
**Universidad Autónoma de Zacatecas**

Unidad Académica de Ingeniería Eléctrica

Programa Académico de Ingeniería de Software.

**Nombre de la Práctica** Arreglos multidimensionales

**Numero de Práctica** 8

**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:** 02/09/2021

# **Práctica 8:** Arreglos multidimensionales

## Introducción:

En esta práctica se introducirá en los arreglos multidimensionales, y se crearan varios métodos que se utilizaran más delante. En esta práctica se estarán viendo los arreglos multidimensionales de 2 dimensiones (filas y columnas). Este tipo de arreglos simulan ser una matriz 2D por lo que la podemos utilizar como tal.

Estos arreglos también nos permiten almacenar una gran cantidad de elementos como los arreglos unidimensionales con la diferencia que en estos arreglos los datos se almacenan como si estuvieran en una tabla.

## Desarrollo:

La práctica pide realizar un TDA de una tabla 2D, y a esta tabla se le agregaran una serie de métodos para poder manipular los datos que se almacenaran dentro de la tabla.

En esta practica se hizo un nuevo paquete (estructurasnolineales), en el cual se agregó la clase Tabla3D que se realizó el profesor durante la clase de teoría. Todas las clases dentro de este paquete, sin incluir la clase Tabla3D son creadas para esta práctica, esto incluye dos enumerados los cuales son:

- TipoColumna: Esta clase es un enumerado que contiene 2 opciones, IZQUIERDA y DERECHA.
- TipoRenglon: Esta clase es un enumerado que contiene 2 opciones, SUPERIOR e INFERIOR.

Estos dos enumerados se van a utilizar para un método cada enumerado, los cuales consisten en eliminar una fila o columna dependiendo la opción del enumerado que se escoja.

En el paquete antes mencionado también se agregó la clase Tabla2D y su interface:

- Tabla2D: Esta clase contiene todos los métodos con los cuales controlar la tabla, los cuales son:

```
Obtiene el elemento en la posición indicada.
public Object obtenerCelda(int fila, int columna);

Evalúa si la índice esta entre el límite 0 hasta tamDimension.
private boolean enLimites(int indice, int tamDimension);

Le asigna un valor a la posición indicada.
public boolean asignarCelda(int fila, int columna, Object contenido);

Obtiene el número de filas.
public int getFilas();
```

- *Obtiene el número de columnas.*
  `public int getColumnas();`

- *Imprime renglón por renglón la matriz.*
  `public void imprimirR();`

- *Imprime columna por columna la matriz.*
  `public void imprimirC();`

- *Le aplica la transpuesta a la matriz2D.*
  `public void transpuesta();`

- *Rellena la tabla con el contenido indicado.*
  `public void rellenar(Object contenido);`

- *Generar y regresar una copia de la matriz.*
  `public Tabla2D clonar();`

- *Compara los elementos de las matrices.*
  `public boolean esIgual(Tabla2D matriz2);`

- *Genera un vector en forma de columnas con los datos especificados.*
  `public boolean vectorColumna(int numFilas, Object contenido);`

- *Genera un vector en forma de renglones con los datos especificados.*
  `public boolean vectorRenglon(int numColumnas,Object contenido);`

- *Crea/redimensiona/substituye la matriz actual por una pasada como argumento.*
  `public boolean redefinirTabla(Tabla2D tabla2);`

- *Agrega el arreglo como renglón a la matriz.*
  `public boolean agregarFila(ArregloDatos arreglo);`

- *Agrega un arreglo como columna a la matriz.*
  `public boolean agregarColumna(ArregloDatos arreglo);`

- *Agrega una matriz a la derecha de la matriz actual.*
  `public boolean agregarTablaxColumna(Tabla2D tabla2);`

- *Agrega una matriz abajo de la matriz actual.*
  `public boolean agregarTablaxRenglon(Tabla2D tabla2);`

- *Hace una matriz 3D a partir de la matriz 2D actual.*
  `public Tabla3D tabla2DaTabla3D(ArregloDatos tablas);`

- *Agrega los elementos de la matriz a un vector de una sola columna.*
  `public Tabla2D aVectorColumna();`

- *Agrega los elementos de una matriz a un vector de un solo renglón.*
  `public Tabla2D aVectorRenglon();`

- *Elimina una columna de la matriz, dependiendo de tipoCol (izquierda / derecha).*
  `public boolean quitarColumna(TipoColumna tipoCol);`

- *Elimina un renglón de la matriz, dependiendo de tipoReng (superior / inferior).*
  `public boolean quitarFila(TipoRenglon tipoReng);`

- *Elimina un renglón de la matriz.*
  `public boolean quitarFila(int indice);`

- *Elimina una columna de la matriz.*
  `public boolean quitarColumna(int indice);`

**Nota:** Toda la documentación esta agregada en la carpeta “doc” dentro de la carpeta del proyecto (“edylab_2021_8/doc”).

