ArrayList en Java

En las guías anteriores trabajamos con los arreglos unidimensionales y bidimensionales, en esta ocasión vamos a continuar con las estructuras de datos pero ahora conociendo las listas o ArrayList en Java.

Si venimos de lenguajes como python, las listas que manejamos allí, son el equivalente a los ArrayList en java, es más, como se mencionó en una guía anterior, la clase ArrayList implementa la interface List y trae consigo una serie de funcionalidades que podemos utilizar.

Algo también importante a tener en cuenta es que a diferencia de los arreglos, los ArrayList son mutables, es decir, si pueden cambiar su valor o tamaño de las mismas, estas estructuras tienen un crecimiento dinámico ya que no depende de un tamaño fijo como si lo tienen los arreglos, por lo que podemos crearlas y agregar elementos en cualquier momento, otra diferencia es que las listas o en general las colecciones no permiten el trabajo con datos primitivos, por lo que cada elemento dentro de las mismas serán objetos, así, si queremos agregar un dato entero por ejemplo, no usamos int sino Integer.

En el siguiente ejemplo creamos una lista que recibe valores de diferentes tipos y vemos como se puede imprimir cada elemento por separado o uno a uno mediante ciclos

```
import java.util.ArrayList;// Se debe importar la clase ArrayList

public class Principal {

  public static void main(String[] args) {
    // Creación de una lista de enteros
    ArrayList<Integer> lista = new ArrayList<Integer>();// El Integer en el new es opcional
    //Asignamos los diferentes valores mediante el método add()
    lista.add(1);
    lista.add(4);
    lista.add(6);
    lista.add(11);
    System.out.println("Contenido de la lista: " + lista);
    System.out.println("Elemento en la posición 0 = " + lista.get(0));
    System.out.println("Elemento en la posición 1 = " + lista.get(1));
    System.out.println("Elemento en la posición 2 = " + lista.get(2));
    System.out.println("Elemento en la posición 3 = " + lista.get(3));

    // Si queremos saber el tamaño, usamos la función .size()
    System.out.println("Tamaño: " + lista.size()); // Imprime tamaño 4
    System.out.println();

    // Podemos recorrer la lista con un ciclo while llegando hasta el
    // tamaño de la misma y definiendo el indice con i
    System.out.println("Elementos recorridos con el While");
    int i = 0;
    while (i < lista.size()) {
      System.out.println(lista.get(i));
      i++;
    }

    System.out.println();

    System.out.println("Elementos recorridos con el For");
    for (int j = 0; j < lista.size(); j++) {
      System.out.println(lista.get(j));
    }

  }

}
```

En el siguiente ejemplo vemos algunas operaciones con un ArrayList de nombres, en este caso realizamos la búsqueda del nombre Juan para determinar si se encuentra dentro de la lista, este proceso lo hacemos con la forma 1 mediante el recorrido y consulta de cada elemento o con la forma 2 validándolo directamente a la estructura de datos.

```
import java.util.ArrayList;

public class Principal {
```

```
  public static void main(String[] args) {
    ArrayList<String> listaNombres=new ArrayList<String>();
    listaNombres.add("Andrea");
    listaNombres.add("Viky");
    listaNombres.add("Cris");
    listaNombres.add("Diego");
    listaNombres.add("Oscar");
    listaNombres.add("Felipe");
    listaNombres.add("Juan");

    System.out.println();
    System.out.println(listaNombres);
    String nombre="Juan";
    //Con esta forma recorremos cada elemento de la lista preguntando por cada posición.
    System.out.println("Forma1");
    for (int i = 0; i < listaNombres.size(); i++) {
      if (nombre==listaNombres.get(i)) {
        System.out.println("El nombre "+nombre+" se encuentra en la posición "+i);
       }
    }

    //Con esta forma validamos directamente usando "contains"
    //y usando la función indexOf obtenemos el indice del mismo
    System.out.println("\nForma2");

    if (listaNombres.contains(nombre))
      System.out.println("El nombre "+nombre+" si está, se "
          + "encuentra en la pos "+listaNombres.indexOf(nombre));
    else
      System.out.println("El nombre No está en la lista");

  }

}
```

Agregar Elementos al final: Como se mencionó las listas son mutables y tienen un crecimiento dinámico, esto quiere decir que podemos agregar valores después de creada, esto lo hacemos como ya vimos mediante la función add(valor)

```
ArrayList<Integer> lista=new ArrayList<Integer>();
lista.add(10); lista.add(20); lista.add(30);

System.out.println(lista.size()); // imprime un 3
lista.add(100);    // Agregamos el valor 100
System.out.println(lista.size());   // imprime un 4
System.out.println(lista.get(0));   // imprime un 10
System.out.println(lista.get(3));   // imprime un 100
```

Agregar Elementos en un indice definido: También se pueden agregar elementos a la lista en un indice definido por el usuario, para esto hacemos uso de la función add(indice,valor) a la que le enviamos la posición y el valor a asignar.

```
ArrayList<Integer> lista=new ArrayList<Integer>();
lista.add(10);
lista.add(20);
lista.add(30);
System.out.println(lista);   // imprime [10, 20, 30]
lista.add(2, 5);
System.out.println(lista);   // imprime [10, 20, 5, 30]
```

Eliminar Elementos: otra función útil dentro de las listas es la posibilidad de remover elementos, esto lo hacemos con la función Remove() a la que le agregamos el elemento a remover

```
ArrayList<Integer> lista=new ArrayList<Integer>();
lista.add(10);
lista.add(20);
lista.add(30);
lista.add(40);
lista.add(50);
lista.add(60);
System.out.println(lista);   // imprime [10, 20, 30, 40, 50, 60]
lista.remove(3);
System.out.println(lista);   // imprime [10, 20, 30, 50, 60]
```

Validar si tiene o no contenidos: Otra función que podemos usar, es la función isEmpty() que nos permitirá saber si la lista tiene o no tiene contenido (también podríamos saberlo preguntando si el tamaño es mayor a 0)

```
ArrayList<Integer> lista=new ArrayList<Integer>();

if (lista.isEmpty()) {
  System.out.println("Está vacia");
}
else {
  System.out.println("Está llena");
}
```

Limpiar Lista: Si queremos limpiar el contenido de la lista podemos usar la función clear() la cual elimina todos los elementos de la lista

```
ArrayList<Integer> lista=new ArrayList<Integer>();
lista.add(10);
lista.add(20);
lista.add(30);
lista.add(40);
lista.add(50);
lista.add(60);
System.out.println(lista);   // imprime [10, 20, 30, 40, 50, 60]
lista.clear();
System.out.println(lista);   // imprime []
```

Como ya se mencionó, se pueden recorrer las listas mediante ciclos, siendo el ciclo for el más optimo para este caso, dado que la lista tiene un índice definido, sin embargo java nos brinda una variante del for ideal para el trabajo con listas, a este for se le conoce como el foreach y posee una estructura diferente, veamos su aplicación

```
ArrayList<Integer> lista=new ArrayList<Integer>();
lista.add(10);
lista.add(20);
lista.add(30);
lista.add(40);
lista.add(50);
lista.add(60);

System.out.println("Impresión con for normal");
for (int i = 0; i < lista.size(); i++) {
  System.out.print(lista.get(i)+" ");
}

System.out.println();


System.out.println("Impresión con foreach");
for (Integer elemento : lista) {
  System.out.print(elemento+" ");
}
```

Instructor: Cristian David Henao Hoyos