
**Universidad Autónoma de Zacatecas**

Unidad Académica de Ingeniería Eléctrica

Programa Académico de Ingeniería de Software.

**Nombre de la Práctica** Arreglos unidimensionales

**Numero de Práctica** 6

**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:** 30/08/2021

# **Práctica 6:** Arreglos unidimensionales

## Introducción:

La manipulación de los datos de un arreglo es muy importante para resolver una gran cantidad de problemas. Para esto los arreglos numéricos son muy importantes cuando se desea almacenar números que están relacionados entre sí. Con estos números almacenados hacer operaciones con los cuales obtener un resultado esperado.

## Desarrollo:

Para realizar esta practica se hizo uso de la herencia, ya que la clase ArregloNumeros va heredar de la clase ArregloDatos.

Para mantener la integridad de ArregloNumeros es necesario sobrescribir los métodos necesarios para que solo se almacenen números, sin importar que números, pero únicamente números. Para esto se hizo uso del instanceof con el que se comparo cada elemento pasado como parámetro a la clase Number que es la clase padre de cualquier número y con esto verificar que no se use mal esta clase.

Para comenzar se sobrescribieron los siguientes métodos en la clase ArregloNumeros:

- public int agregar(Object elemento)
- public Object buscar(Object elemento)
- public Object eliminar(Object elemento)
- public boolean esIgual(Object listaDatos2)
- public boolean cambiar(Object elementoViejo, Object elementoNuevo, int numVeces)
- public boolean cambiar(int indice, Object elemento)
- public boolean cambiarArregloDatos(ArregloDatos indicesBusqueda, ArregloDatos elementosNuevos)
- public ArregloDatos buscarValores(Object elemento)
- public boolean agregarLista(Object listaDatos2)
- public int contar(Object elemento)
- public boolean eliminarLista(Object listaDatos2)
- public void rellenar(Object elemento, int cantidad)
- public void rellenar(Object elemento)
- public boolean esSublista(Object listaDatos2)
- public boolean insertar(int indice, Object elemento)
- public boolean copiarLista(ArregloDatos listaDatos2)

Todos estos métodos contienen algún parámetro con el cual se perdería la integridad de la clase, ya sea un arreglo o un elemento, todos estos datos deben comprobar que son números y no algún otro elemento. Para esto solo se hizo una condición la cual verificara que los elementos fueran números o de tipo ArregloNumeros.

Para todos los métodos pedidos en esta practica se hizo un recorrido del arreglo actual y si era el caso también se recorrió el arreglo pasado como parámetro y había otros en los cuales el arreglo pasado como parámetro contenía arreglos por lo que estos también se recorrieron para manipular cada elemento y así obtener el resultado esperado.

Como ya se dijo solo se creo una clase la cual tendrá función nueva:

**ArregloNumeros:**

```
- Multiplica el escalar por todos los datos de la lista.
  public boolean xEscalar(Number escalar)
- Suma cada elemento del arreglo por el escalar.
  public boolean sumaEscalar(Number escalar)
- Suma la posición 1 del arreglo actual con la posición 1 de
  arreglo2, y así sucesivamente.
  public boolean sumar(ArregloNumeros arreglo2)
- multiplica la posición 1 del arreglo actual con la posición 1 de
  arreglo2, y así sucesivamente.
  public boolean multiplicar(ArregloNumeros arreglo2)
- Hace la operación de potencia por cada elemento del arreglo
  actual.
  public boolean potencia(Number escalar)
- Hace la operación de potencia de cada posición del arreglo actual
  por el elemento de la posición del arregloEscalar.
  public boolean potencia(ArregloNumeros arregloEscalar)
- suma la multiplicación de cada elemento de la lista actual por el
  arreglo2.
  public Double productoPunto(ArregloNumeros arreglo2
- Saca la magnitud / módulo / norma L2 del vector.
  public Double modulo()
- Debe calcular la norma euclidiana de los vectores numéricos
  actual y arreglo2.
  public Double normaEuclidiana(ArregloNumeros arreglo2)
- Debe sumar de uno por uno un conjunto de arreglos al arreglo
  actual.
  public Double sumarArreglosDatos(ArregloDatos arreglos)
- Suma todos los escalares al cada elemento de la posición actual
  de todos los escalares de los arreglos.
  public Double sumarEscalares(ArregloNumeros escalares)
- Debe sumar del arreglo actual las posiciones de él que indica el
  arreglo llamado arregloIndices.
  public Double sumarIndices(ArregloNumeros arregloIndices)
- Determina si el conjunto de arreglos pasados como parámetro son
  linealmente dependientes.
  public boolean sonLinealmenteDep(ArregloDatos arreglosVectores)
- Determina si el conjunto de arreglos pasados como parámetro son
  linealmente independientes.
  public boolean sonLinealmenteIdep(ArregloDatos arreglosVectores)
- Determina si el arreglo actual es ortogonal al arreglo pasado
  como argumento.
  public boolean esOrtogonal(ArregloNumeros arreglo2)
```

```
- Determina si el arreglo actual es paralelo al arreglo pasado como
  argumento.
  public boolean esParalelo(ArregloNumeros arreglo2)
```

Además, también se agregó un nuevo método a la clase ArregloDatos:

```
- Debe regresar un arreglo conteniendo los elementos del arreglo
  actual que se obtienen del arreglo de índices "arregloIndices".
  public ArregloDatos subLista(ArregloNumeros arregloIndices)
```

Para este método no fue necesario sobrescribirlo en la clase ArregloNumeros porque no contiene datos que se tengan que evaluar que sean números por lo que sirve tanto para arreglos que almacenen cualquier tipo de datos y los que solo almacenan números.

