**ESCUELA COLOMBIANA DE INGENIERÍA**

**PROGRAMACIÓN ORIENTADA A OBJETOS 2023-2**

**Diseño y Pruebas. Interacción entre objetos.**

**CRISTIAN ALVAREZ – JULIANA BRICEÑO**

# Laboratorio 2/6

## OBJETIVOS

Desarrollar competencias básicas para:

1. Desarrollar una aplicación aplicando BDD y MDD.
2. Realizar diseños (directa e inversa) utilizando una herramienta de modelado (astah)
3. Manejar pruebas de unidad usando un *framework ( junit)*
4. Apropiar nuevas clases consultando sus especificaciones (API java)
5. Experimentar las prácticas XP : **Coding** Code the [unit test first](http://www.extremeprogramming.org/rules/testfirst.html). **Testing** All code must have [unit tests](http://www.extremeprogramming.org/rules/unittests.html).



## ENTREGA

* Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.
* En el foro de entrega deben indicar el estado de avance de su laboratorio y los problemas pendientes por resolver.
* Deben publicar el avance (al final de la sesión) y la versión definitiva (en la fecha indicada) en los espacios preparados para tal fin

**CONTEXTO**

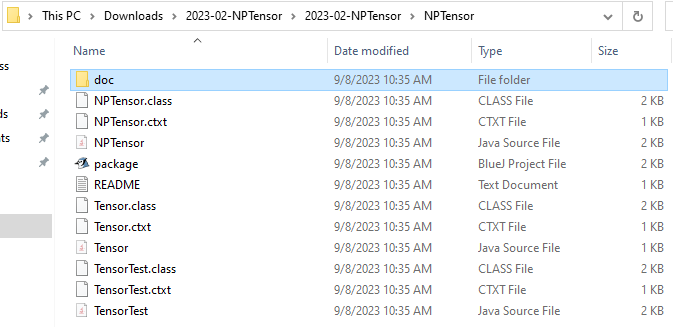
# Objetivo

En matemáticas, un **tensor** es un objeto algebraico que describe una relación multilineal entre conjuntos de objetos algebraicos relacionados con un espacio vectorial. Un **tensor** puede representarse como una arreglo multidimensional.

El objetivo de este laboratorio es implementar un subconjunto de las operaciones que ofrece numpy. Numpy es una biblioteca para el lenguaje de programación Python que da soporte para tensores.

# Conociendo el proyecto [En lab02.doc]

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



Dentro del directorio hay archivos .class, .java y .ctxt. Los archivos .class son archivos que la máquina virtual de java JVM puede ejecutar. Los archivos .java son el código fuente del programa. Los archivos .ctxt son archivos generados automáticamente por BlueJ en la compilación, donde se guarda información extra como de estructura y documentación. Tambien sale una carpeta llamada doc que guardan archivos .html que son la documentación del programa.

1. Exploren el proyecto en BlueJ

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

Tiene tres clases: NPTensor, Tensor y TensorTest. NPTesor tiene un atributo HashMap que almacena objetos de tipo Tensor, y TensorTest utiliza dentro de sus métodos objetos de tipo Tensor.

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

La clase principal es NPTensor porque es con la que se comunican los que quieren usar la aplicacion. La reconocemos porque el nombre del proyecto es el mismo.

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

La clase diferente es TensorTest, dado que su propósito es comprobar el correcto comportamiento del código y detectar fallos.

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

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

NPTensor

La documentación no está completa. Para empezar, no tiene una descripción general de la clase, y ningún método tiene documentación. Lo métodos tienen comentarios en el código fuente que describen el método y sus parámetros, pero estos no salen en la documentación.

Tensor

La documentación no está completa. La clase no tiene una descripción genera, y solamente uno de los dos constructores tiene la documentación. La clase ni siquiera tiene comentarios en el código como NPTensor.

1. 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 son el código, la documentación y los comentarios?

