Otros Métodos utilizados en ArrayList

¡Comencemos!

¿Qué aprenderás?

- Aplicar métodos importantes como size(), sort(), entre otros, para manejar fácilmente volúmenes de información.

Introducción

Aplicaremos nuevos métodos de un ArrayList para utilizar buenas prácticas de programación, agilizando el tiempo y el código dentro del programa.

Cada método que aplicaremos permitirá el mejor desempeño del estudiante a la hora de abordar casos complejos con volúmenes grandes de información. Para esto es necesario aplicar y reforzar métodos ya aprendidos en conjunto con los nuevos, porque los ArrayList nos disponibiliza métodos para cada caso.

¡Vamos con todo!

Reemplazar elementos según índice

Usaremos el método `set(int index, E element)` para reemplazar un elemento del arreglo. Este método retorna el elemento previo en dicha posición.

```
ArrayList<String> a = new ArrayList <String>();
a.add("a");
a.add("b");
a.add("c");
a.add("d");
System.out.println(a); //[a, b, c, d]
a.set(1, "k");
System.out.println(a); //[a, k, c, d]
```

```
String elementoCambiado = a.set(0,"j");
System.out.println("elemento cambiado" + elementoCambiado);
elemento cambiado a
```

Contar elementos del `ArrayList` con el método `size()`

Podemos contar la cantidad de elementos de un array con el método `size()`. Cuando no existen elementos en el arreglo, su valor es cero.

```
ArrayList<String> a = new ArrayList <String>();
a.add("a");
a.add("b");
a.add("c");
a.add("d");
System.out.println(a.size()); //4
```

Importar ArrayList

En Java podemos aplicar otros métodos sobre los ArrayList que permiten facilitar el trabajo al momento de operar sobre los elementos. Para estas operaciones utilizaremos la librería.

```
import java.util.ArrayList;
```

Ordenar los elementos con el método sort()

Nos permite ordenar la lista de manera ascendente. En el caso de los Strings, los ordena comparando carácter a carácter:

```
ArrayList<String> paises = new ArrayList<String>();
paises.add("Chile");
paises.add("Argentina");
paises.add("Colombia");
paises.add("Perú");
paises.add("Venezuela");
Collections.sort(paises);
System.out.println(paises); //[Argentina, Chile, Colombia, Perú,
Venezuela]
```

Al ordenarse quedan en orden alfabético:

```
[Argentina, Chile, Colombia, Perú, Venezuela]
```

Agregaremos un nuevo elemento a la lista, pero con minúscula la primera letra y llamaremos al método `sort()`.

```
paises.add("chile");
Collections.sort(paises);
System.out.println(paises);
```

```
[Argentina, Chile, Colombia, Perú, Venezuela, chile]
```

Podemos ver que “chile”, luego de ordenar el arreglo, queda al final cuando debería haber quedado en la posición 1. Para solucionar esto escribiremos lo siguiente:

```
Collections.sort(paises,String.CASE_INSENSITIVE_ORDER);
System.out.println(paises);
```

```
[Argentina, Chile, chile, Colombia, Perú, Venezuela]
```

¿Y si ahora queremos tener el orden descendente?

Invertir los elementos del ArrayList con el método `reverse()`.

Para invertir una lista, existe el método `reverse()` que se encarga de invertir una ArrayList.

```
Collections.reverse(paises);
System.out.println(paises);
```

```
[Venezuela, Perú, Colombia, chile, Chile, Argentina]
```

Obtener el mínimo y máximo valor del ArrayList

- `min ()`: Retorna el valor mínimo dentro del ArrayList.
- `max()`: Retorna el valor máximo dentro del ArrayList.

```
ArrayList<Integer> numeros = new ArrayList<Integer>();
numeros.add(5);
numeros.add(1);
numeros.add(4);
numeros.add(1);
numeros.add(2);
numeros.add(6);
System.out.println(Collections.min(numeros)); //1
System.out.println(Collections.max(numeros)); //6
```

Obtener la frecuencia de un elemento en el ArrayList

Si queremos saber cuántas veces existe la ocurrencia de un elemento en el ArrayList, podemos usar el método `frequency()`.

```
System.out.println(Collections.frequency(numeros, 1)); //2
```

Ejercicio guiado: Lista de platos

Crear un método llamado “ordenar” que nos permita ordenar alfabéticamente una lista de platos de un restaurante, también se debe mostrar lista ordenada por pantalla.

Esta lista cuenta con los siguientes datos:

- Cazuela.
- Porotos.
- Pastel de Choclo.
- Ají de gallina.
- Ceviche.
- Arepas.

Paso 1: Creamos el método estático llamado ordenar.

```
public static void ordenar() {
}
```

Paso 2: Se crea una variable local de tipo ArrayList llamada lista.

```
public static void ordenar() {
   ArrayList<String> lista = new ArrayList<String>();
}
```

Paso 3: Agregamos cada elemento al arreglo.

```
public static void ordenar() {
   ArrayList<String> lista = new ArrayList<String>();
   lista.add("Cazuela");
   lista.add("Porotos");
   lista.add("Pastel de Choclo");
   lista.add("Ají de Gallina");
   lista.add("Ceviche");
   lista.add("Arepas");
}
```

Paso 4: Utilizamos el método sort para ordenar la lista.

```
Collections.sort(lista);
```

Paso 5: Se muestra por consola la lista.

```
System.out.println("La lista de comida es " + lista);
```

La solución completa quedaría así:

```
	public static void main(String[] args) {
		ordenar();
	}

	// Paso 1
	public  static  void ordenar() {

		// Paso 2
		ArrayList<String> lista = new ArrayList<String>();

		// Paso 3
		lista.add("Cazuela");
		lista.add("Porotos");
		lista.add("Pastel de Choclo");
		lista.add("Aji de Gallina");
		lista.add("Ceviche");
		lista.add("Arepas");

		// Paso 4
		Collections.sort(lista);
		// Paso 5
		System.out.println("La lista de comida es " + lista);
	}
```