Universidad Autónoma de Zacatecas

Unidad Académica de Ingeniería Eléctrica

Programa Académico de Ingeniería de Software.

Nombre de la Práctica Arreglos multidimensionales

Numero de Práctica 9

Nombre de la carrera Ingeniería de Software

Nombre de la materia Lab. Estructuras de Datos

Nombre del alumno Jesús Manuel Juárez Pasillas

Nombre del docente Aldonso Becerra Sánchez

Fecha: 06/09/2021

Práctica 9: Arreglos multidimensionales

Introducción:

Los arreglos multidimensionales pueden simular matrices con las cuales se pueden hacer todas las operaciones que requiera una matriz. Estas matrices almacenan datos numéricos por lo que será conveniente verificar que los datos que se introduzcan sean o hereden de este tipo.

Las matrices tienen una gran cantidad de utilidades con las que se pueden resolver una gran cantidad de problemas que se planteen, para resolver estos problemas es muy indispensable tener las operaciones correctas con las cuales resolver el problema.

Desarrollo:

Para esta practica se pide elaborar algunos métodos básicos con los cuales hacer operaciones básicas de matrices, como la suma y multiplicación de una matriz por un escalar, suma y multiplicación de matrices validando todos los requisitos con los cuales poder hacer estas operaciones.

Para poder realizar esto se creo una clase (Tabla2DNumeros) la cual heredara de la clase “Tabla2D”. Como esta clase contiene métodos los cuales permiten ingresar todo tipo de valores, es necesario validar que todo lo que se ingrese sea o herede de la clase “Number”, la clase padre de todos los números, esto se lograra con la sobreescritura de métodos, se valida que lo que se pasa como parámetro sea de tipo numérico usando “instanceof”.

Una vez sobrescritos los métodos necesarios, se agregarán más métodos con los cuales hacer operaciones con los datos que se almacenan. Teniendo seguro que los datos son números, las operaciones no requieren validar ningún tipo de dato, solo se concentrará en hacer las operaciones como se piden.

En la clase **Tabla2DNumeros** se agregaron los siguientes métodos (sin mencionar los sobrescritos):

- *Multiplica el escalar por cada posición de la tabla.*
  `public boolean xEscalar(Number escalar)`

- *Multiplica el escalar de cada posición por cada posición (columna) correspondiente en la tabla.*
  `public boolean xEscalares(ArregloNumeros escalares)`

- *Suma el escalar por cada posición de la tabla*
  `public boolean sumarEscalar(Number escalar)`

- *Suma el escalar de cada posición por cada posición (columna) correspondiente en la tabla.*
  `public boolean sumarEscalares(ArregloNumeros escalares)`

- *Multiplica la tabla por la tabla pasada como parámetro.*
  `public boolean multiplicar(Tabla2DNumeros tabla2)`

- *Suma la tabla por la tabla pasada como parámetro.*
  `public boolean sumar(Tabla2DNumeros tabla2)`

- *Eleva a la potencia cada indicada cada elemento de la tabla posición por posición.*
  `public boolean potenciaExE(Number escalar)`

- *Se aplica el logaritmo indicado a cada elemento de la tabla.*
  `public boolean logaritmo(TipoLogaritmo tipoLogaritmo)`

- *Agrega una diagonal con el elemento indicado.*
  `public boolean matrizDiagonal(Number contenido)`

- *Determina si la tabla es una matriz triangular superior.*
  `public boolean esDiagonalSup()`

- *Determina si la tabla es una matriz triangular inferior.*
  `public boolean esDiagonalInf()`

- *Aplica esta operación dado por A2=AA, A3=AAA, ... donde el número de A es determinado por exponente.*
  `public boolean potencia(int exponente)`

- *Dobla la matriz por columnas a la mitad, de tal forma que los elementos que sobrepasan la mitad queden sumados a los elementos que no lo hacen.*
  `public boolean doblarColumnas()`

- *Dobla la matriz por filas a la mitad, de tal forma que los elementos que sobrepasan la mitad queden sumados a los elementos que no lo hacen.*
  `public boolean doblarFilas()`

En el método “boolean logaritmo(TipoLogaritmo tipoLogaritmo)” se pide hacer un enumerado con 3 opciones, el enumerado se llamara “TipoLogaritmo” y tendrá las opciones:

- NATURAL: Si se escoge esta opción, el logaritmo se hará con base e (Euler).
- BASE10: Con esta opción el logaritmo se hará con base 10.
- BASE2: Esta opción hace el logaritmo con base 2.