NPTensor

1. Atributos y métodos: Los atributos parecen estar bien, pero los métodos no hacen nada.
2. Código, documentación y comentarios: Los métodos no están definidos, ni siquiera el constructor está definido. No tiene la documentación, solo hay comentarios.

Tensor

1. Atributos y métodos: No tiene atributos y no hay nada en los métodos.
2. Código, documentación y comentarios: Solo un constructor tiene documentación, no hay comentarios y en general el código está vacío.

¿Qué son el código, la documentación y los comentarios?

Todos estos elementos se definen en el .java

1. Código:

- El código se refiere a las instrucciones escritas en lenguaje Java que definen la funcionalidad de un programa.

- Es la parte principal y ejecutable de un programa Java.

- Debe seguir las reglas de sintaxis de Java para compilar y ejecutar correctamente.

- El código contiene las declaraciones de clases, métodos, variables, expresiones y otras estructuras necesarias para que el programa funcione.

2. Documentación:

- La documentación se refiere a información escrita que explica el propósito, el funcionamiento y el uso de un programa, clases o métodos.

- En Java, se utiliza principalmente JavaDoc para generar documentación a partir de comentarios especiales colocados en el código fuente.

- La documentación es útil para otros programadores que trabajan en el mismo proyecto, ya que proporciona una referencia clara sobre cómo usar y comprender el código.

- La documentación bien escrita mejora la mantenibilidad del código y la colaboración en equipo.

3. Comentarios:

- Los comentarios son anotaciones escritas en el código fuente que no afectan la ejecución del programa, pero proporcionan información adicional para los programadores.

- Los comentarios se utilizan para explicar el propósito de las líneas de código, aclarar el flujo de control o hacer observaciones importantes.

- En Java, los comentarios pueden ser de una línea (//) o de varias líneas (/\* ... \*/).

- Los comentarios son cruciales para la comprensión del código y su mantenimiento, especialmente para otros desarrolladores que puedan trabajar en el proyecto en el futuro.

**Ingeniería reversa** [En lab02.doc TensorNP.asta]

## MDD MODEL DRIVEN DEVELOPMENT

1. Complete el diagrama de clases correspondiente al proyecto

(No incluya la clase de pruebas)

1. ¿Cuál nuevo contenedor está definido? Consulte la especificación y el API Java [[1]](#footnote-1)¿Qué diferencias hay entre el nuevo contenedor, el ArrayList y el vector [] que conocemos?

El nuevo contenedor es HashMap.

Diferencias:

1. `HashMap`:
   * Implementa una estructura de tabla de dispersión que almacena pares clave-valor.
   * No permite elementos duplicados en las claves y permite un valor nulo.
   * Los elementos no están ordenados por defecto.
   * Es eficiente para buscar y recuperar valores a través de claves (búsqueda rápida).
   * No garantiza un orden específico de los elementos.
2. `ArrayList`:
   * Implementa una lista dinámica (arreglo redimensionable) que almacena elementos de manera secuencial.
   * Permite elementos duplicados y valores nulos.
   * Los elementos están ordenados según el orden de inserción.
   * Es eficiente para acceder a elementos por índice (búsqueda rápida).
   * No es eficiente para eliminar elementos en posiciones intermedias.
3. `Vector`:
   * Es similar a `ArrayList`, pero es una versión sincronizada, lo que significa que es seguro para operaciones concurrentes.
   * Permite elementos duplicados y valores nulos.
   * Los elementos están ordenados según el orden de inserción.
   * Es menos eficiente que `ArrayList` para operaciones no concurrentes debido a la sincronización.
   * Se utiliza generalmente en situaciones donde se requiere seguridad en hilos de ejecución concurrentes.

# Conociendo Pruebas en BlueJ [En lab02.doc \*.java]

## De TDD → BDD (TEST → BEHAVIOUR DRIVEN DEVELOPMENT)

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

1. Revisen el código de la clase TensorTest. ¿cuáles etiquetas tiene (componentes con símbolo @)? ¿cuántos métodos tiene? ¿cuantos métodos son de prueba? ¿cómo los reconocen?

