLIOGRÁFICAS

1. <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>
2. <https://docs.oracle.com/javase/8/docs/api/java/util/Vector.html>

3.<https://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html>

4.<https://howtodoinjava-com.translate.goog/java/collections/arraylist/arraylist-vs-vector/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc>

5.[Junit Assert y AssertEquals con ejemplo](https://www.guru99.com/es/junit-assert.html)

6. <https://www.geeksforgeeks.org/array-of-arraylist-in-java/>

7. <https://www.geeksforgeeks.org/adjacency-matrix/>

8. <https://www.geeksforgeeks.org/union-and-intersection-of-two-graphs/>

9. <https://www.geeksforgeeks.org/arrays-aslist-method-in-java-with-examples/>

10. <https://programacion.net/articulo/ordenar-alfabeticamente-un-array-de-textos-con-java_1178>

11. <https://stackoverflow.com/questions/16184653/sorting-arraylist-of-arrayliststring-in-java>

12. <https://www.geeksforgeeks.org/how-to-remove-duplicates-from-arraylist-in-java/>

13. <https://www.geeksforgeeks.org/trim-remove-leading-trailing-spaces-string-java/>

14. <https://www.geeksforgeeks.org/arraylist-retainall-method-in-java/>

15.<https://www.w3schools.com/java/ref_arraylist_removeall.asp>

16. <https://sentry.io/answers/how-do-i-declare-and-initialize-an-array-in-java/#:~:text=The%20easiest%20way%20to%20declare,by%20using%20the%20following%20syntax.&text=int%5B%5D%20myArray%20%3D%20new%20int,of%20an%20array%20of%20integers>.

17. <https://es.stackoverflow.com/questions/221440/como-agregar-elementos-a-un-array-en-java>

18. <https://es.stackoverflow.com/questions/206103/pasar-arraylist-a-arreglo>

19. <https://www.geeksforgeeks.org/how-to-iterate-over-a-treemap-in-java/>

20. <https://ceur-ws.org/Vol-1810/GraphQ_paper_04.pdf>

21. <https://www.geeksforgeeks.org/implementation-of-bfs-using-adjacency-matrix/>

22. <https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/Array.html>

23. <https://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html>

24. <https://www.geeksforgeeks.org/treemap-put-method-in-java/>

25. <https://www.geeksforgeeks.org/how-to-iterate-over-a-treemap-in-java/>

26. <https://www.geeksforgeeks.org/initialize-an-arraylist-in-java/>

27. <https://es.stackoverflow.com/questions/206103/pasar-arraylist-a-arreglo>

28. <https://www.w3schools.com/java/tryjava.asp?filename=demo_ref_arraylist_removeall>

29. <https://www.geeksforgeeks.org/implementation-of-bfs-using-adjacency-matrix/>

30. <https://math.stackexchange.com/questions/1691013/how-does-one-join-two-graphs-in-graph-theory>

31. <https://blog.visual-paradigm.com/es/everything-you-need-to-know-about-sequence-diagrams/>

32. <https://www.geeksforgeeks.org/clone-method-in-java-2/>

33. <https://codegym.cc/es/groups/posts/es.297.treemap-en-java>