Universidad Autónoma de Zacatecas

Unidad Académica de Ingeniería Eléctrica

Programa Académico de Ingeniería de Software.

Nombre de la Práctica Aplicaciones de Árboles.

Numero de Práctica 28.

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: 08/11/2021.

Práctica 28: Aplicaciones de Árboles.

Introducción:

Las aplicaciones con árboles binarios de búsqueda nos permiten tener una gran cantidad de elementos ordenados, a los cuales podemos buscar en el árbol de una manera mas eficiente que en un árbol normal debido a su composición la cual tiene nodos mas chicos que el a la izquierda y nodos mas grandes a la derecha.

Desarrollo:

Para poder realizar esta practica se tuvo que evaluar los registros de cada tabla para encontrar un patrón en el índice que se pueda utilizar. Al final los índices que se utilizaron fueron los 4 primeros caracteres de cada línea de cada tabla (a excepción de la tabla customers la cual solo toma los primeros 3 caracteres y de la tabla categories_tab la cual su 3er columna era la adecuada para obtener el índice), ya que estos representaban el índice de la tabla. Para la tabla categories_tab se hizo uso de un método el cual separaba los elementos de las líneas con un símbolo (en este caso la coma), para así obtener el índice querido, para realizar este método se creó la clase “Separador” la cual se encuentra dentro del paquete “herramientas.texto” y esta clase contiene el método realizado.

- Separa la cadena con forme al separador especificado.
  public static ListaEncadenada separarCadena(String cadena, String separador).

Todos los índices y las direcciones se convirtieron a long para evitar problemas de longitud.

El índice y la dirección de cada registro de almaceno en un objeto llamado “NodoBusquedaBinaria”, esta clase contiene los getters y setters de los atributos índice y dirección, además también contiene el método toString el cual regresa el índice y se encuentra en el paquete “registros.basedatos”.

Para recrear la base de datos oe se hizo la clase con este nombre (OE) en la cual se agregaron 6 arboles binarios de búsqueda, los cuales cada uno representa una tabla, a cada árbol se le dio el nombre de la tabla que representara.

Para agregar los datos a los arboles se fue accediendo a cada uno de los archivos que contienen los datos y se fueron extrayendo los elementos de importancia, los cuales eran el índice y su dirección (número de carácter con el que comienza la línea en el archivo)

y con esto se fuero agregando datos de NodoBusquedaBinaria a cada árbol con sus respectivos datos de cada tabla.

Esta clase se agregó en el paquete recién creado “basedatos”, el cual esta dentro del paquete “registros” y el ella contiene los siguientes métodos los cuales cubren lo que se pide en la práctica:

- Agrega los objetos NodoBusquedaBinaria al especificada por el nombre. El índice se saca de los 5 primeros caracteres de la línea y la dirección con la posición de la línea.
  `private void leerDatos(String nombre)`
- Permite escoger una tabla y regresa el nombre de esta, además de su dirección.
  `private String escogerTabla().`
- Obtiene un dato del árbol especificado y con un índice específico.
  `private NodoBusquedaBinaria buscarDatoArbol(String tabla,Long indice).`
- Permite consultar un dato de alguna tabla en particular.
  `public String consultarDato().`
- Agrega un nuevo registro al archivo y al árbol escogidos.
  `public boolean insertarRegistro().`
- Elimina un registro del archivo y del árbol.
  `public String eliminar().`
- Modifica el archivo para eliminar el registro.
  `private String modificarArchivo(NodoBusquedaBinaria dato,String tabla).`
- Elimina la línea que se le pase como parámetro del archivo.
  `private void eliminarRegistro(String registroBorrar,String tabla).`

Los métodos privados son auxiliares para los métodos públicos los cuales son los principales. Los principales siempre piden la tabla a la cual se le va a realizar la operación y luego los datos que se requieran para realizar esta acción.

En el caso de insertar se pide el elemento completo, en este caso es necesario agregar una cadena la cual este separada por comas para cada uno de sus datos, esto permite guardar la integridad del archivo.

Finalmente se agrego una clase llamada MenuOE con la cual se hizo un menú el cual llamara a algún método elegido en este menú, el cual estará en un ciclo hasta que el usuario le dé a la opción salir (opción 4).

Nota: Toda la documentación del proyecto esta agregada en la carpeta “doc” dentro de la carpeta del proyecto (“edylab_2021_28/doc”).

Capturas del programa funcionando:

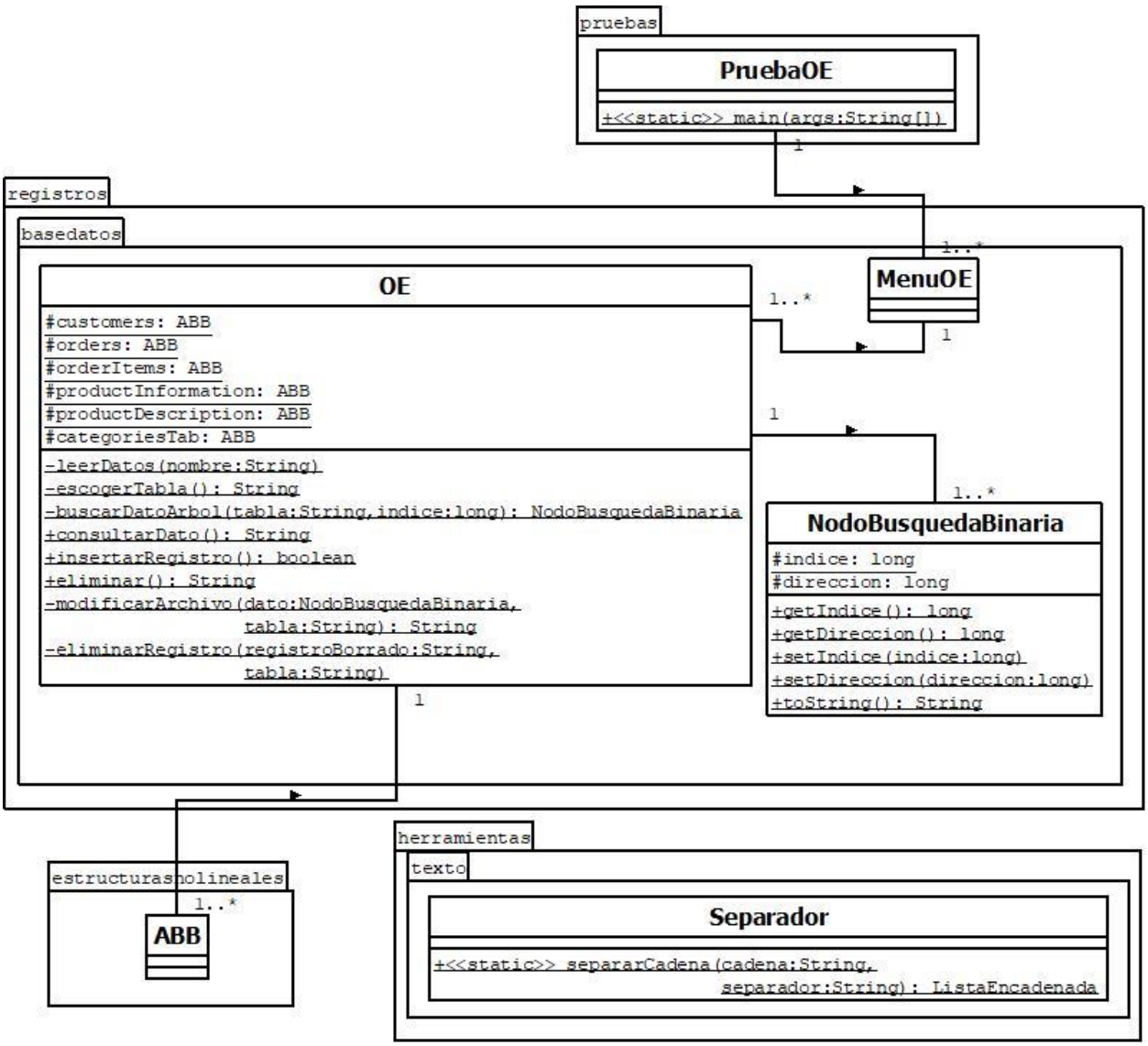
```
1.- Insertar un dato
2.- Consultar un dato
3.- Eliminar un dato
4.- Salir
Ingrese el numero de la opcion deseada: 1
1.- Customers
2.- Orders
3.- Order Items
4.- Product Information
5.- Product Description
6.- Categories Tab
Ingresa el numero de la tabla: 2
Ingresa el registro a agregar:
2458,08/11/2021  09:22:16 p. m.,direct,121,5,4567.3,155
1.- Insertar un dato
2.- Consultar un dato
3.- Eliminar un dato
4.- Salir
Ingrese el numero de la opcion deseada: 2
1.- Customers
2.- Orders
3.- Order Items
4.- Product Information
5.- Product Description
6.- Categories Tab
Ingresa el numero de la tabla: 2
```

```
Ingresa el indice a consultar: 2458
2458,08/11/2021  09:22:16 p. m.,direct,121,5,4567.3,155
1.- Insertar un dato
2.- Consultar un dato
3.- Eliminar un dato
4.- Salir
Ingrese el numero de la opcion deseada: 3
1.- Customers
2.- Orders
3.- Order Items
4.- Product Information
5.- Product Description
6.- Categories Tab
Ingresa el numero de la tabla: 2
Ingresa el indice del elemento a eliminar: 2458
Elemento eliminado: 2458,08/11/2021  09:22:16 p. m.,direct,121,5,4567.3,155
1.- Insertar un dato
2.- Consultar un dato
3.- Eliminar un dato
4.- Salir
Ingrese el numero de la opcion deseada: 2
1.- Customers
2.- Orders
3.- Order Items
4.- Product Information
5.- Product Description
6.- Categories Tab
```

```
Ingresa el numero de la tabla: 2
Ingresa el indice a consultar: 2458
Dato no encontrado
1.- Insertar un dato
2.- Consultar un dato
3.- Eliminar un dato
4.- Salir
Ingrese el numero de la opcion deseada: 4
```

Código agregado:

Todas las clases son nuevas a excepción de ABB, además de todos los métodos agregados y subrayados son nuevos, también el paquete “basedatos” es nuevo todos los demás no lo son.



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:

Usar los árboles para almacenar apuntadores a los datos reales en un archivo nos ayuda a no ocupar demasiado espacio guardando los datos completos, sino solo una referencia a estos, además de un identificador con el cual poder hacer consultas.

Hacer las consultas en un árbol tiende a ser menos complicado que en una lista ya que un árbol de búsqueda tendrá elementos ordenados siendo que en la izquierda hay elementos más pequeños y en la derecha elementos más grandes y con esto ir disminuyendo el número de nodos por visitar siempre por la mitad hasta que se encuentra o no.