Las etiquetas son: @BeforeClass, @Before y @Test.

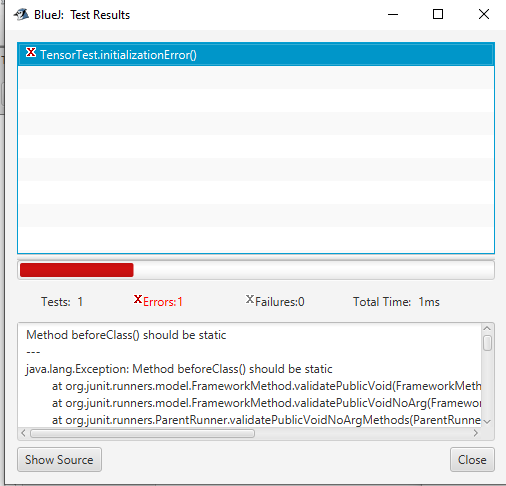
Tiene 6 métodos.

Tiene 4 métodos de prueba. Los reconocemos porque tienen la etiqueta @Test

1. Ejecuten los tests de la clase TensorTest. (click derecho sobre la clase, Test All)

¿cuántas pruebas se ejecutan? ¿cuántas pasan? ¿por qué?

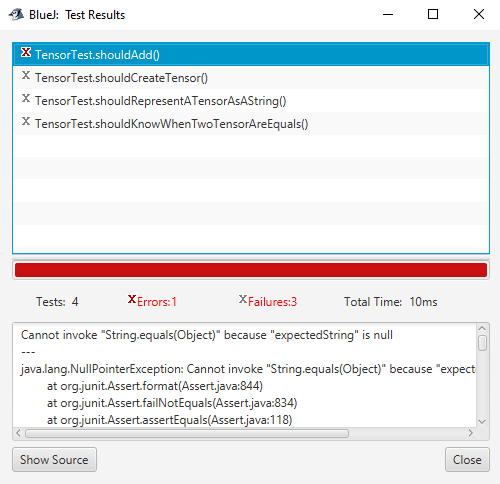
Al principio solo se ejecuta una prueba porque sale un error en el @BeforeClass, diciendo que debe ser estático.



Después de hacer ese cambio se ejecutan 4 pruebas, de las cuales 1 tiene un error y 3 tienen un fallo.

El error se genera porque espera un objeto para comparar, pero le llega NULL, por lo que no tiene con que comparar.

Los fallos se dan porque los resultados de las pruebas no son los esperados.



1. Estudie las etiquetas encontradas en 1. Expliquen en sus palabras su significado.

@BeforeClass: se ejecuta una vez, antes de comenzar todos los tests. Esto métodos deben definirse como static para trabajar con JUnit.

@Before:  ejecuta el método que la contiene justo antes de cada test.

@Test: indica que el método que la contiene es un test: expected y timeout.

1. Estudie los métodos assertTrue, assertFalse, assertEquals, assertArrayEquals, assertNull y fail de la clase assert del API JUnit [[2]](#footnote-2). Explique en sus palabras que hace cada uno de ellos.

assertTrue: sirve para evaluar si una condición es true.

assertFalse: sirve para evaluar si una condición es false.

