40818 - Programación II

Práctica 1: Desarrollo de un contenedor lineal con una lista encadenada.

Disponible desde: lunes, 4 de febrero de 2019, 00:05
Límite de entrega: domingo, 24 de febrero de 2019, 23:55
Ficheros requeridos: ContenedorDeEnteros.java, PruebaContenedor.java, salida1.txt (Descargar)
Tamaño máximo de cada fichero de subida: 1 MiB
Tipo de trabajo: En grupo Equipo 01-44 01

Práctica 1: Desarrollo de un contenedor lineal con una lista encadenada a la que no se le exige ordenación.

Objetivos: Con la realización de esta práctica se pretende que el alumno de Programación II desarrolle un contenedor lineal en lenguaje **Java**, mediante una lista encadenada en memoria dinámica a la que no se le exige ninguna ordenación.

Interfaz: Crear una clase denominada "ContenedorDeEnteros" que representará una colección de enteros (**int**) en la que no se admitirán elementos repetidos y que pertenecerá a un paquete denominado "practica1".

La clase debe tener las siguientes operaciones:

- Un constructor por defecto que permita crear contenedores vacíos.
- Una función "cardinal" que devuelva el número de elementos del contenedor.
- Una función "insertar" que añada al contenedor un nuevo elemento pasado por parámetro, devuelve verdadero si lo añadió y falso en caso contrario.
- Una función "extraer" que extraiga del contenedor el elemento pasado por parámetro, devuelve verdadero si lo eliminó y falso en caso contrario. Si no se encuentra no se altera el contenedor.
- Una función "buscar" que devuelva verdadero si el valor pasado por parámetro pertenece al contenedor y falso en caso contrario.
- Un procedimiento "vaciar" que deje el contenedor sin ningún elemento.
- Una función "elementos" que devuelve un vector de enteros con los elementos que se encuentren en el contenedor.

Implementación: El contenedor se implementará usando como estructura interna de almacenamiento una **lista encadenada en memoria dinámica** en el que se insertarán los elementos sin precisar orden alguno. **No usar** una lista encadenada de la librería de contenedores.

Pruebas a realizar:

- Desarrolle la clase PruebaContenedor con un programa principal que compruebe el correcto funcionamiento de todas las operaciones de ContenedorDeEnteros **sin utilizar para ello JUnit**. Puede partir del contenido que se proporciona en el fichero PruebaContenedor.java, no olvide que este programa debe compilar sin modificarlo. En esta misma clase se incluirán las pruebas de rendimiento que a continuación se detallan.
  - No debe realizar ninguna entrada de datos por teclado y sólo debe mostrar errores, en caso de que existan.
  - Ficheros necesarios para la realización de las pruebas de rendimiento: datos.dat y datos_no.dat.
  - Realizará sucesivamente las inserciones de los elementos dados en el fichero datos.dat (es un fichero de acceso directo, RandomAccessFile, con cien mil números enteros distintos, int) y deberá calcular el tiempo promedio que se tarda en realizar las diez mil primeras inserciones, las diez mil segundas y así sucesivamente hasta las diez mil últimas (el tiempo promedio se ha de medir en milisegundos por cada mil inserciones).
  - Con la estructura resultante de haber realizado las inserciones comentadas en el párrafo anterior, realizará sucesivamente las extracciones de los elementos dados en el fichero datos.dat y deberá calcular el tiempo promedio que se tarda en realizar las diez mil primeras extracciones, las diez mil segundas y así sucesivamente hasta las diez mil últimas (el tiempo promedio se ha de medir en milisegundos por cada mil extracciones).
  - A fin de estudiar el comportamiento frente a la operación de búsqueda exitosa se realizará el siguiente experimento:
    - Se insertan los diez mil primeros elementos dados en el fichero datos.dat; en la estructura resultante se buscan tales elementos. Se ha de calcular el tiempo promedio de la búsqueda (en milisegundos por cada mil búsquedas).
    - Se insertan los diez mil elementos siguientes del fichero; en la estructura resultante se buscan los veinte mil elementos presentes en la estructura. Se ha de calcular el tiempo promedio de la búsqueda (en milisegundos por cada mil búsquedas).
    - Se insertan los diez mil elementos siguientes; en la estructura resultante se buscan los treinta mil elementos presentes en la estructura. Se ha de calcular el tiempo promedio de la búsqueda (en milisegundos por cada mil búsquedas).
    - Se continúa de diez mil en diez mil sucesivamente hasta haber insertado y buscado los cien mil elementos presentes en el fichero datos.dat.
  - A fin de estudiar el comportamiento frente a la operación de búsqueda infructuosa se realizará el siguiente experimento:
    - Se insertan los diez mil primeros elementos dados en el fichero datos.dat; en la estructura resultante se buscan los elementos que se encuentran en el archivo datos_no.dat (es un fichero de acceso directo, RandomAccessFile, con veinte mil números enteros, int, que no se encontrarán en la estructura). Se ha de calcular el tiempo promedio de la búsqueda (en milisegundos por cada mil búsquedas).

