Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería Campus Zacatecas

ARRAYLIST VS LINKEDLIST

Programación Orientada a Objetos

José de Jesús Zapata Sánchez

2CM1

Roberto Oswaldo Cruz Leija

24/Octubre/2019

LIST:

Lista de Java es una colección ordenada. Es una interfaz que amplía la interfaz de la Colección. Proporciona control sobre la posición donde puede insertar un elemento y puede acceder a los elementos por su índice y también buscar elementos en la lista.

Puntos importantes sobre Java List son;

1. La interfaz Java List es miembro del Java Collections Framework .
2. La lista le permite agregar elementos duplicados.
3. La lista le permite tener elementos 'nulos'.
4. La interfaz de lista tiene muchos métodos predeterminados en Java 8, por ejemplo replaceAll, sort y spliterator.
5. Los índices de lista comienzan desde 0, al igual que las matrices.
6. List admite Generics y deberíamos usarlo siempre que sea posible. El uso de Generics with List evitará ClassCastException en tiempo de ejecución.

Métodos de la Lista Java son;

1. *int size ():* Para obtener el número de elementos en la lista.
2. *boolean isEmpty ():* Para verificar si la lista está vacía o no.
3. *boolean contiene (Objeto o):* Devuelve verdadero si esta lista contiene el elemento especificado.
4. *Iterator <E> iterator ():* Devuelve un iterador sobre los elementos de esta lista en la secuencia adecuada.
5. *Object [] toArray () :* Devuelve una matriz que contiene todos los elementos de esta lista en la secuencia adecuada
6. *add boolean (E e):* Agrega el elemento especificado al final de esta lista.
7. *remove booleano (objeto o):* Elimina la primera aparición del elemento especificado de esta lista.
8. *boolean retiene Todos (Colección <?> c):* Retiene solo los elementos de esta lista que están contenidos en la colección especificada.
9. *void clear ():* Elimina todos los elementos de la lista.
10. *E get (int index):* Devuelve el elemento en la posición especificada en la lista.
11. *E set (int index, E element):* Reemplaza el elemento en la posición especificada en la lista con el elemento especificado.
12. *ListIterator <E> listIterator ():* Devuelve un iterador de lista sobre los elementos en la lista.
13. *Lista <E> subList (int fromIndex, int toIndex):* Devuelve una vista de la parte de esta lista entre el fromIndex especificado, inclusive, y toIndex, exclusivo. La lista devuelta está respaldada por esta lista, por lo que los cambios no estructurales en la lista devuelta se reflejan en esta lista, y viceversa.

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.

Las estructuras de datos de las Pilas, Colas, Listas, Arboles, los ArrayList ignoran toda la teoría 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.

Algo importante a la hora de trabajar con los ArrayList son los "Iteradores" (Iterator). Los Iteradores sirven para recorrer los ArrayList y poder trabajar con ellos. Los Iteradores solo tienen 3 métodos que son:

1. hasNext(): Para comprobar que siguen quedando elementos en el iterador,
2. next(): Para que nos dé el siguiente elemento del iterador
3. remove(): Sirve para eliminar el elemento del Iterador.

Principales Métodos:

1. *Declaración de un ArrayList de "String". Puede ser de cualquier otro Elemento u Objeto (float, Boolean, Object,...)*

```
ArrayList<String> nombreArrayList = new ArrayList<String>();
```

2. *Añade el elemento al ArrayList*

```
nombreArrayList.add("Elemento");
```

3. *Añade el elemento al ArrayList en la posición 'n'*

```
nombreArrayList.add(n, "Elemento 2");
```

4. *Devuelve el número de elementos del ArrayList*

```
nombreArrayList.size();
```

5. *Devuelve el elemento que está en la posición '2' del ArrayList*

```
nombreArrayList.get(2);
```

6. *Comprueba se existe del elemento ('Elemento') que se le pasa como parámetro*

```
nombreArrayList.contains("Elemento");
```

7. *Devuelve la posición de la primera ocurrencia ('Elemento') en el ArrayList*

```
nombreArrayList.indexOf("Elemento");
```

8. *Devuelve la posición de la última ocurrencia ('Elemento') en el ArrayList*

```
nombreArrayList.lastIndexOf("Elemento");
```

9. *Borra el elemento de la posición '5' del ArrayList*

```
nombreArrayList.remove(5);
```

10. *Borra la primera ocurrencia del 'Elemento' que se le pasa como parámetro.*

```
nombreArrayList.remove("Elemento");
```

11. *Borra todos los elementos de ArrayList*

```
nombreArrayList.clear();
```

12. *Devuelve True si el ArrayList esta vacío. Sino Devuelve False*

```
nombreArrayList.isEmpty();
```

13. *Copiar un ArrayList*

```
ArrayList arrayListCopia = (ArrayList) nombreArrayList.clone();
```

14. *Pasa el ArrayList a un Array*

```
Object[] array = nombreArrayList.toArray();
```

LINKEDLIST:

Son estructuras de datos lineales donde los elementos no se almacenan en ubicaciones contiguas y cada elemento es un objeto separado con una parte de datos y una parte de dirección. Los elementos están vinculados mediante punteros y direcciones. Cada elemento se conoce como un nodo. Debido a la dinámica y la facilidad de las inserciones y eliminaciones, son preferibles a las matrices.

