Escuela Colombiana de Ingeniería Julio Garavito

Ingeniería de Sistemas

Programación Orientada a Objetos

LABORATORIO 02
08/09/2023

Joan Steven Acevedo Aguilar
joan.acevedo@mail.escuelaing.edu.co

Santiago Sanchez Monroy
santiago.sanchez-m@mail.escuelaing.edu.co

Profesor a cargo:

Maria Irma Diaz Rozo
maria.diaz@mail.escuelaing.edu.co

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. Testing All code must have unit tests.

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 un 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

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.

    - Directorios:
        - C:\Users\joan.acevedo\Downloads\2023-02-NPTensor
        - C:\Users\joan.acevedo\Downloads\2023-02-NPTensor\NPTensor
        - C:\Users\joan.acevedo\Downloads\2023-02-NPTensor\NPTensor\doc
    - Extensiones:
        - .astah
        - .class
        - .ctxt
        - .java
        - .bluej
        - .TXT
        - .css
        - .html
        - .js

2. Exploren el proyecto en BlueJ ¿Cuántas clases tiene? ¿Cuál es la relación entre ellas? ¿Cuál es la clase principal de la aplicación? ¿Cómo la reconocen? ¿Cuáles son las clases “diferentes”? ¿Cuál es su propósito?

    - Tiene 3 clases
    - Relaciones de uso
    - NP Tensor
    - Ya que en ella encontramos los métodos de asignación de valores (assign()) y de verificación (ok())
    - La clase diferente es “TensorTest”

- Evaluar el buen funcionamiento de los distintos métodos

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 de cada clase: encabezado y métodos)

   No está completa la documentación de todas las clases
   - NPTensor
     - Encabezado: No tiene encabezado
     - Métodos: ningún método tiene la documentación de manera correcta, debido a que está como si fueran comentarios en el código.
   - Tensor
     - Encabezado: No tiene encabezado.
     - Métodos: Solamente uno de los constructores tiene documentación, por otra parte, los demás métodos no tienen.

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 son el código, la documentación y los comentarios?

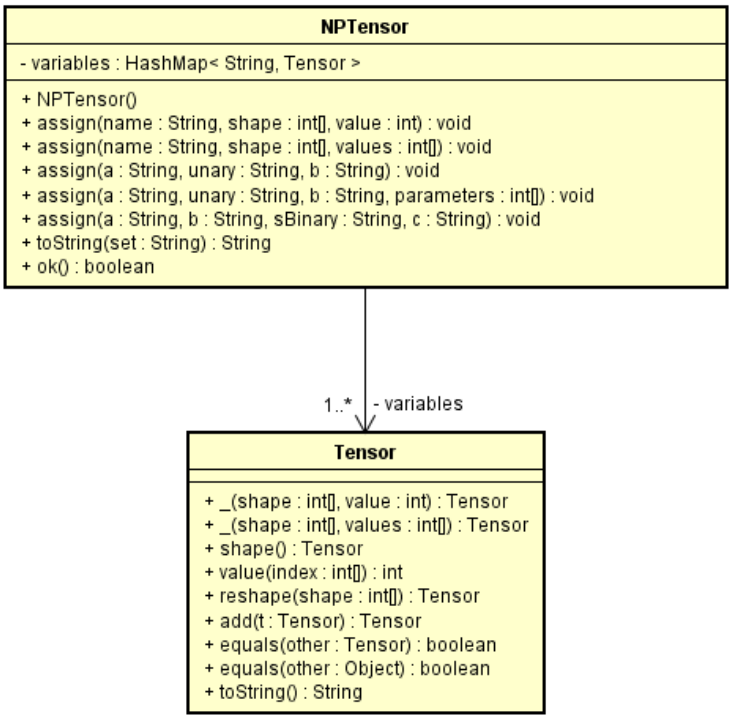
   - NPTensor
     - Primera dimensión:
       - 1 atributo privado
       - 8 métodos públicos
     - Segunda dimensión
       - No cuenta con una correcta documentación
       - Respecto al código, solo están inicializados los métodos pero no tienen contenido
       - Cada método cuenta con comentarios
   - Tensor
     - Primera dimensión:
       - 0 atributos
       - 9 métodos públicos
     - Segunda dimensión
       - No tiene documentación
       - Respecto al código, solo están inicializados los métodos pero no tienen contenido
       - No tiene ningún comentario