    - Se insertan los diez mil elementos siguientes; en la estructura resultante (que deberá tener veinte mil elementos) se buscan los elementos que se encuentran en el archivo datos_no.dat. Se ha de calcular el tiempo promedio de la búsqueda (en milisegundos por cada mil búsquedas).
    - Se continúa de diez mil en diez mil elementos sucesivamente hasta haber insertado los cien mil elementos presentes en el fichero datos.dat; en cada paso se han de buscar todos los elementos de datos_no.dat.
  - Los resultados de todos los experimentos anteriores deberán ser escritos en un fichero de texto denominado "**salida1.txt**" al objeto de realizar el informe que se comenta en el próximo punto.
  - Tanto los ficheros de entrada (datos.dat y datos_no.dat) como el de salida (salida1.txt), deben situarse en el directorio por defecto y no usar rutas absolutas en PruebaContenedor.
- A fin de cumplimentar adecuadamente el informe final de prácticas que se le solicitará cuando las haya concluido todas, debe realizar un informe en el que se incluya como mínimo lo siguiente:
  - Una gráfica del tiempo promedio de inserción en la fase creciente (en el eje X se tendrán los valores de diez mil en diez mil elementos hasta los cien mil presentes en la estructura y en el eje Y los tiempos promedios correspondientes en milisegundos por cada mil operaciones). Debería incluir lo obtenido en las prácticas anteriores.
  - Una gráfica del tiempo promedio de extracción en la fase decreciente (en el eje X se tendrán los valores de diez mil en diez mil elementos hasta los cien mil presentes en la estructura y en el eje Y los tiempos promedios correspondientes en milisegundos por cada mil operaciones). Debería incluir lo obtenido en las prácticas anteriores.
  - Una gráfica del tiempo promedio de la búsqueda exitosa para los diferentes tamaños de la estructura (en el eje X se tendrán los valores de diez mil en diez mil elementos hasta los cien mil presentes en la estructura y en el eje Y los tiempos promedios correspondientes en milisegundos por cada mil operaciones). Debería incluir lo obtenido en las prácticas anteriores.
  - Una gráfica del tiempo promedio de la búsqueda infructuosa para los diferentes tamaños de la estructura (en el eje X se tendrán los valores de diez mil en diez mil elementos hasta los cien mil presentes en la estructura y en el eje Y los tiempos promedios correspondientes en milisegundos por cada mil operaciones). Debería incluir lo obtenido en las prácticas anteriores.
  - Una comparativa entre los tiempos de inserción y extracción.
  - Una comparativa entre los tiempos de búsquedas exitosa e infructuosa.
  - Para cada gráfica dede exponerse una justificación de su contenido y de las posibles causas del comportamiento observado.

Ficheros requeridos

PruebaContenedor.java

```java
package practica1;

public class PruebaContenedor {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        ContenedorDeEnteros a = new ContenedorDeEnteros();
        int[] v;
        System.out.println("El contenedor a tiene "+a.cardinal()+" elementos.");
        for(int i=0; i<10; i++){
            a.insertar(i);
            a.buscar(i);
        }
        v = a.elementos();
        for(int i=0; i<a.cardinal(); i++) System.out.println(v[i]);
        a.vaciar();
        for(int i=0; i<100; i++){
            a.insertar(i);
            a.extraer(i);
        }
    }

}

```

VPL

Tema 10: Grafos.

Ir a... Práctica 2: Desarrollo de un contenedor lineal con un vector ordenado.

Resumen de conservación de datos
Descargar la app para dispositivos móviles

Ayuda MiULPGC

Web: Asistencia técnica
E-mail: 1234@ulpgc.es
Tlf: (+34)928 45 1234
Qué se puede solicitar

Ayuda Campus virtual

e-mail: campusvirtual@ulpgc.es
Tlf: (+34)928 45 9596
Sólo para estas páginas