Práctica 4

Datos generales:

Nombre de la Práctica	Arreglos unidimensionales
Nombre de la carrera	Ingeniería de Software
Nombre de la materia	Estructuras de Datos
Número y nombre de Unidad(es) temática(s)	1. Introducción a las estructuras de datos y estructuras fundamentales.
Docente que imparte la materia	Aldonso Becerra Sánchez
Fecha de entrega para los alumnos	20-febrero-2025
Fecha de entrega con extensión y penalización	21-febrero-2025
Fecha de elaboración	18-febrero-2025

Objetivo de la tarea	Practicar el uso de arreglos unidimensionales en la solución de planteamientos comunes.
Tiempo aproximado de realización	2.5 horas
Introducción	Existe diversidad de usos que se les puede dar a los arreglos. Desde este punto de vista, los arreglos propician que muchos planteamientos puedan tener una solución sencilla si se llevan a cabo con la ayuda de ellos.

Referencias que debe consultar el alumno (si se requieren):

Referencia 1:

1.Cairo, Osvaldo; Guardati, Silvia. Estructura de Datos, Tercera Edición. McGraw-Hill, México, Tercera Edición, 2006.

Referencia 2:

2.Mark Allen Weiss. Estructura de datos en Java. Ed. Addison Wesley.

Referencia 3:

3. Joyanes Aguilar, Luis. Fundamentos de Programación. Algoritmos y Estructuras de Datos. Tercera Edición, 2003. McGraw – Hill.

Actividades que debe realizar el alumno:

Actividad inicial:

Lea primero toda la práctica. No inicie a programar sin leer todo cuidadosamente primero. Recuerde que debe generar el reporte en formato IDC.

Actividad 1:

Primero genere la **Introducción**.

Actividad 2:

Haga los siguientes métodos en un TDA:

Clase **ArregloNumerico**. Esta clase tiene como objetivo manejar puros valores numéricos, es decir que hereden de la clase Number (utiliza el operador instanceof para facilitar este proceso) cada valor que contendrá. Defina esta clase para que herede de **Arreglo.**
Se le pide que:

- Para ello deberá validar que el contenido que se intenta agregar, eliminar, buscar, modificar y demás operaciones heredadas de Arreglo sean de la clase Number. Para hacer eso tiene que sobre-escribir los métodos necesarios y hacer las invocaciones necesarias a la super clase. Minimice el código reescrito. Si cualquier valor proporcionado no es numérico, debe prohibir la acción del método. Se recomienda convertir todo valor numérico a double, ya que es el máximo en tamaño.
- boolean porEscalar(Number escalar): multiplicar el escalar dado por cada posición del arreglo numérico. Regrese falso si el arreglo actual está vacío.
- boolean sumarEscalar(Number escalar): sumar el escalar dado a cada posición del arreglo numérico. Regrese falso si el arreglo actual está vacío.

- boolean sumar(ArregloNumerico lista2): sumar la posición 1 del arreglo actual con la posición 1 de arreglo2, y así sucesivamente. Note que los valores pueden ser negativos, en tal caso debe seguir funcionando este proceso. Valide dimensiones.
- boolean multiplicar(ArregloNumerico lista2): que haga producto de posición 1 del arreglo actual por posición 1 de arreglo2, y así sucesivamente. Regrese falso si el arreglo actual está vacío.
- boolean aplicarPotencia(Number escalar): que haga la operación de potencia de cada elemento del arreglo (base) por el exponente pasado como escalar. Regrese falso si el arreglo actual está vacío.
- boolean aplicarPotencia(ArregloNumerico listaEscalares): que haga la operación de potencia de cada elemento del arreglo (base) por el exponente pasado como arreglo, posición por posición. Valide dimensiones.
- double productoEscalar(ArregloNumerico lista2): por ejemplo considere los vectores matemáticos (arreglos) siguientes de dimensión 3: $\{a, b, c\} \bullet \{x, y, z\}^T = a \bullet x + b \bullet y + c \bullet z$. Donde la $\{\}^T$ es la transpuesta del vector y $\bullet$ es la multiplicación elemento por elemento.
- double norma(): saca la magnitud / módulo / norma L_2 del vector (arreglo); por ejemplo considere el siguiente vector numérico de dimensión 3: $V = \{a, b, c\}$, $||V|| = \sqrt{a^2 + b^2 + c^2}$. Donde sqrt es la raíz cuadrada.
- double normaEuclidiana(ArregloNumerico arreglo2): debe calcular la norma euclidiana del vector numérico AB (arreglos n dimensionales). Este cálculo está dado por:

  $$\|\vec{AB}\| = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + \ldots + (b_n - a_n)^2}$$

  donde $A=\{a_1, a_2, a_3, \ldots, a_n\}$ y $B=\{b_1, b_2, b_3, \ldots, b_n\}$.