Ingeniería reversa

MDD MODEL DRIVEN DEVELOPMENT

1. Complete el diagrama de clases correspondiente al proyecto (No incluya la clase de pruebas)

   Hasta el momento, este es el diagrama de clases actual:



2. ¿Cuál nuevo contenedor está definido? Consulte la especificación y el API Java ¿Qué diferencias hay entre el nuevo contenedor, el ArrayList y el vector [] que conocemos?
   El nuevo contenedor que está definido en la clase NPTensor es un HashMap y está así:

```
private HashMap<String, Tensor> variables;
```

   HashMap en Java:
   Descripción: HashMap es una estructura de datos en Java que implementa la interfaz Map. Representa una colección de pares clave-valor, donde cada clave se asigna a un valor único. HashMap utiliza una tabla hash para almacenar los datos, lo que permite un acceso rápido a los valores mediante la clave.

   Características clave:

   - No permite claves duplicadas; cada clave está asociada con un solo valor.
   - Las claves pueden ser nulas (solo una clave nula permitida).
   - Los valores pueden ser nulos (puede haber múltiples valores nulos).
   - No garantiza un orden específico de los elementos.

Operaciones comunes:

- put(key, value): Agrega un par clave-valor al mapa.
- get(key): Recupera el valor asociado con una clave.
- remove(key): Elimina un par clave-valor del mapa.
- containsKey(key): Verifica si una clave está presente en el mapa.
- size(): Devuelve el número de pares clave-valor en el mapa.

Diferencias entre HashMap, ArrayList y [] (Arreglos en Java):

- Dinamicidad: HashMap y ArrayList son dinámicos, lo que significa que pueden crecer o reducirse según sea necesario, mientras que los arreglos ([]) tienen un tamaño fijo.

- Acceso por clave/índice: HashMap se utiliza para buscar valores por clave, ArrayList por índice y los arreglos por índice.

- Ordenación: HashMap no garantiza un orden específico de los elementos, ArrayList mantiene el orden de inserción y los arreglos tienen un orden fijo.

- Duplicados: HashMap no permite claves duplicadas, ArrayList y los arreglos pueden contener duplicados.

- Eficiencia: La eficiencia varía según el tipo de operaciones. HashMap es eficiente para buscar valores por clave, mientras que ArrayList es eficiente para acceder a elementos por índice. Los arreglos son eficientes para el acceso directo por índice, pero no son dinámicos.

Conociendo Pruebas en BlueJ

De TDD → BDD (TEST → BEHAVIOUR DRIVEN DEVELOPMENT)

Para poder cumplir con las 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?

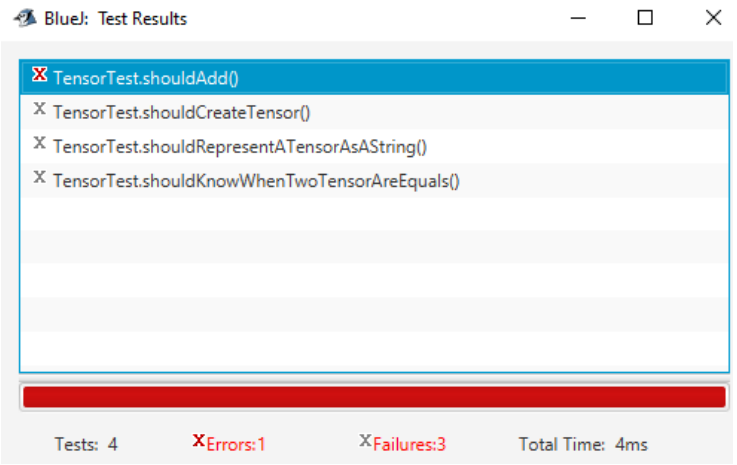
    - “@BeforeClass”, “@Before”, “@Test”
    - Tiene 6 métodos
    - 4 métodos son de prueba
    - Se reconocen por su etiqueta “@Test”

2. Ejecuten los tests de la clase TensorTest. (click derecho sobre la clase, Test All) ¿cuántas pruebas se ejecutan? ¿Cuántas pasan? ¿por qué?