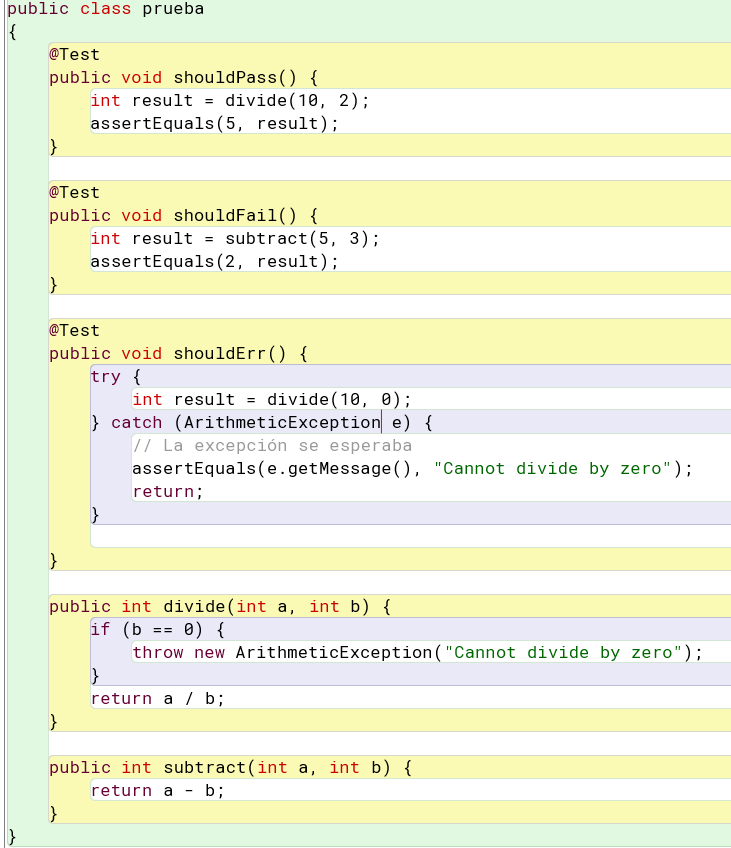
assertEquals: sirve para verificar si dos valores son iguales, compara el valor esperado con el valor real y determina si son iguales.

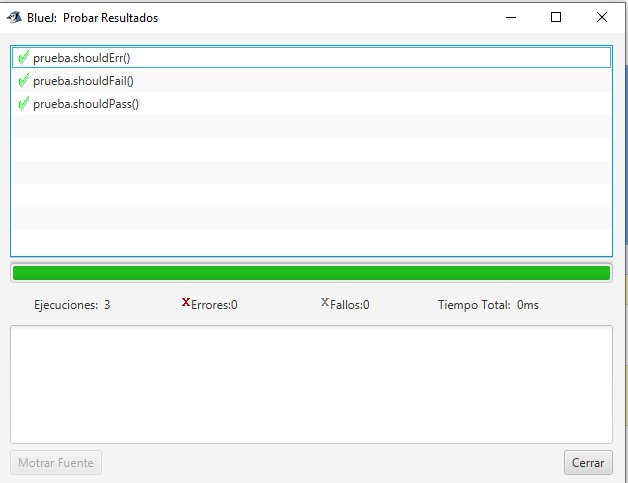
assertArrayEquals: sirve para verificar si dos arreglos son iguales en términos de contenido y orden.

assertNull: sirve para verificar si un objeto es Null.

fail: sirve para identificar cuando una condición específica de una prueba falla.

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





# Prácticando Pruebas en BlueJ [En lab02.doc \*.java]

## De TDD → BDD (TEST → BEHAVIOUR DRIVEN DEVELOPMENT)

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

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

R//: Determinamos los atributos de Data2D, Data3D, Data4D, esto debido a que debíamos guardar tensores de diferentes dimensiones y fue la manera que encontramos para guardarlos dependiendo de su tamaño.

1. Determinen el invariante de la clase Tensor. Justifique la decisión.

R//:

getNumberOfElements()

* El invariante se mantiene en todas las funciones del código, ya sea que se inicialice el tensor con un valor o con una matriz de valores, o se cambie su forma.

Por ejemplo, en la función reshape(), el código primero verifica que la nueva forma es compatible con el número de elementos del tensor original. Si es así, el código copia los valores del tensor original en una nueva matriz y luego establece la forma del tensor en la nueva forma.

* Estas líneas de código declaran que las variables data2D, data3D y data4D son matrices de números enteros.