- boolean sumarListaEstatica(Arreglo listas): debe sumar de uno por uno (como el método de arriba pero ahora son varias listas) un conjunto de arreglos de tipo ArregloNumerico almacenados en la variable listas al arreglo actual. Cada posición de listas guarda un arreglo.
- boolean sumarEscalares(ArregloNumerico escalares): debe sumar de uno por uno un conjunto de escalares almacenados en la variable escalares al arreglo actual. Escalares es un arreglo que guarda en cada posición un escalar.
- double sumarIndices(ArregloNumerico listaIndices). Debe sumar, del arreglo actual, las posiciones de él que indica el arreglo llamado listaIndices, el cual almacena las posiciones que se deben tomar del arreglo actual para hacer la suma.
- [en la clase Arreglo] Arreglo subLista(ArregloNumerico listaIndices). Debe regresar un arreglo conteniendo los elementos del arreglo actual que se obtienen del arreglo de índices "listaIndices", el cual contiene las posiciones de los índices de donde se obtendrán los datos a retornar. Significa que se deberán hacer dos métodos, el correspondiente en la super clase (Arreglo), para datos generales y el correspondiente a la clase ArregloNumerico, correspondiente solo a números.

- Teniendo en cuenta la siguiente definición: sean v_1, v_2, ..., v_n n vectores de un espacio vectorial V. Se dice que dichos vectores son linealmente dependientes (L.D.) si existen n escalares (K) k_1, k_2, ..., k_n, no todos nulos, tales que $k_1v_1+k_2v_2+...+k_nv_n=0$. Haga un método (boolean sonLinealmenteDependientes(Arreglo listaVectores)) que determine si el conjunto de arreglos pasado como argumentos (es un arreglo de arreglos numéricos) (representa V) son linealmente dependientes. De tal forma que el arreglo actual es el conjunto de n escalares K.
- Teniendo en cuenta la siguiente definición: sean v_1, v_2, ..., v_n n vectores de un espacio vectorial V. Se dice que dichos vectores son linealmente independiente (L.I.) si la única combinación lineal de ellos que da por resultado el valor nulo es aquella que tiene todos los escalares nulos. Es decir, existen n escalares (K) k_1, k_2, ..., k_n, guardados en el arreglo actual, tal que $k_1v_1+k_2v_2+...+k_nv_n=0$ donde $k_1=k_2= ... =k_n=0$
  Haga un método (boolean sonLinealmenteIndependientes(Arreglo listaVectores)) que determine si el conjunto de arreglos pasado como argumentos (es un arreglo de arreglos numéricos) (representa V) son linealmente independientes. De tal forma que el arreglo actual es el conjunto de n escalares K.
- Dos vectores son ortogonales si y sólo si su producto escalar es nulo. Es decir: u•v=0. Genere el método boolean esOrtogonal(ArregloNumerico lista2) para determinar si el arreglo actual es ortogonal al arreglo pasado como argumento.
- Dos vectores son paralelos si y sólo si uno de ellos es múltiplo escalar del otro. Es decir: u = kv. Genere el método boolean esParalelo(ArregloNumerico lista2) para determinar si el arreglo actual es paralelo al arreglo pasado como argumento.

Esta actividad debe entrar en la parte de **Desarrollo**.

Actividad 3:

Pruebe el funcionamiento del programa de la actividad 2 con todo y sus capturas de pantalla.

Actividad 4:

Realice la sección de **Código agregado** (diagrama de clases UML).

Actividad 5:

Realice la sección de **Pre-evaluación** (use los lineamientos establecidos).

Actividad 6:

Finalmente haga las **Conclusiones**.

Actividad 5:

Enviar en http://ingsoftware.reduaz.mx/moodle

Archivo anexo que se requiere para esta tarea (opcional):

Dudas o comentarios: a7donso@gmail.com