Nota: Toda la documentación esta agregada en la carpeta “doc” dentro de la carpeta del proyecto (“edylab_2021_9/doc”).

Capturas del programa funcionando:

PruebaTabla2D:

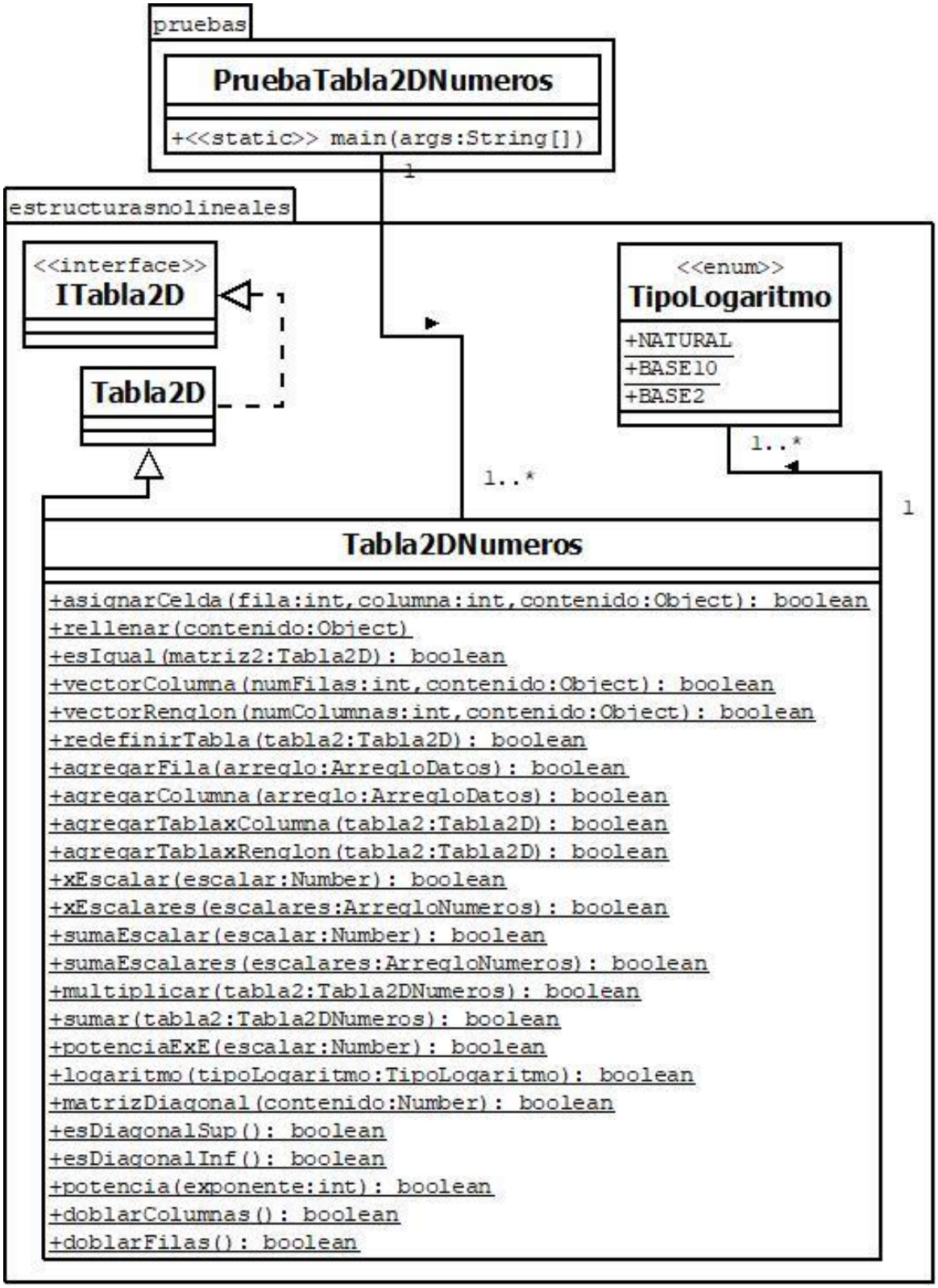
```
Por escalar (7): true
35.0 35.0 35.0 35.0
35.0 35.0 35.0 35.0
35.0 35.0 35.0 35.0
Por escalares: true
140.0 105.0 70.0 35.0
35.0 35.0 35.0 35.0
35.0 35.0 35.0 35.0
Sumar escalar (9): true
149.0 114.0 79.0 44.0
44.0 44.0 44.0 44.0
44.0 44.0 44.0 44.0
Sumar escalares: true
153.0 117.0 81.0 44.0
44.0 44.0 44.0 44.0
44.0 44.0 44.0 44.0
Multiplicar tablas: true
790.0 790.0 790.0
352.0 352.0 352.0
352.0 352.0 352.0
Sumar tablas: false
790.0 790.0 790.0
352.0 352.0 352.0
352.0 352.0 352.0
```

```
Potencia ExE (3): true
4.93039E8 4.93039E8 4.93039E8
4.3614208E7 4.3614208E7 4.3614208E7
4.3614208E7 4.3614208E7 4.3614208E7
Logaritmo natural: true
1.0986122886681098 1.0986122886681098 1.0986122886681098
1.0986122886681098 1.0986122886681098 1.0986122886681098
1.0986122886681098 1.0986122886681098 1.0986122886681098
Logaritmo base 10: true
0.47712125471966244 0.47712125471966244 0.47712125471966244
0.47712125471966244 0.47712125471966244 0.47712125471966244
0.47712125471966244 0.47712125471966244 0.47712125471966244
Logaritmo base 2: true
1.5849625007211563 1.5849625007211563 1.5849625007211563
1.5849625007211563 1.5849625007211563 1.5849625007211563
1.5849625007211563 1.5849625007211563 1.5849625007211563
Matriz diagonal (9): true
9.0 0.0 0.0
0.0 9.0 0.0
0.0 0.0 9.0
Es diagonal superior?: true
Es diagonal inferior?: true
Potencia: true
559872.0 559872.0 559872.0
559872.0 559872.0 559872.0
559872.0 559872.0 559872.0
```

```
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
Doblar filas: true
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0 5.0
Doblar columnas: true
5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0
5.0 5.0 5.0 5.0 5.0
```