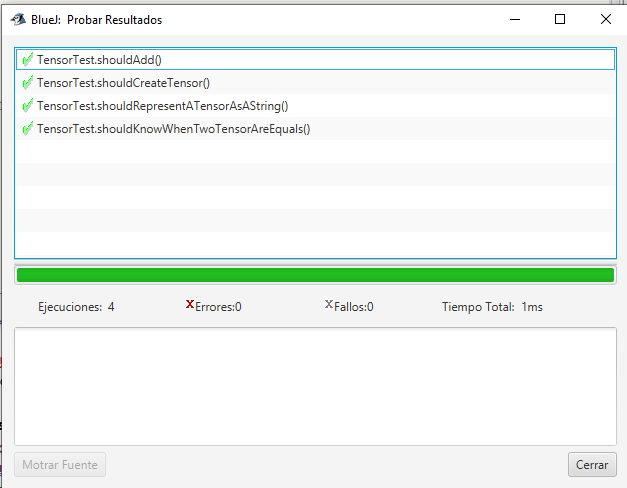
El invariante se mantiene en todas las funciones del código, ya sea que se inicialice el tensor con un valor o con una matriz de valores, o se cambie su forma.

Por ejemplo, en la función value(), el código devuelve el valor del tensor en los índices especificados. El código primero verifica que los índices sean válidos. Si lo son, el código devuelve el valor del tensor en los índices especificados de la matriz correspondiente.

1. Implementen únicamente los métodos de Tensor necesarios para pasar todas las pruebas definidas. ¿Cuáles métodos implementaron?

R//: Implementamos como nuevos los métodos de getNumberOfDimensions(), getNumberOfElements(), getShape(), getDataIntoArray(), equals(Tensor other), slice(int[] list, int start, int end, int step).

1. Capturen los resultados de las pruebas de unidad.



# Desarrollando TensorNP

**BDD - MDD**

[En lab02.doc, TensorNP.asta, \*.java]

Para desarrollar esta aplicación vamos a considerar algunos mini-ciclos. En cada mini-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

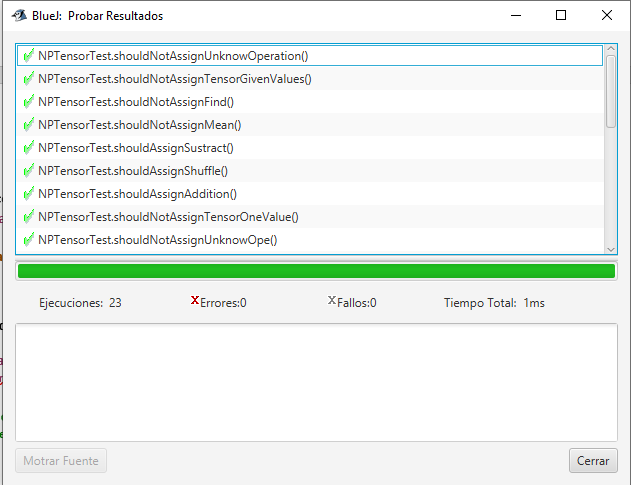
**(piense en los deberia y los noDeberia (should and shouldNot)**

