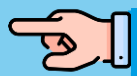
Arreglos y archivos

Operaciones básicas en un array dinámico

{desafío} latam_

Utilizar la sintaxis básica del lenguaje Java para la construcción de programas que resuelven un problema de baja complejidad

- Unidad 1: Flujo, ciclos y métodos
- Unidad 2: Arreglos y archivos
- Unidad 3: Programación orientada a objetos
- Unidad 4: Pruebas unitarias y TDD



Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Comprender la documentación de la clase ArrayList para hacer uso de sus métodos y codificar de manera rápida.*
- *Aplicar operaciones de un array dinámico para “agregar”, “eliminar”, “ordenar” y “contar” para conocer los métodos esenciales de un ArrayList.*

¿Qué entendemos por arreglos dinámicos?

/* Clase ArrayList */

Comprender e interpretar la documentación

1. Ver a qué módulo y paquete pertenece la clase ArrayList, cómo se define una descripción, entre otros.

1. Definir los constructores y todos los métodos que tiene la clase.

1. Detallar cada uno de sus constructores y métodos:

Agregar un elemento

Modificador y tipo	Método y descripción
boolean	`add(E e)` Agrega un elemento específico al final de la lista.
void	`add(int index, E element)` Inserta un elemento específico en la posición indicada.

Agregar un elemento

El método principal para agregar elementos a una Arraylist es add, a medida que se vayan agregando, el índice va aumentando de forma secuencial, partiendo como valor inicial en cero.

```
ArrayList <String> a = new ArrayList <String> ();
a.add(1);
a.add(2);
a.add(3);
```

```
System.out.println(a);
```

```
[1,2,3] // el ArrayList permite mostrar todos sus elementos sin recorrer.
```

Ejercicio guiado

Agregar elementos a un arreglo

Dado un arreglo llamado “ingredientes” se nos pide crear un programa donde el usuario pueda consultar si un ingrediente existe en la pizza, y si no existe debe ser añadido a la lista de ingredientes.

```
ingredientes // [piña, jamón, salsa, queso]
```

Agregar elementos a un arreglo

PASO 1:
Importar la clase ArrayList

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

Agregar elementos a un arreglo

PASO 2:
Crear un ArrayList de tipo String llamado “ingredientes” el cual se le agrega los valores por defectos citados en el ejercicio.

```
ArrayList<String> ingredientes = new ArrayList<String> ();
ingredientes.add("piña");
ingredientes.add("jamón");
ingredientes.add("salsa");
ingredientes.add("queso");
```

Agregar elementos a un arreglo

PASO 3:
Crear un objeto Scanner el cual nos permitirá leer los datos ingresados por consola.

```
Scanner sc = new Scanner(System.in);
String ingrediente = sc.nextLine();
```

Agregar elementos a un arreglo

PASO 4:
En la condición if, preguntar si el ingrediente ingresado está en el arreglo, si esto existe, mostramos por consola el mensaje correspondiente.

```
if(ingredientes.contains(ingrediente)){
     System.out.printf("El ingrediente ya se encuentra dentro de la pizza\n");
}
```

Agregar elementos a un arreglo

PASO 5:
Si el ingrediente no existe dentro del arreglo se mostrará el mensaje correspondiente más el arreglo completo.

```
else {
     ingredientes.add(ingrediente);
     System.out.printf("El ingrediente %s fue agregado\n",ingrediente);
}
System.out.println(ingredientes);
```

Agregar elementos a un arreglo

Solución completa

```
import java.util.ArrayList; // Paso 1
// Paso 2
ArrayList<String> ingredientes = new ArrayList<String> ();
ingredientes.add("piña");
ingredientes.add("jamón");
ingredientes.add("salsa");
ingredientes.add("queso");
// Paso 3
Scanner sc = new Scanner(System.in);
String ingrediente = sc.nextLine();
if(ingredientes.contains(ingrediente)){ // Paso 4
      System.out.printf("El ingrediente ya se encuentra dentro de la pizza\n");
}
else { // Paso 5
      ingredientes.add(ingrediente);
      System.out.printf("El ingrediente %s fue agregado\n",ingrediente);
}
System.out.println(ingredientes);
```

Agregar elementos a un arreglo

Si agregamos el elemento "champiñon", esta sería la salida:

```
El ingrediente champiñón fue agregado
[piña, jamón, salsa, queso, champiñón]
```

/* Remover elementos */

Eliminar todos los elementos de un arreglo

Método clear();

```
// Crear arreglo nombres
ArrayList<String> nombres = new ArrayList<String> ();
// Añadir “Juan” al arreglo nombres
nombres.add("Juan");
// Eliminar el arreglo nombres
nombres.clear();
//Imprimir mensaje de salida
System.out.println(" Valores en el arreglo" + nombres);
```

```
Valores en el arreglo []
```

Eliminar elemento según índice

Método remove(int index);

```
ArrayList<String> nombres = new ArrayList <String>();
nombres.add("Juan");
nombres.add("Pedro");
nombres.add("Luis");
nombres.remove(1); // "Pedro"
System.out.println(nombres);
```

Cuando realizamos esta operación, el valor de retorno del método es el valor eliminado, por lo que podríamos hacer lo siguiente: ***Eliminar el elemento que coincida con el valor entregado***.

Eliminar elemento según índice

¿Qué pasa ahora si entregamos el valor del elemento que queremos eliminar?

Si queremos eliminar un elemento, por ejemplo, el valor "a".
En este caso, el método `remove()` nos retornará true o false, si se hizo o no la eliminación.

```
ArrayList<String> a = new ArrayList <String>();
a.add("a");
a.add("b");
a.add("c");
a.add("d");
System.out.println(a);
String borrado = a.remove(1); // "b"
System.out.println(a);
System.out.println("Elemento borrado: " + borrado);
```

```
[a, b, c, d]
[a, c, d]
```

```
Elemento borrado: b
```

Importante: Antes de eliminar un elemento debes validar si existen elementos dentro del arreglo, esto para que el índice que se desea borrar al menos exista y el programa no se caiga .

Eliminar elemento que coincida con el valor entregado

Método remove();

```
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.remove("a");
System.out.println(a); //[b,c,d]
```

```
[a, b, c, d]
[b, c, d]
```

En este caso, el método `remove()` nos retornará true o false, dependiendo si se realiza o no la eliminación.

Eliminar elemento que coincida con el valor entregado

¿Qué pasa si tenemos más de un elemento con el mismo valor?

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

```
[a, b, c, c, c, c, a]
[b, c, c, c, c, a]
```

Va a eliminar solo la primera ocurrencia de este.

Eliminar todos los elementos dentro de una colección

Método public boolean removeAll(Collection c);

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

```
[a, b, c, c, c, c, a, d]
[b, d]
```

{desafío} latam_

Ejercicio guiado

Agregar número par

Crear un método que permita agregar solo números pares a un ArrayList y mostrar el o los elementos del ArrayList.

- Paso 1: Se crea un método llamado `agregarNumeroPar` que recibe como parámetro de entrada un número de tipo entero.
- Paso 2: Crear una variable local de tipo ArrayList llamada números.
- Paso 3: Realizamos la condición if para validar si el número ingresado es un número par.
- Paso 4: Si la condición se cumple, agregamos el elemento al ArrayList.

Agregar número par

Crear un método que permita agregar solo números pares a un ArrayList y mostrar el o los elementos del ArrayList.

- Paso 5: Mostramos el resultado con la sentencia `System.out.println`.

```
public static void main(String[] args) {
    agregarNumeroPar(3);
}
    public static void agregarNumeroPar(int numero) {
        ArrayList<Integer> numeros = new ArrayList<Integer>();
        if(numero%2 == 0) {
        numeros.add(numero);
    }
    System.out.println(numeros);
}
```

¿Qué hace el método `clear()`?

{desafío} latam_

¿Qué hace el método add( )?

Próxima sesión...

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

{desafío} latam_

{desafío}
latam_

Academia de
talentos digitales