## Capturas del programa funcionando:

**PruebaTabla2D:**

```
Obtener 50
Cambiar true
Imprimir Renglones
50 50 50 50
50 50 50 50
50 50 50 45
Imorimir columna
50
50
50

50
50
50

50
50
50

50
50
45
```

```
Traspuesta:
50 50 50
50 50 50
50 50 50
50 50 45
Rellenar con 99:
99 99 99
99 99 99
99 99 99

Clonar:
50 50 50
50 50 50
50 50 50
50 50 45

Es igual: true
Vector columna: true
33 0 0
33 0 0
33 0 0
```

```
Vector renglon true
77 77 77
33 0 0
33 0 0

Redifinir true
99 99 99
99 99 99
99 99 99

Agregar renglon true
34 24 15
99 99 99
99 99 99
Agregar columna true
34 24 15
24 99 99
15 99 99
Agregar tabla por columna: true
50 50 50 50 50 50
50 50 50 50 50 50
50 50 50 50 50 50
50 50 45 50 50 45
```

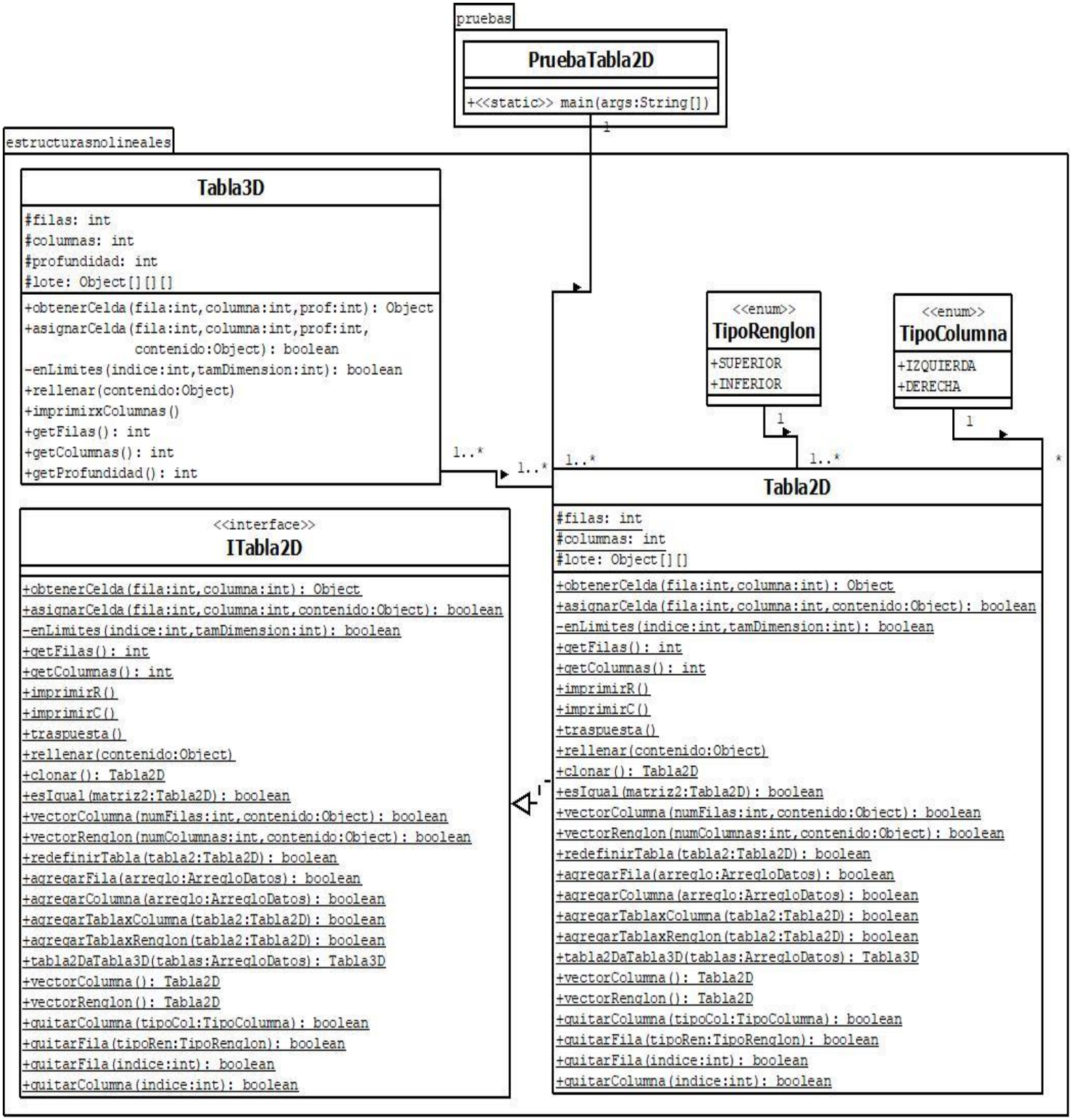
```
Agregar tabla por renglon: true
34 24 15
24 99 99
15 99 99
34 24 15
24 99 99
15 99 99

Sacar matriz 3D:
1 2 3 4
1 2 3 4
1 2 3 4

1 2 3 4
1 2 3 4
1 2 3 4

1 2 3 4
1 2 3 4
1 2 3 4
```

```
Vector columna:
34
24
15
34
24
15
24
99
99
24
99
99
15
99
99
15
99
99
Vector renglon:
50 50 50 50 50 50 50 50 50 50 50 45 50 50 50 50 50 50 50 50 50 50 45
```

```
Eliminar columna derecha: true
50 50
50 50
50 50
50 50
Eliminar renglon superior: true
50 50
50 50
50 50
Eliminar un renglon: true
Eliminar una columna: true
```

## Código agregado:

Todos los métodos subrayados son nuevos, además de todas las clases que se muestran a excepción de Tabla3D.



## 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 hacer uso de una clase con la que se puedan manejar matrices de cualquier tamaño es ideal, ya que con esta misma podemos generar vectores y no es necesario crear una clase nueva. Aunque acceder a cada elemento de la matriz es más complicado que acceder a un elemento del arreglo, porque la matriz es necesario tener las dos posiciones para encontrar un dato, en cambio el arreglo común solo necesita de una posición. Además, que estas nos permiten hacer una gran cantidad de actividades donde se guardan una gran cantidad de datos más ordenados que un arreglo común.