Inicialmente, ninguna prueba se ejecuta, ya que lo único que se ejecuta es “beforeClass”, pero este no es un “@Test” sino un “BeforeClass”. Por lo tanto, ninguna prueba se ejecuta y ninguna prueba pasa.

El ejecutar la clase “TensorTest” sale un error de que nos indica que debería ser “static”

Después de corregirlo tenemos:



Ninguna prueba sigue sin pasar y nos encontramos con 1 Error y 4 Fallos.

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

4. Estudie los métodos assertTrue, assertFalse, assertEquals, assertArrayEquals, assertNull y fail de la clase assert del API JUnit 2. Explique en sus palabras que hace cada uno de ellos.

- AssertTrue:
  Nos indica si una condición es verdadera, si no lo es arroja un “AssertionError”
- AssertFalse:
  Nos indica si una condición es falsa, si no lo es arroja un “AssertionError”
- AssertEquals:
  Nos indica si dos objetos son iguales, si no lo son arroja un “AssertionError”. En caso de que los dos objetos sean nulos, se consideran iguales
- AssertArrayEquals:
  Nos indica si dos matrices de objetos son iguales, si no lo son arroja un “AssertionError”. En caso de que las dos matrices sean nulas, se consideran iguales

- AssertNull:
  Nos indica si un objeto es nulo, si no lo es arroja un “AssertionError”
- Fail:
  Nos indica una falla en una prueba

5. Investiguen la diferencia entre un fallo y un error en Junit.
   Una falla significa que debe corregir el código que se está probando (escritura). Un error significa que puede ser la prueba unitaria la que necesita ser corregida (lógica).

   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.

Practicando Pruebas en BlueJ

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

1. Determinen los atributos de la clase Tensor. Justifique la selección.
   Los atributos que se determinaron para la clase Tensor son:
   - private int[] shape; // Atributo para almacenar la forma (shape) del tensor
   - private int[] values; // Atributo para almacenar los valores del tensor

```
public class Tensor{
    private int[] shape;
    private int[] values;
```

2. Determinen el invariante de la clase Tensor. Justifique la decisión.
   - La longitud de la matriz shape debe ser igual a la longitud de la matriz values.
   - Cada elemento en la matriz shape debe ser mayor que cero (para representar dimensiones válidas).
   - No debe haber elementos negativos en la matriz values (todos los valores deben ser no negativos).
   - La cantidad total de elementos en la matriz values debe ser coherente con la forma especificada en la matriz shape.

3. Implementen únicamente los métodos de Tensor necesarios para pasar todas las pruebas definidas. ¿Cuáles métodos implementaron?
   Los métodos que se van a implementar para que las pruebas queden definidas son:

```java
/**
 * Creates a tensor with the defined size and value
 */
public Tensor(int[] shape, int value) {
    this.shape = shape;
    this.values = new int[calculateSize(shape)];
    for (int i = 0; i < values.length; i++) {
        this.values[i] = value;
    }
}
```

```java
public Tensor(int[] shape, int[] values) {
    this.shape = shape;
    this.values = values;
}
```

```java
public Tensor shape() {
    return new Tensor(new int[]{shape.length}, shape);
}
```

```java
/**
 *
 */
public int value(int[] index) {
    int flatIndex = 0;
    for (int i = 0; i < index.length; i++) {
        flatIndex = flatIndex * shape[i] + index[i];
    }
    return values[flatIndex];
}
```