Código agregado:

La clase “Tabla2DNumeros”, el enumerado “TipoLogaritmo” y “PruebaTabla2DNumeros” son nuevas, además todos los métodos subrayados también son nuevos.



Pre-evaluación:

Pre-Evaluación para prácticas de Laboratorio de Estructuras de Datos	PRE-EVALUACIÓN DEL ALUMNO
CUMPLE CON LA FUNCIONALIDAD SOLICITADA.	Sí
DISPONE DE CÓDIGO AUTO-DOCUMENTADO.	Sí
DISPONE DE CÓDIGO DOCUMENTADO A NIVEL DE CLASE Y MÉTODO.	Sí
DISPONE DE INDENTACIÓN CORRECTA.	Sí
CUMPLE LA POO.	Sí
DISPONE DE UNA FORMA FÁCIL DE UTILIZAR EL PROGRAMA PARA EL USUARIO.	Sí
DISPONE DE UN REPORTE CON FORMATO IDC.	Sí
LA INFORMACIÓN DEL REPORTE ESTÁ LIBRE DE ERRORES DE ORTOGRAFÍA.	Sí
SE ENTREGÓ EN TIEMPO Y FORMA LA PRÁCTICA.	Sí
INCLUYE LA DOCUMENTACIÓN GENERADA CON JAVADOC.	Sí
INCLUYE EL CÓDIGO AGREGADO EN FORMATO UML.	Sí
INCLUYE LAS CAPTURAS DE PANTALLA DEL PROGRAMA FUNCIONANDO.	Sí
LA PRÁCTICA ESTÁ TOTALMENTE REALIZADA (ESPECIFIQUE EL PORCENTAJE COMPLETADO).	100%
Observaciones:	

Conclusión:

El tener una matriz que solo acepte valores numéricos tiene una gran variedad de ventajas en cuanto a operaciones matemáticas que se pueden hacer con estas y no con una matriz normal que acepte cualquier valor, ya que podrían aparecer errores al no saber si se contienen solo valores numéricos. Las matrices tienen una alta gama de posibilidades como lo pude ser resolviendo distintos problemas de distintos ámbitos, como la economía, geometría, estadística, etc.