Práctica 3

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 límite para los alumnos	13-febrero-2025
Fecha de entrega límite con extensión y penalización	14-febrero-2025
Fecha de elaboración	11-febrero-2025

Objetivo de la tarea	Crear un TDA arreglo unidimensional ordenado y desordenado para su posterior uso en aplicaciones comunes.
Tiempo aproximado de realización	6 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:

Defina el TDA llamado "ArregloOrdenado" (que herede de Arreglo, tal cual se muestra en el diagrama de clases proporcionado en sesiones anteriores), el cual debe tener atributos y métodos (los cuales deberán sobre-escribirse según sea el caso; utilice el estereotipo @Override antes del método para garantizar la correcta verificación de sobreescritura en la herencia). Recuerde que el orden en números no es lo mismo que el orden en cadenas y otros tipos de objetos; ya que debe personalizar este mecanismo según sea el tipo de contenido. Para poder hacer la práctica se ocupa un módulo comparador de objetos de cualquier tipo (pero los dos objetos deben ser del mismo tipo)

Nota: no utilice ningún método de ordenamiento formal por lotes.

Se pide que realice los siguientes métodos dentro de esta clase (deben estar enfocados para datos estrictamente ordenados, lo cual no es igual para datos desordenados):

- public ArregloOrden(int maximo, TipoOrdenamiento orden). Es el constructor del arreglo con orden. Este arreglo, además de indicar el tamaño, debe crear un enumerado en donde se indique el orden/acomodo de las inserciones, ya sea INC (1) o DEC (2) [incremental o decremental]. Ese orden se respetará en todo el conjunto de métodos.
- public Integer poner(Object valor). Insertar elementos mediante mecanismos ordenados en un orden definido en el constructor.

- public Object buscar(Object valor). Buscar elementos mediante mecanismos ordenados.
- public boolean modificar(int indice, Object valor). Este método deberá sobreescribirse para que el contenido del arreglo se reorganice.
- public Object quitar(Object valor). Eliminar un elemento del arreglo.
- public boolean agregarLista(ListaDatos lista2). Debe permitir agregar los elementos de lista2 (que debe validar que sea un arreglo [ordenado o desordenado] en este caso) en el arreglo actual. Debe reorganizar todos los elementos insertados, de tal manera que el arreglo siga ordenado. Recuerde que el arreglo ordenado no permite valores duplicados. lista2 debe ser de tipo ArregloOrdenado.
- [declarado en interface ListaDatos] public void invertir(). Debe invertir el orden de los elementos del arreglo. Al mismo tiempo que se cambia el orden, debe cambiarse el valor de la variable TipoOrdenamiento, sino mostrará inconsistencias.
- [declarado en interface ListaDatos] public void rellenar(Object valor). "valor" indica el límite superior en los valores a rellenar, siempre y cuando sea numérico. Por ejemplo, si valor es 6, se insertaron elementos del 1 al 6 (en las posiciones de 0, 1, 2, 3, 4, 5) [orden INC]. Si valor es 6 y orden es DEC, se insertaron elementos del 6 al 1 (en las posiciones de 0, 1, 2, 3, 4, 5). Si el valor es negativo, por ejemplo -6, se debe insertar elementos del -1 al -6 (en las posiciones 0, 1, 2, 3, 4, 5) [orden DEC]. Si el valor es negativo y el orden es INC, se debe insertar elementos del -6 al -1 (en las posiciones 0, 1, 2, 3, 4, 5). Si es una letra, por ejemplo H, se deberán insertar elementos como A, B, C, D, E, F, G, H (en las posiciones 0, 1, 2, 3, 4, 5….) [orden INC]. Si es una letra, por ejemplo H, se deberán insertar elementos como H, G, F, E, D, B, A (en las posiciones 0, 1, 2, 3, 4, 5….) [orden DEC]. Si es cadena, por ejemplo "hola", debe insertar solo este elemento [orden INC]. Si es otro tipo de objeto, también debe insertar solamente ese valor de manera única, ya que no se permiten valores duplicados. Recuerde que este método está en función del tamaño del arreglo, no debe sobrepasarse.
- [declarado en interface ListaDatos] public ListaDatos arregloDesordenado(). Debe regresar un arreglo desordenado, de tal manera que los elementos almacenados deben reburujarse a tal grado que ya no estén ordenados, no solo regresar un objeto tipo Arreglo.
- [declarado en interface ListaDatos] public boolean esSublista(ListaDatos lista2). Debe indicar si la lista2 (que es otro arreglo ordenado) es una sublista o subconjunto de la lista actual. Por ejemplo, la lista actual es: 1, 2, 3, 4, 5 y la lista2 es: 3, 4, 5; el valor de retorno debe ser true.
- [declarado en interface ListaDatos] public boolean modificarLista(ListaDatos lista2, ListaDatos lista2Nuevos). Debe cambiar los elementos de lista2 que se encuentren en la lista actual con los elementos de la lista2Nuevos. Cada elemento

de lista2 coincide en posición con su nuevo valor a cambiar en lista2Nuebos. Ejemplo, lista2={ 2, 3, 4}, lista2Nuevos={50, 40, 80}. Quiere decir que si encuentra un 2 en lista actual debe substituirlo por un 50, si encuentra un 3 en lista actual debe substituirlo por 40, etc.

- [declarado en interface ListaDatos] public boolean retenerLista(ListaDatos lista2). Debe dejar en la lista actual solo los elementos que se encuentran en lista2.
- El resto de los métodos de Arreglo deben sobreescribirse en blanco en ArregloOrdenado para que no estropeé la funcionalidad de la clase ordenada.

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

Actividad 3:

Complete el TDA llamado "Arreglo". Para esto deberá codificar el método:

- [declarado en interface ListaDatos] public boolean poner(int indice, Object info). Este método insertará en la posición "indice" el contenido de valor.
  - También modifique el método para sea sobre-escrito en la clase ArregloOrdenado de tal manera que se valide que siga ordenado.
- [declarado en interface ListaDatos] public boolean substituir(ListaDatos lista2). Copiar el contenido de lista2 a la lista actual. El contenido de la lista actual se perderá al ser substituido por la lista2. Este método ya lo debe tener de la práctica 1.
  - Modifique el método para sea sobre-escrito en la clase ArregloOrdenado tal manera que se valide que siga ordenado. Es decir, si tiene un orden definido en lista2, se debe poder copiar si se respeta el orden INC o DEC, en caso contrario no debe poderse hacer.

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

Actividad 4:

Defina un programa en Java que permita guardar los datos de un índice de términos/subtérminos de un libro (habitualmente vienen al final de un libro, y sirve para consultar o encontrar fácilmente términos en él), así como sus páginas donde aparece (se incluye un ejemplo en el archivo anexo). Se le pide que estos datos sean agregados en orden basado en nombre.

- Genere un menú de opciones con las siguientes consideraciones.

- Agregar términos, subtérminos y sus páginas (éstas pueden ser individuales o rangos).
- Consultar un término con base en su descripción/nombre.
- Consultar un subtérmino con base en su descripción/nombre.
- Listar todos los términos/subtérminos y sus páginas.
- Listar solo los términos de un rango de letras iniciales, por ejemplo, solo los que inicien con B y D.

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

Actividad 5:

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

Actividad 6:

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

Actividad 7:

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

Actividad final:

Finalmente haga las **Conclusiones**.

Conclusión:

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

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

Dudas o comentarios: a7donso@gmail.com