Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería Campus Zacatecas

Ingeniería en Sistemas Computacionales.

Programación Orientada a Objetos

Roberto Oswaldo Cruz Leija

“Diferencia entre ArrayList y LinkedList”

Daniel Alejandro Armendáriz Rodríguez

Grupo: 2CM1

24/Octubre/2019

List

Esta interfaz también conocida como “secuencia” normalmente acepta elementos repetidos o duplicados, y al igual que los arrays es lo que se llama “basada en 0”. Esto quiere decir que el primer elemento no es el que está en la posición “1”, sino en la posición “0”.

Esta interfaz proporciona debido a su uso un iterador especial (la interfaz Iterator e Iterable las hemos podido conocer en anteriores entregas) llamada ListIterator. Este iterador permite además de los métodos definidos por cualquier iterador (recordemos que estos métodos son hasNext, next y remove) métodos para inserción de elementos y reemplazo, acceso bidireccional para recorrer la lista y un método proporcionado para obtener un iterador empezando en una posición específica de la lista.

Debido a la gran variedad y tipo de listas que puede haber con distintas características como permitir que contengan o no elementos null, o que tengan restricciones en los tipos de sus elementos, hay una gran cantidad de clases que implementan esta interfaz.

Todas las superinterfaces:

Colección <E>, Iterable <E>

ArrayList

La clase ArrayList en Java, es una clase que permite almacenar datos en memoria de forma similar a los Arrays, con la ventaja de que el número de elementos que almacena, lo hace de forma dinámica, es decir, que no es necesario declarar su tamaño como pasa con los Arrays. Para todos aquellos que hayáis estudiado en alguna asignatura las estructuras de datos de las Pilas, Colas, Listas, Arboles (AVL, B, B+, B*) etc. hay decir que los ArrayList "tiran por tierra" toda la teoria que hay detrás de esas estructuras de datos ya que los ArrayList nos permiten añadir, eliminar y modificar elementos (que pueden ser objetos o elementos atómicos) de forma trasparente para el programador.

De forma general un ArrayList en Java se crea de la siguiente forma:

```
ArrayList nombreArray = new ArrayList();
```

Esta instrucción crea el ArrayList *nombreArray* vacío.

MÉTODO	DESCRIPCIÓN
size()	Devuelve el número de elementos (int)
add(X)	Añade el objeto X al final. Devuelve true.
add(posición, X)	Inserta el objeto X en la posición indicada.
get(posicion)	Devuelve el elemento que está en la posición indicada.
remove(posicion)	Elimina el elemento que se encuentra en la posición indicada. Devuelve el elemento eliminado.
remove(X)	Elimina la primera ocurrencia del objeto X. Devuelve true si el elemento está en la lista.
clear()	Elimina todos los elementos.
set(posición, X)	Sustituye el elemento que se encuentra en la posición indicada por el objeto X. Devuelve el elemento sustituido.
contains(X)	Comprueba si la colección contiene al objeto X. Devuelve true o false.
indexOf(X)	Devuelve la posición del objeto X. Si no existe devuelve -1

Class ArrayList <E>

- java.lang.Object
  - java.util.AbstractCollection <E>
    - java.util.AbstractList <E>
      - java.util.ArrayList <E>

LinkedList

LinkedList permite eliminar e insertar elementos en tiempo constante usando iteradores, pero el acceso es secuencial por lo que encontrar un elemento toma un tiempo proporcional al tamaño de la lista.
Normalmente la complejidad de esa operación promedio sería O(n/2) sin embargo usar una lista doblemente ligada el recorrido puede ocurrir desde el principio o el final de la lista por lo tanto resulta en O(n/4).

Operación	Complejidad Promedio
`get`	O(n/4)
`add(E element)`	**O(1)**
`add(int index, E element)`	O(n/4)

Operación	Complejidad Promedio
`remove(int index)`	O(n/4)
`Iterator.remove()`	**O(1)**
`ListIterator.add(E element)`	**O(1)**

Diferencia entre LinkedLisy y ArrayList

La clase LinkedList implementa la interface List. Eso quiere decir que tendrá una serie de métodos propios de esta interface y comunes a todas las implementaciones. Así utilizando siempre que se pueda declaración de objetos del tipo definido por la interface podemos cambiar de forma relativamente fácil su implementación (por ejemplo pasar de ArrayList a LinkedList y viceversa) y conseguir mejoras en el rendimiento de nuestros programas con poco esfuerzo.

Ahora centrándonos en la clase LinkedList, ésta se basa en la implementación de listas doblemente enlazadas

La clase Linkedlist no permite posicionarse de manera absoluta (acceder directamente a un elemento de la lista) y por tanto no es conveniente para búsquedas pero en cambio sí permite una rápida inserción al inicio/final de la lista y funciona mucho más rápido que Arraylist por ejemplo para la posición 0 (imaginemos que en un ArrayList deseamos insertar en la posición 0 y tenemos muchos elementos, no solo tendremos que realizar la operación de insertar este nuevo elemento en la posición 0 sino que tendremos que desplazar el elemento que teníamos de la posición 0 a la 1, el que estaba anteriormente en la posición 1 a la 2 y así sucesivamente… Si tenemos un gran número de elementos la operación de inserción es más lenta que en la implementación LinkedList, donde el elemento simplemente “se enlaza” al principio, sin necesidad de realizar desplazamientos en cadena de todos los demás elementos).

Ventajas y desventajas

LinkedList

Ventajas	Desventajas
Añadir y remover elementos con un iterador	Uso de memoria adicional por las referencias a los elementos anterior y siguiente
Añadir y remover elementos al final de la lista	El acceso a los elementos depende del tamaño de la lista

ArrayList

Ventajas	Desventajas
Añadir elementos	Costos adicionales al añadir o remover elementos
Acceso a elementos	La cantidad de memoria considera la capacidad definida para el ArrayList, aunque no contenga elementos

ArrayList:

- Usa internamente un arreglo dinámico para almacenar los elementos.
- Proporciona una manipulación lenta
- Es la mejor opción para almacenar y acceder a datos o elementos consecutivos.

LinkedList:

- Proporciona una manipulación más rápida porque utiliza una lista doblemente enlazada.

- Se puede utilizar como lista y cola porque implementa interfaz de List, Deque y Queue.
- Es mejor para manipulación de elementos, es decir, para insertar y eliminar elementos.

Por lo tanto si necesitamos agregar elementos consecutivos y recorrerlos debemos utilizar ArrayList y sí en cambio necesitamos insertar y eliminar valores no consecutivos debemos usar LinkedList.