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	25-agosto-2021
Fecha de entrega con extensión y penalización	26-agosto-2021
Fecha de elaboración	25-agosto-2021

Objetivo de la tarea	Crear un TDA arreglo unidimensional ordenado y desordenado para su posterior uso en aplicaciones comunes.
Tiempo aproximado de realización	3 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 "ArregloOrden" (que herede de ArregloDatos, 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.

Nota: no utilice ningún método de ordenamiento.

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 capacidad, TipoOrden 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 ASCENDENTE (1) o DESCENDENTE (0). Ese orden se respetará en todo el conjunto de métodos.
- public int agregar(Object elemento). Insertar elementos mediante mecanismos ordenados en un orden definido en el constructor.
- public Object buscar(Object elemento). Buscar elementos mediante mecanismos ordenados.

- [declarado en interface ListaDatos] public boolean cambiar(Object elementoViejo, Object elementoNuevo, int numVeces). Este método debe reorganizar los datos después de ser modificado el valor indicado. Note que numVeces siempre tenderá a ser uno, dado que en este tipo de arreglos no se permiten valores duplicados.
- public boolean cambiar(int indice, Object elemento). Este método deberá sobreescribirse para que el contenido del arreglo se reorganice.
- public Object eliminar(Object elemento). Eliminar un elemento del arreglo.
- [declarado en interface ListaDatos] public boolean agregarLista(ListaDatos listaDatos2). Debe permitir agregar los elementos de listaDatos2 (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. ListaDatos2 es 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 TipoOrden, sino mostrará inconsistencias.
- [declarado en interface ListaDatos] public void rellenar(Object elemento). "elemento" indica el límite superior en los valores a rellenar, siempre y cuando sea numérico. Por ejemplo si elemento es 6, se insertaron elementos del 1 al 6 (en las posiciones de 0, 1, 2, 3, 4, 5) [orden ASCENDENTE]. Si elemento es 6 y orden es DESCENDENTE, se insertaron elementos del 6 al 1 (en las posiciones de 0, 1, 2, 3, 4, 5). Si el valor de elemento es negativo, por ejemplo -6, se debe insertar elementos del -1 al -6 (en las posiciones 0, 1, 2, 3, 4, 5) [orden DESCENDENTE]. Si el valor de elemento es negativo y el orden es ASCENDENTE, 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 ASCENDENTE]. 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 DESCENDENTE]. Si es cadena, por ejemplo "hola", debe insertar solo este elemento [orden ASCENDENTE]. 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.
- 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 ArregloDatos.
- [declarado en interface ListaDatos] public boolean esSublista(ListaDatos listaDatos2). Debe indicar si la listaDatos2 (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 listaDatos2 es: 3, 4, 5; el valor de retorno debe ser true.

- [declarado en interface ListaDatos] public boolean cambiarLista(ListaDatos listaDatos2, ListaDatos listaDatos2Nuevos). Debe cambiar los elementos de listaDatos2 que se encuentren en la lista actual con los elementos de la listaDatos2Nuevos. Cada elemento de listaDatos2 coincide en posición con su nuevo valor a cambiar en listaDatos2Nuebos. Ejemplo, listaDatos2={ 2, 3, 4}, listaDatos2Nuevos={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 listaDatos2). Debe dejar en la lista actual solo los elementos que se encuentran en listaDatos2.

No olvide colocar los comentarios de documentación para javadoc. Ya que el reporte IDC debe tener el anexo de la documentación.

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

Actividad 3:

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

- [declarado en interface ListaDatos] public boolean insertar(int indice, Object elemento). Este método insertará en la posición "indice" el contenido de elemento.
  - También modifique el método para sea sobre-escrito en la clase ArregloOrden de tal manera que se valide que siga ordenado.
- [declarado en interface ListaDatos] public boolean copiarLista(ListaDatos listaDatos2). Copiar el contenido de listaDatos2 a la lista actual. Debe validarse el tamaño que sea del mismo tipo. El contenido de la lista actual se perderá al ser substituido por la listaDatos2.
  - También modifique el método para sea sobre-escrito en la clase ArregloOrden tal manera que se valide que siga ordenado. Es decir, si tiene un orden definido en listaDatos2, se debe poder copiar si se respeta el orden ASCENDENTE O DESCENDENTE, en caso contrario no debe poderse hacer.

No olvide colocar los comentarios de documentación para javadoc. Ya que el reporte IDC debe tener el anexo de la documentación.

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

Actividad 4:

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

Actividad 5:

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

Actividad 6:

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