**3. Diseñar los métodos**

**(Use diagramas de secuencia. En astah, adicione el diagrama al método)**

## 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:**

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

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

**(piense en los deberia y los noDeberia (should and shouldNot)**

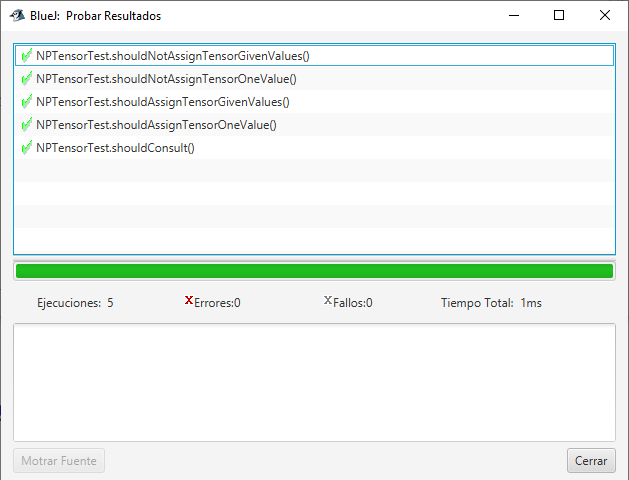
**3. Diseñar los métodos**

**(Use diagramas de secuencia. En astah, adicione el diagrama al método)**

**Ya**

## 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)



**CICLO 2:**

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

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

**(piense en los deberia y los noDeberia (should and shouldNot)**

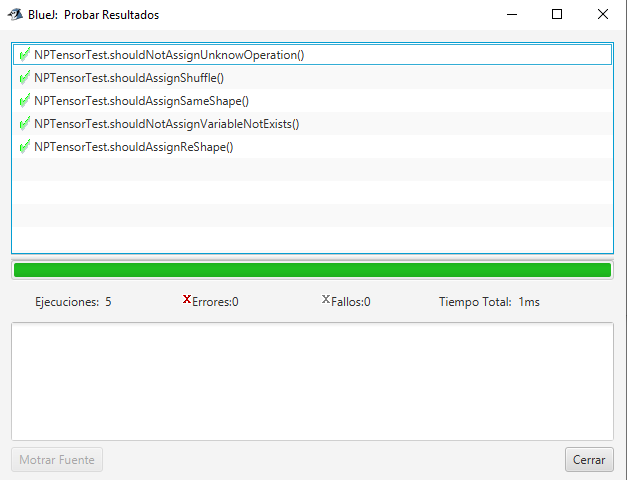
**3. Diseñar los métodos**

**(Use diagramas de secuencia. En astah, adicione el diagrama al método)**

ya

## 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)



**CICLO 3:**

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

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

**(piense en los deberia y los noDeberia (should and shouldNot)**

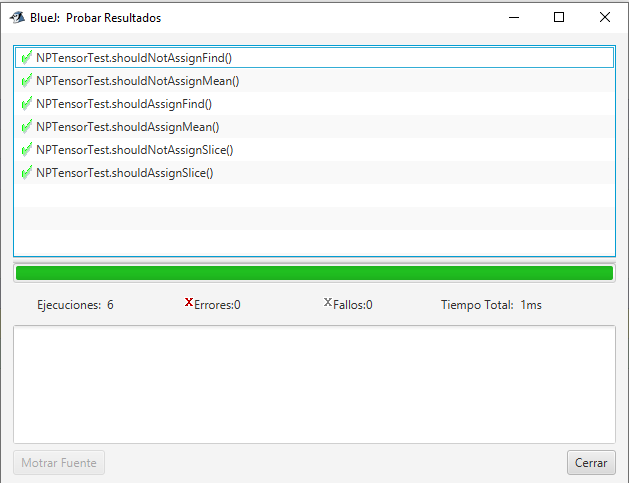
**3. Diseñar los métodos**

**(Use diagramas de secuencia. En astah, adicione el diagrama al método)**

**1.**

## 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)



**CICLO 4:**

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

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

**(piense en los deberia y los noDeberia (should and shouldNot)**

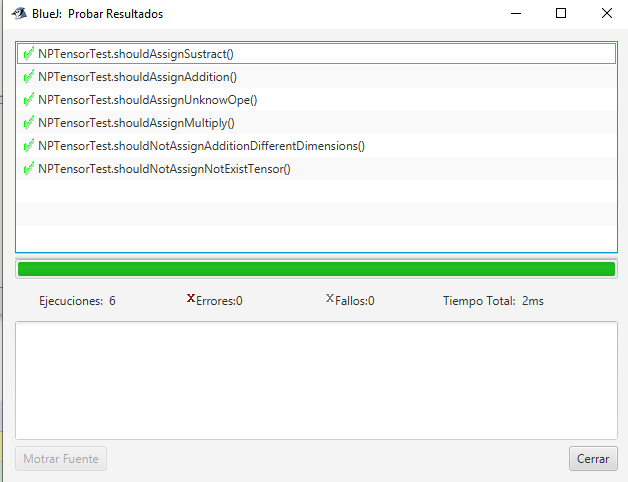
**3. Diseñar los métodos**

**(Use diagramas de secuencia. En astah, adicione el diagrama al método)**

**Ya**

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

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

1. 