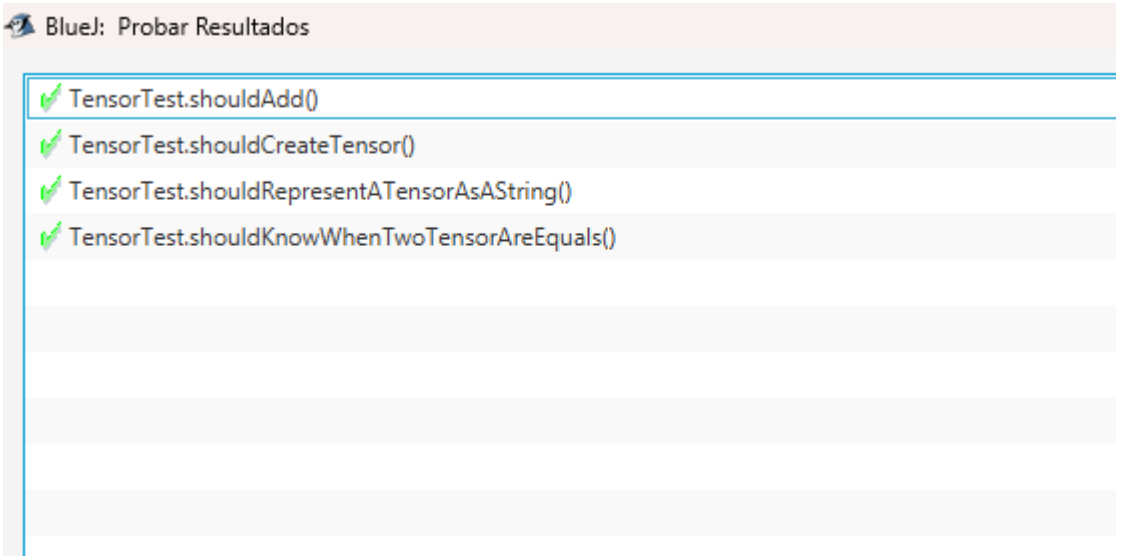
```java
public Tensor reshape(int[] newShape) {
    return new Tensor(newShape, values);
}
```

```java
/**
 * Recibe un Tensor y suma sus valores a un Tensor ya existente
 *
 * @param t, nuevo Tensor
 * @return Tensor con los nuevos valores
 */
public Tensor add(Tensor t) {
    int[] newValues = new int[values.length];
    for (int i = 0; i < values.length; i++) {
        newValues[i] = this.values[i] + t.values[i];
    }
    return new Tensor(shape, newValues);
}
```

```java
public boolean equals(Tensor other) {
    if (!java.util.Arrays.equals(this.shape, other.shape)) {
        return false;
    }
    for (int i = 0; i < values.length; i++) {
        if (this.values[i] != other.values[i]) {
            return false;
        }
    }
    return true;
}
```

```java
@Override
public boolean equals(Object other) {
    return equals((Tensor) other);
}
```

4. Capturen los resultados de las pruebas de unidad.



Desarrollando TensorNP

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 correspondientes al mini-ciclo actual.

   LISTO

2. Definir y programar los casos de prueba de esos métodos (piense en los debería y los noDeberia (should and shouldNot)

   NO SE REALIZARON PRUEBAS UNITARIAS

3. Diseñar los métodos (Use diagramas de secuencia. En astah, adicione el diagrama al método)

   ESTÁN EN ASTAH

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

   ESTÁ EN BLUEJ

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

   NO SE REALIZARON PRUEBAS UNITARIAS

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

Ciclo 1 : Operaciones básicas de tensores: declarar, asignar un valor y consultar
Ciclo 2 : Operaciones unarias sin parámetros: dimensiones, redimensionar y barajar
Ciclo 3 : Operaciones unarias con parámetros: rebanada, media por ejes, find
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

Mini - ciclo	TensorNP	TensorNPTest
1	● assign(String name, int[] shape, int value) ● get(String variable, int[] index)	
2	● assign(String a, String unary, String b)	
3	● assign(String a, String unary, String b, int[] parameters)	
4	● assign(String a, String b, String sBinary, String c)	

RETROSPECTIVA

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/Nombre)
   10 horas Santiago Sanchez Monroy
   10 horas Joan S. Acevedo Aguilar

2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?
   El laboratorio se encuentra finalizado en un 90%, no se realizaron las pruebas unitarias, esto es debido a que son semanas pesadas de parciales y no se pudo dedicar el tiempo deseado al laboratorio.

3. Considerando las prácticas XP del laboratorio. ¿Cuál fue la más útil? ¿por qué?
   Las pruebas unitarias. Ya que estas nos demostraron su gran importancia a la hora de probar código y lo eficaces que pueden ser.

4. ¿Cuál consideran fue el mayor logro? ¿Por qué?
   Realizar todos los métodos que se requieren para el funcionamiento de NPTensor y las pruebas de unidad, esto debido al gran esfuerzo y tiempo que requirieron.

5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?
   Las pruebas de unidad, específicamente una que comparaba el Tensor en string con otro. Para solucionarlo cambiamos un poco la prueba de unidad.

6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?
   La carga fue más equitativa esta vez, esperemos que sigan siendo así los próximos laboratorios.