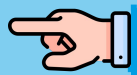
Arreglos y archivos

Iterando a partir del índice

{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 interfaz Iterator y sus principales métodos para tener una mejor claridad del uso y lo que nos facilita como programador.*
- *Aplicar las distintas operaciones de un arreglo para "buscar", "agregar", "eliminar" y "mostrar" agilizando el manejo y dando utilidad al interfaz Iterator.*

{desafío} latam_

¿Qué entendemos por iterador?

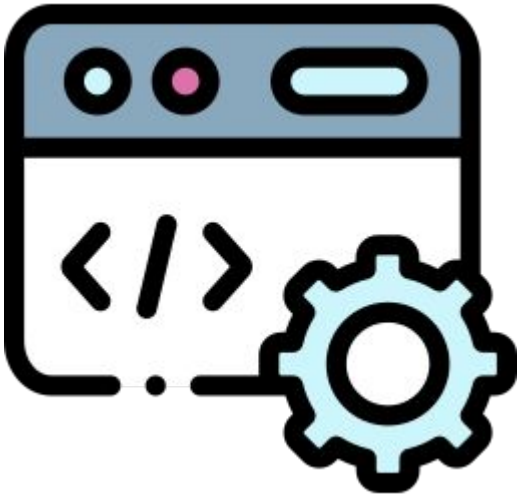
{desafío} latam_

/* Iterator */

¿Qué es una interfaz?

Una interfaz en Java es un conjunto de métodos abstractos y constantes que sirven como contrato para las clases que la implementan.

Una interfaz establece un conjunto de métodos que deben ser implementados, proporcionando así un estándar para la funcionalidad que se espera en las clases que la adoptan.



Usando Iterator

Interfaz que permite recorrer un arreglo en un bucle

Permite recorrer cualquier arreglo de tipo dinámico ArrayList, List, LinkedList, entre otros.

Métodos especiales para su uso y únicos para esta interfaz.

- **`hasNext()`**: Comprueba que siguen quedando elementos en el iterador.
- **`next()`**: Nos da el siguiente elemento del iterador.
- **`remove()`**: Elimina el elemento del Iterador.

Para trabajar con la interfaz Iterator se necesita importar su librería:

```
import java.util.Iterator;
```

Ejercicio guiado

{desafío}
latam_

Uso de Iterator con Arraylist

PASO 1:
Se declara e inicializa un ArrayList de tipo String.

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

Uso de Iterator con Arraylist

PASO 2:
Se agregan elementos al ArrayList.

```
random.add("Primero");
random.add("Segundo");
random.add("Tercero");
random.add("Cuarto");
random.add("Quinto");
```

Uso de Iterator con Arraylist

PASO 3:
Se declara un ciclo for donde se crea una variable temporal llamado iterator que es de tipo Iterator.

```
for (Iterator iterator = random.iterator(); iterator.hasNext();) {
}
```

Uso de Iterator con Arraylist

PASO 4:
La sentencia `String elementos = (String) iterator.next();` rescata el primer elemento del arreglo en el Iterator y lo asigna a la variable local llamada elementos.

```
for (Iterator iterator = random.iterator(); iterator.hasNext();) {
String elemento = (String) iterator.next();
}
```

Uso de Iterator con Arraylist

PASO 5:
Luego se muestra por consola el elemento dentro del bucle.

```
System.out.println("El elemento es"+ elemento);
```

Uso de Iterator con Arraylist

Solución completa

```
// Paso 1
ArrayList<String> random = new ArrayList<String>();
// Paso 2
random.add("Primero");
random.add("Segundo");
random.add("Tercero");
random.add("Cuarto");
random.add("Quinto");
// Paso 3
for (Iterator iterator = random.iterator(); iterator.hasNext();) {
     // Paso 4
     String elemento = (String) iterator.next();
     // Paso 5
     System.out.println("El elemento es"+ elemento);
}
```

Uso de Iterator con Arraylist

Solución completa

El código obtenido nos permite recorrer e interpretar el ciclo dentro de un ArrayList.