Ciclo 1 : Operaciones básicas de tensores: declarar, asignar un valor y consultar

Ciclo 2 : Operaciones unarias sin parámetros: dimensiones, redimensionar y barajar[[3]](#footnote-3)

Ciclo 3 : Operaciones unarias con parámetros: rebanada, media por ejes, find[[4]](#footnote-4)

Ciclo 4 : Operaciones binarias uno a uno: suma, resta, multiplicación

**BONO** Ciclo 5 : Defina tres nueva funcionalidades.

Completen la siguiente tabla indicando el número de ciclo y los métodos asociados de cada clase.

|  |  |  |
| --- | --- | --- |
| Mini-ciclo | TensorNP | TensorNPTest |
| 1 | assign(String name, int shape[], int value ) | public void shouldAssignTensorOneValue()  shouldNotAssignTensorOneValue() |
| 1 | assign(String name, int shape[], int[] values) | shouldAssignTensorGivenValues()  shouldNotAssignTensorGivenValues() |
| 1 | String consult(String key) | shouldConsult()  shouldNotConsultDontExist() |
| 2 | assign(String a, String unary, String b) | shouldAssignSameShape()  shouldAssignReShape()  shouldAssignShuffle()  shouldNotAssignVariableNotExists()  shouldNotAssignUnknowOperation() |
| 3 | assign(String a, String unary, String b, int [] parameters) | shouldAssignSlice()  shouldNotAssignSlice()  shouldAssignMean()  shouldNotAssignMean()  shouldAssignFind()  shouldNotAssignFind() |
| 4 | assign(String a, String b, String sBinary, String c) | shouldAssignAddition()  shouldAssignSustract()  shouldAssignMultiply()  shouldNotAssignAdditionDifferentDimensions()  shouldNotAssignNotExistTensor()  shouldNotAssignUnknowOpe() |

## RETROSPECTIVA

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

R//: 35 horas/ Juliana, 35 horas/ Cristian

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

R//: Esta completo porque lo hicimos con tiempo y detenidamente

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

R//: Usamos las Unit test, descubrimos que las pruebas unitarias son una práctica esencial en XP. Ayudan a mejorar la calidad del código, aumentar la confianza en el mismo, y proporcionar una mejor documentación, el hecho de escribir pruebas antes del código en sí, es muy importante. Esto se debe a que ayuda a garantizar que el código esté diseñado de forma que se pueda probar.

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

R//: No rendirnos, porque toco dar muchas soluciones a problemas de ambigüedad que nos llevaban a sobre pensar el código.

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

R//: Dificultad para entender el funcionamiento de numpy y luego implementarlo a los n dimensiones de los tensores

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

R//: Hicimos bien el hecho de que no procrastinamos y que tuvimos mucha y buena comunicación durante el desarrollo del laboratorio.

1. https://docs.oracle.com/javase/8/docs/api/ [↑](#footnote-ref-1)
2. [(http://junit.org/javadoc/latest/](http://junit.org/javadoc/latest/)) [↑](#footnote-ref-2)
3. [shape](https://www.w3schools.com/python/numpy/numpy_array_shape.asp) , [reshape](https://www.w3schools.com/python/numpy/numpy_array_reshape.asp), [shuffle](https://www.w3schools.com/python/numpy/numpy_random_permutation.asp) [↑](#footnote-ref-3)
4. [slice](https://www.w3schools.com/python/numpy/numpy_array_slicing.asp) , [mean](https://www.pythonprogramming.in/numpy-aggregate-and-statistical-functions.html), find (indices of tensor elements equals) [↑](#footnote-ref-4)