Constructores para Java LinkedList:

LinkedList (): se utiliza para crear una lista vinculada vacía.

LinkedList (Colección C): se utiliza para crear una lista ordenada que contiene todos los elementos de una colección específica, tal como la devuelve el iterador de la colección.

Métodos para LinkedList:

add (int index, E element): este método inserta el elemento especificado en la posición especificada en esta lista.

add (E e): este método agrega el elemento especificado al final de esta lista.

addAll (int index, Collection C): Este método inserta todos los elementos de la colección especificada en esta lista, comenzando en la posición especificada.

addAll (Colección C): Este método agrega todos los elementos de la colección especificada al final de esta lista, en el orden en que los devuelve el iterador de la colección especificada.

addFirst (E e): este método inserta el elemento especificado al principio de esta lista.

addLast (E e): este método agrega el elemento especificado al final de esta lista.

clear (): este método elimina todos los elementos de esta lista.

clone (): este método devuelve una copia superficial de esta LinkedList.

contiene (Objeto o): este método devuelve verdadero si esta lista contiene el elemento especificado.

descendingIterator (): este método devuelve un iterador sobre los elementos de esta deque en orden secuencial inverso.

element (): este método recupera, pero no elimina, el encabezado (primer elemento) de esta lista.

get (int index) : este método devuelve el elemento en la posición especificada en esta lista.

getFirst (): este método devuelve el primer elemento de esta lista.

getLast (): este método devuelve el último elemento de esta lista.

indexOf (Object o): este método devuelve el índice de la primera aparición del elemento especificado en esta lista, o -1 si esta lista no contiene el elemento.

lastIndexOf (Object o): este método devuelve el índice de la última aparición del elemento especificado en esta lista, o -1 si esta lista no contiene el elemento.

listIterator (int index): este método devuelve un iterador de lista de los elementos de esta lista (en la secuencia adecuada), comenzando en la posición especificada en la lista.

oferta (E e): este método agrega el elemento especificado como la cola (último elemento) de esta lista.

offerFirst (E e): este método inserta el elemento especificado al principio de esta lista.

offerLast (E e): este método inserta el elemento especificado al final de esta lista.

peek (): este método recupera, pero no elimina, el encabezado (primer elemento) de esta lista.

peekFirst (): este método recupera, pero no elimina, el primer elemento de esta lista, o devuelve nulo si esta lista está vacía.

peekLast (): este método recupera, pero no elimina, el último elemento de esta lista, o devuelve nulo si esta lista está vacía.

poll (): este método recupera y elimina el encabezado (primer elemento) de esta lista.

pollFirst (): este método recupera y elimina el primer elemento de esta lista, o devuelve nulo si esta lista está vacía.

pollLast (): este método recupera y elimina el último elemento de esta lista, o devuelve nulo si esta lista está vacía.

pop (): este método muestra un elemento de la pila representada por esta lista.

push (E e): este método empuja un elemento en la pila representada por esta lista.

remove (): este método recupera y elimina el encabezado (primer elemento) de esta lista.

remove (int index): este método elimina el elemento en la posición especificada en esta lista.

remove (Object(): oeste método elimina la primera aparición del elemento especificado de esta lista, si está presente.

removeFirst (): este método elimina y devuelve el primer elemento de esta lista.

removeFirstOccurrence (Object o): este método elimina la primera aparición del elemento especificado en esta lista (al recorrer la lista de principio a fin).

removeLast (): este método elimina y devuelve el último elemento de esta lista.

removeLastOccurrence (Object o): este método elimina la última aparición del elemento especificado en esta lista (al recorrer la lista de principio a fin).

set (int index, E element): este método reemplaza el elemento en la posición especificada en esta lista con el elemento especificado.

size (): este método devuelve el número de elementos en esta lista.

spliterator (): este método crea un Spliterator de enlace tardío y rápido a prueba de fallos sobre los elementos de esta lista.

toArray (): este método devuelve una matriz que contiene todos los elementos de esta lista en la secuencia adecuada (del primer al último elemento).

toArray (T [] a): este método devuelve una matriz que contiene todos los elementos de esta lista en la secuencia adecuada (del primer al último elemento); El tipo de tiempo de ejecución de la matriz devuelta es el de la matriz especificada.

DIFERENCIAS ENTRE ARRAYLIST Y LINKEDLIST

LinkedList usa internamente una lista doblemente ligada, mientras que ArrayList usa un arreglo que cambia de tamaño dinámicamente.

LinkedList permite eliminar e insertar elementos usando iteradores, pero el acceso es secuencial por lo que encontrar un elemento toma un tiempo proporcional al tamaño de la lista.

ArrayList ofrece acceso en tiempo constante, pero si quieres añadir o remover un elemento en cualquier posición que no sea la última es necesario mover elementos