**Nota:** Toda la documentación esta agregada en la carpeta "doc" dentro de la carpeta del proyecto ("edylab_2021_6/doc").

## Capturas del programa funcionando:

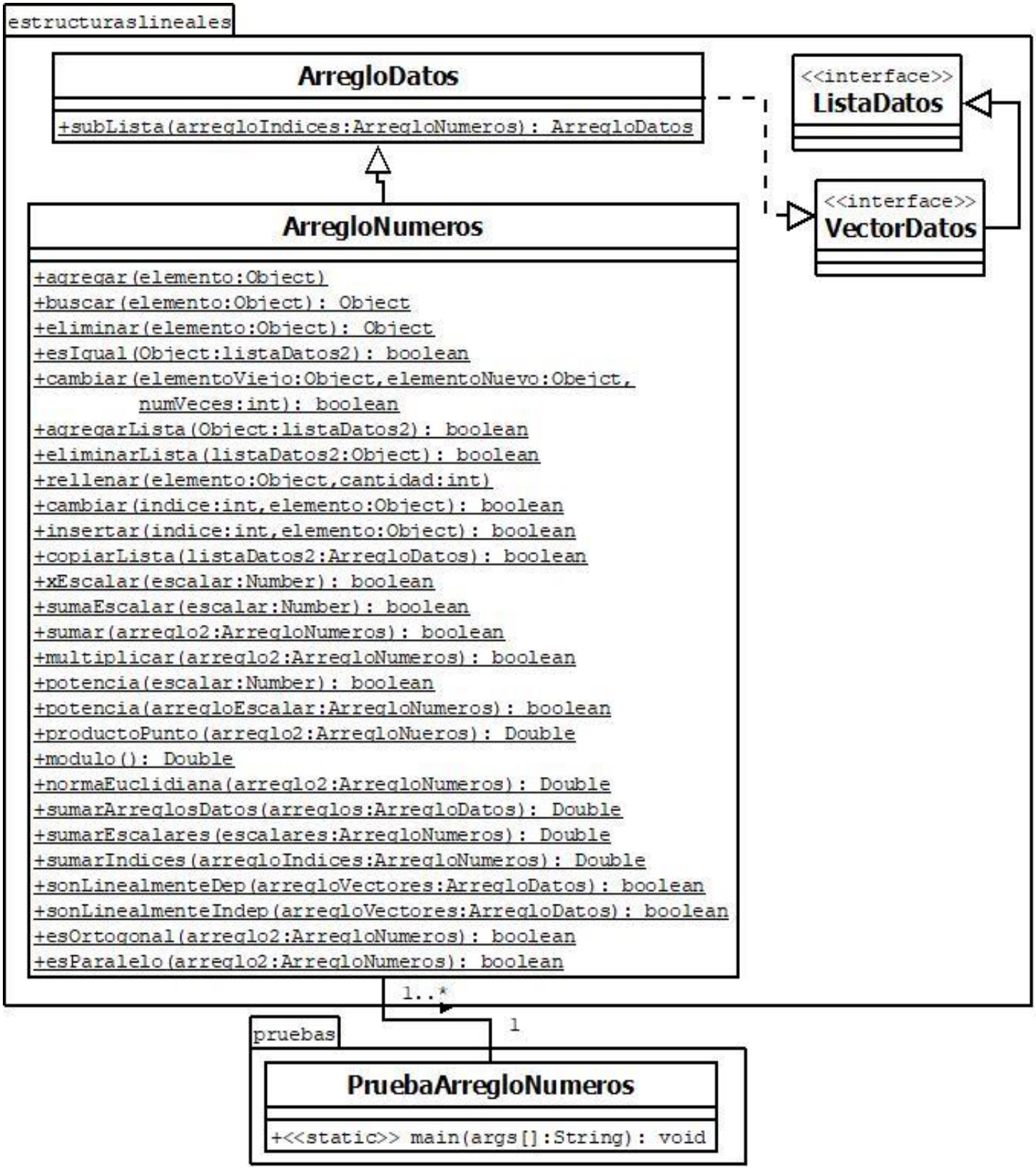
**PruebaArregloNumeros:**

```
Agregar una letra: -1
xEscalar (5.2): true
30.16
36.4
5.720000000000001
20.28
9.0
sumaEscalar (4.0): true
34.16
40.4
9.72
24.28
9.0
sumar arreglos: true
44.16
52.8
19.62
27.580000000000002
15.8
multiplicar arreglos: true
54.16
65.2
29.520000000000003
30.880000000000003
22.6
```

```
potencia a arreglo2 (2): true
100.0
153.76000000000002
98.01
10.889999999999999
46.239999999999995
Potencia arreglo2 con arreglo3: true
10000.0
3635215.0773760015
9.043820750088047E9
1.816331681783799E7
46.239999999999995
ProductoPunto: 4.535717270587723E10
10000.0
3635215.0773760015
9.043820750088047E9
1.816331681783799E7
46.239999999999995
Modulo: 9.043839719980093E9
NormaEuclidiana: 8.179103634514556E19
SumaArreglos: 9.0656295495832 61E9
SumaEscalares: 1.903782161140485E11
SumaIndices: 9.067809199104784E10
Linealmente dependientes: true
Linealmente independientes: true
Ortogonales: true
Paralelos: true
```

```
ArregloDatos:
H
D
Z
M
G
subLista:
D
M
G
```

**Código agregado:**



## 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:

Los arreglos numéricos nos sirven para solo almacenar números, y sabiendo esto se pueden resolver una gran cantidad de problemas. Los números almacenados estarán relacionados por lo que es muy importante poder realizarle operaciones a cada elemento del arreglo.

Además, poder definir la longitud de los arreglos podemos saber cuántos elementos existen dentro de esto y con base en esto poder hacer operaciones entre arreglos.