- Con la variable elementos podemos realizar condiciones de búsqueda dentro del bucle.

- El método `remove()` elimina elementos dentro del iterator, pero no del Arraylist.

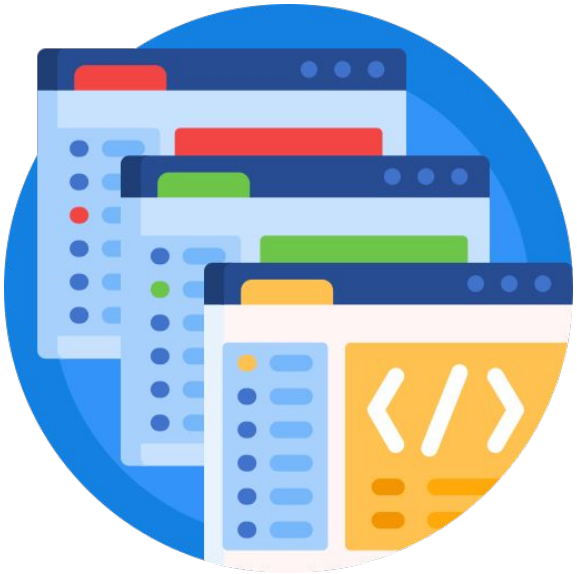
/* Matrices o arreglos de múltiples dimensiones */

¿Qué son?

- **Arreglos**

Estructuras de datos homogéneas (todos los datos son del mismo tipo) que permiten almacenar un determinado número de datos bajo un mismo identificador.

Importante: Dependiendo del tamaño del arreglo será la dimensión que este tenga.



Dimensión de un arreglo

No hay restricciones en cuanto al número de dimensiones que puede tener un arreglo, pero para los y las programadores es más compleja su manipulación ante un aumento significativo del número de dimensiones.

Ejemplo:

ArrayList con 1 fila y 5 columnas o dicho de otra forma, un vector (matriz unidimensional) de 5 columnas.

Dimensión de un arreglo

Si agregamos una segunda fila a continuación de esta, podríamos hablar del **origen de una matriz**, ya que hay más de una fila y una columna involucradas al mismo tiempo.

El índice también será para contar los datos que van llegando desde la segunda fila.

Índice	0	1	2	3	4
0	Primero	Segundo	Tercero	Cuarto	Quinto
1	Sexto	Séptimo	Octavo	Novemo	Décimo

/* Declaración y construcción */

Declaración de un arreglo bidimensional

```
tipo[][] nombre;
```

- **[]**: Indican que se trata de un arreglo bidimensional o matriz (cada corchete corresponde a una dimensión distinta)
- **tipo**: Hace referencia a los tipos de datos que se usarán (String, int, float, etc)
- **nombre**: Corresponde al alias que se le colocará a este arreglo

Declaración de un arreglo bidimensional

Otra forma de ver la declaración es igual que los arreglos unidimensionales, donde el proceso de declaración y construcción puede realizarse 2 sentencias, tal como se muestra a continuación:

```
nombre = new tipo[Filas][Columnas];
```

- **nombre**: alias
- **new**: creación del arreglo
- **tipo**: tipo de datos
- **[Filas]**: número total de filas
- **[Columnas]**: número total de columnas

Declaración de un arreglo bidimensional

También podemos encontrarlas mediante la expresión de constantes simbólicas, ya sea las 2 dimensiones o sola una de éstas.

```
public static final R = 10;
public static final C = 6;
matriz = new boolean [R][C]
```

Y también pueden usarse variables enteras previamente inicializadas:

```
int filas = 12;
int columnas = 8;
matriz = new double [filas][columnas]
```

La primera dimensión se refiere al total de filas de la matriz y la segunda se refiere al total de columnas.

Propiedad Length

- Propiedad para un arreglo de cualquier tipo de elementos.
- Contiene el número de elementos en el arreglo, pero para un arreglo bidimensional, esta propiedad puede indicar el número de filas como el número de columnas, dependiendo de cómo se utilice.

Por ejemplo:

```
int [] [] matriz = new int [5][3]; // se declara una matriz con enteros de 5 filas y 3 columnas
int x = matriz.length; // se declara una variable entera y se almacena el resultado de la propiedad length.
int y = matriz[2].length; // se establece que sea en la fila 2
```

Tipos de matrices

Matriz cuadrada

10	3	4	5
4	8	3	2
2	5	9	0
7	5	9	6

Matriz no cuadrada

10	3	4	5	8
4	8	3	2	2
2	5	9	0	5

Matriz irregular

1	2		
3	4	5	
6	7		
8	9	10	11

Declaración de la última matriz

```
int[][] matriz = new int[4][]; /* Solo se indica el número de
renglones, pero no el de columnas, porque será diferente en cada renglón */
matriz[0] = new int[2]; // En el renglón 0, se construyen 2 columnas
matriz[1] = new int[3]; // El renglón 1 contiene 3 columnas
matriz[2] = new int[2]; // El renglón 2 contiene 2 columnas
matriz[3] = new int[4]; // El renglón 3 contiene 4 columnas
```

Además, se puede declarar, construir e inicializar esta matriz realizando lo siguiente:

```
int[][] matriz = { {1, 2}, {3, 4, 5}, {6, 7}, { 8, 9, 10, 11} };
```

Introducción de datos a un arreglo bidimensional

Para introducir un dato a una casilla específica de un arreglo usaremos:

```
arreglo [fila][columna] = valor;
```

- **arreglo**: nombre del ArrayList
- **[fila]**: dimensión de la fila
- **[columna]**: dimensión de la columna
- **valor**: valor a almacenar en esa casilla

Búsqueda de un dato en un arreglo bidimensional

Si queremos acceder a los elementos de un arreglo bidimensional debemos hacer el conteo de índices que inicia desde 0, por lo que si un arreglo ha sido declarado y construido de la siguiente forma:

```
int[][] arreglo = new int[5][4];
```

Significa que tiene 20 casillas, elementos o espacios donde almacenará datos del tipo entero (int).

- Tendrá 5 filas, donde la dimensión va desde 0 hasta 4
- Tendrá 4 columnas, donde dimensión va desde 0 hasta 3

Búsqueda de un dato en un arreglo bidimensional

En el ejemplo anterior, podemos agregar elementos haciendo lo siguiente:

```
int[][] arregloMatriz = {{1,2,3,4}, {5,6,7,8}, {9,10,11,12},
{13,14,15,16}, {17,18,19,20}}
```

Por lo que la matriz sería la siguiente:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

¿Qué debemos hacer para buscar el elemento que está ubicado en la fila 3 y la columna 2?

¿Cuál será el resultado?

{desafío} latam_

Ejercicio guiado

Iteración por fila

Crearemos una matriz de 3x3, incorporando números del 1 al 9.

```
int n=3;
int matrizEjemplo[][] = {{1,2,3},{4,5,6},{7,8,9}}
for (int i=0; i<auxiliar.length; i++){
    for (int j=0; j<auxiliar.length; j++){
        System.out.println(matrizEjemplo[i][j] + "\t")
    }
}

// El resultado es: 1, 4, 7
```

Iteración por columna

Crearemos una matriz de 3x3, incorporando números del 1 al 9.

```
int n=3;
int matrizEjemplo[][] = {{1,2,3},{4,5,6},{7,8,9}}
for (int i=0; i<auxiliar.length; i++){
    for (int j=0; j<auxiliar.length; j++){
        System.out.println(matrizEjemplo[j][i] + “\t”)
    }
}

// El resultado es: 1, 2, 3
```

¿Qué hace el método `hasNext()`?

¿Cuál es la opción correcta para importar el Iterator?

1. `import Iterator;`
2. `util.Iterator;`
3. `import java.Iterator;`
4. `import java.util.Iterator;`
5. `No se puede determinar`

Próxima sesión...

- *Guia de ejercicios*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales