<codoa codo/>

Agencia de Aprendizaje a lo largo de la vida

Codo a Codo inicial

Estructuras de almacenamiento -Colecciones-

BA

Les damos la bienvenida

Vamos a comenzar a grabar la clase

Formulario de presentismo

Link:

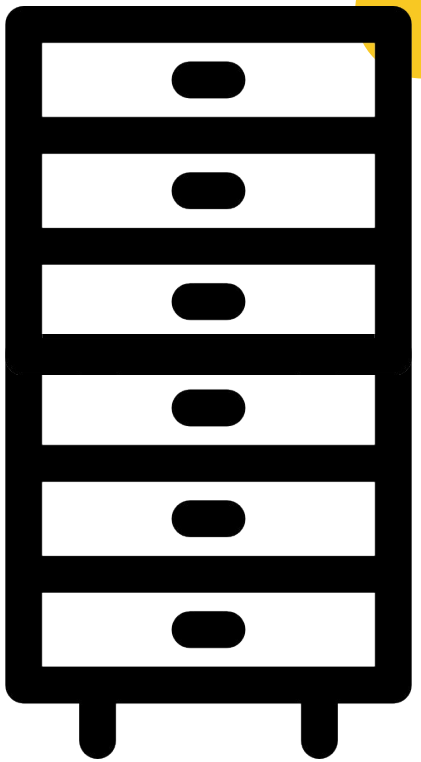
Write Once, Run Anywhere

(Escríbelo una vez, ejecútalo en cualquier lugar)

Estructuras de almacenamiento

Arrays

- **Un array es un objeto.**
- Es una **estructura de almacenamiento** que permite **almacenar muchos objetos de la misma clase o tipo e identificarlos con el mismo nombre.**
- **Siempre almacenamos elementos del mismo tipo**.
- Se lo declara con una **cantidad fija a almacenar** que luego **no se podrá modificar**.



¿Cómo creamos un Array?

Un array no necesita ser importado de manera explícita, Java se encarga de ello. La creación de un array consta de tres partes:

1. **Declaración** del tipo y nombre del array.
2. **Instanciación** nuestro objeto array **donde fijamos su capacidad de almacenamiento**.
3. **Inicialización** de los valores del array con la lista de valores a almacenar.

A continuación desarrollaremos cada uno de estos pasos.

```
public class Main {

    public static void main(String[] args) {
     //Declaracion


     //Instanciacion


     //Inicializacion




    }
```

1. Sintaxis de declaración de un Array

```
<tipoBase-o-clase>[] <identificador-array>;
o
<tipoBase-o-clase> <identificador-array>[];
```

Ejemplos de declaraciones de arrays:

```
int[] numerosEnteros;    // array de tipo int
double[] numerosReales;  // array de tipo double
String[] nombres;        // array de tipo String
Object[] objetos;        // array de la clase Object
```

¿Qué es el tipo base de un Array?

- Se denomina **tipo base** del array al tipo que se declara **para sus elementos**.
- Este tipo base puede ser un **tipo primitivo de Java**, un **objeto o una clase** definida.
- En los ejemplos anteriores se han utilizado tipos primitivos y clases como tipo base.
    - El array **numerosEnteros** almacena objetos del primitivo **int**.
    - El array **nombres** almacena objetos de la clase **String**.
    - El array **objetos** almacena referencias a instancias de la **clase Object** de Java.

2. La instanciación de un objeto array.

- **Instanciar** un array es **prepararlo** como un **nuevo objeto**. Imaginemos si estamos por preparar galletitas para hornearlas, **declarar** es como **seleccionar el molde** e **instanciar** hace referencia a **rellenarlo.**
- **En la instanciación** es necesario **indicar el número de elementos que va a almacenar**.
- **Cuando se instancia** un objeto array **se asigna un espacio de memoria ram para almacenar los elementos del array**.
- Para esto **es necesario saber el número total de elementos** que va a almacenar.

2. Sintaxis de instanciación de un array.

```
<identificador-array> = new <tipo_u_ojeto>[cantidad];
```

Instanciación de un array:

```
//Declaración del array
String[] nombres;         // array de tipo String
int[] numeros;            // array de tipo int


//Instanciación del array

nombres = new String[3]; //Almacenará 3 objetos tipo String.
numeros = new int[5];  //Almacenará 5 valores int.
```

3. Inicialización de valores del array.

- Cuando se crea un array se **inicializa** el valor de todos sus elementos al **valor por defecto** del tipo base del array, por ejemplo:
  - **cero** para los **números**,
  - **false** para los **boolean**,
  - **null** para las referencias a **objetos**.
- De forma similar al resto de objetos de Java, **un array se puede inicializar al momento de la declaración.** En este caso se inicializa al valor por defecto del tipo del array.

3. Inicialización de valores de un array.

```
identificador-array [indice] = valor;
```

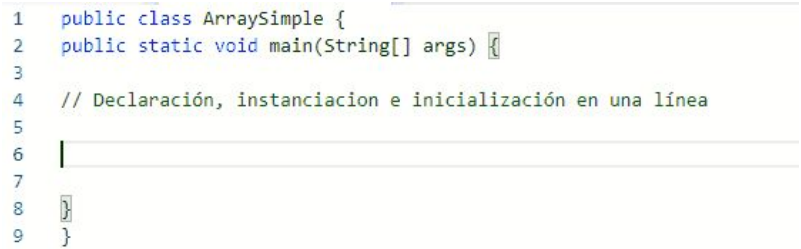
```
//Declaración del array
int[] numeros;           // array de tipo int


//Instanciación del array
numeros = new int[3];  //Almacenará 3 valores int.

//Inicialización de los valores, índice a índice
numeros[0] = 15;
numeros[1] = 56;
numeros[2] = 46;
```

Creación simplificada en una sola línea

Otra opción de crearlo de manera más directa es hacerlo en una sola línea:

1. **Se declara el array** con el **tipo y nombre** que corresponda y **se lo iguala** al **contenido que almacenará, encerrado entre llaves { }**.

Es una manera **mas simple y directa**, que resulta mas legible, no obstante **al dar los primeros pasos** muchas veces es **recomendable hacerlo mediante los tres pasos anteriores** hasta adquirir cierto manejo fluido con estos objetos.



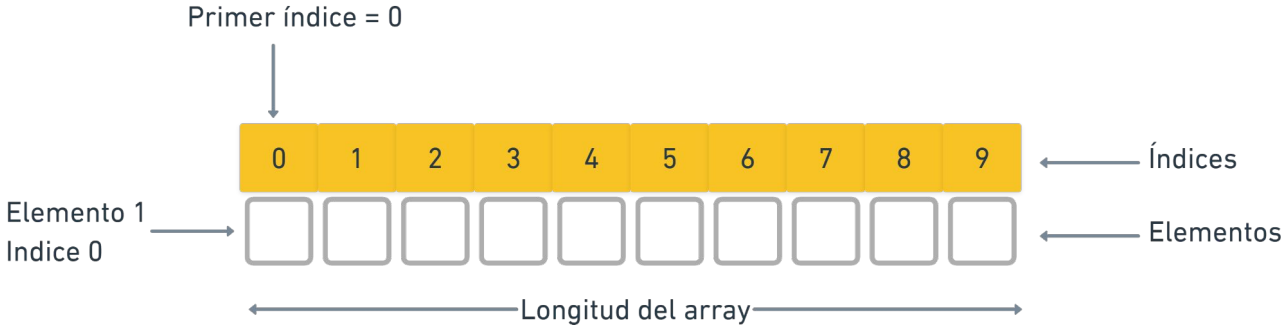
Declaración e inicialización de un array en una línea.

```
tipo[] identificador_Array = {valor1,...,valorN};
```

```
Un array también se puede inicializar indicando la lista de valores que va a
almacenar:

String[] diasLaborables = {"Lunes","Martes","Miércoles","Jueves","Viernes"};

int[] enteros = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Referencia en arrays, índice y posición.

- Para hacer **referencia** a cada elemento de un array es necesario **indicar la posición** que ocupa en la estructura de almacenamiento.
- Esta **posición** se denomina **índice**.
- **El primer elemento** de un array se almacena en la **posición cero** y el último elemento en la posición n-1, donde **n es el tamaño** del array.



Caso práctico de clase

Queremos almacenar las notas de 10 alumnos/as de una clase.
Calcular la media o promedio de notas

Soluciones posibles

1. **Crear 10 variables independientes** que almacenen las notas individualmente, cargar cada variable, codificar el promedio y obtener los resultados... Muchas líneas de código con seguridad.
2. **Crear un array de 10 elementos** y valernos de los métodos del array para obtener la información deseada... **Menos líneas de código, más óptimo.**

Caso práctico de clase

```
//Declaracion del array
int notas[];
//Instanciación del array
notas = new int[10]; //valores de tipo int
//Luego cargar de a uno

// O bien hacer declaración e inicialización del array
int[] notas={8,7,6,9,10,2,7,10,6,7};
```

Métodos de un array

- Podemos llamar un elemento de un un array refiriendonos a su **índice**, del ejemplo anterior:

```
System.out.println("El elemento 1 es" + diasLaborables[0])
```

  Esto imprime en consola: **El elemento 1 es Lunes**

- Podemos llamar a la longitud de un array mediante el método **length**

```
System.out.println("La longitud del array es" + diasLaborables.length);
```

  Esto imprime en consola: **La longitud del array es 5**

Métodos mas utilizados

- **Arrays.sort(array-de-datos)**. Ordena el contenido del array en orden ascendente.
  Arrays.sort(numeros) ordena todos los elementos del array numeros.
- **Arrays.sort(array-de-datos, inicio, fin)**. Ordena el contenido del array en orden ascendente, desde la posición de inicial hasta la posición final.
  Arrays.sort(numeros, 0, 49) ordena los elementos almacenados entre la posición 0 y la 49 del array numeros.
- **Arrays.binarySearch(array-de-datos, clave)**. Busca la clave indicada en el array de números enteros.
  Arrays.binarySearch(numeros, 1991) busca el número 1991 en el array numeros.
- **Arrays.fill(array-de-datos, dato)**. Rellena el array con el valor dado. Se puede utilizar con todos los tipos primitivos, String y con cualquier otro tipo de objeto.
  Arrays.fill(numeros, 5) rellena con el valor 5 todo el array numeros.
- **Arrays.fill(array-de-datos, dato, inicio, fin)**. Rellena el array con el valor dado, indicando la posición inicial y final.
  Arrays.fill(numeros, 5, 0, 5) rellena con el valor 5 desde la posición 0 hasta la posición 5 del array numeros.

ArrayList

Arrays redimensionables o Arraylist

- **Un ArrayList** es un **array redimensionable.**
- Puede almacenar un **número indefinido de elementos.**
- ***ArrayList*** es una clase de la biblioteca **java.util por lo que necesita ser importado** en el espacio de trabajo.

Arrays redimensionables o Arraylist

- Como **métodos** para operar con listas podemos señalar: **añadir** un objeto en una posición determinada, añadir un objeto al final de la lista, **recuperar** un objeto situado en determinada posición, etc.
- Los objetos de un ArrayList **se van insertando al final de la lista.**
- En arraylist **no utilizamos primitivas** para declararlo, **utilizamos wrappers.**

Tipos de wrappers en Java

Tipos primitivos (no son objetos y por tanto no poseen métodos)	Wrappers (son objeto y por tanto poseen métodos)
byte	Byte
short	Short
int	Integer
long	Long
boolean	Boolean
float	Float
double	Double
char	Character

Sintaxis de declaración e inicialización de un arraylist.

```
import java.util.ArrayList;

ArrayList<tipo_wrappers> nombre_lista=new ArrayList<tipo_wrappers>();
```

Instanciación de un ArrayList:

```
ArrayList<String> autos = new ArrayList<String>();
 cars.add("Volvo");
 cars.add("BMW");
 cars.add("Ford");
 cars.add("Mazda");

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

Mostrará por consola: [Volvo, BMW, Ford, Mazda]

Métodos mas utilizados en Arraylist

- **nombreDelArray.add(<valor>).** Inserta un valor en el arraylist.
- **nombreDelArray.get(numero_indice).** Muestra el elemento del indice indicado.
- **nombreDelArray.set(numero_indice, nuevo elemento).** Cambia en el índice indicado por el nuevo elemento.
- **nombreDelArray.remove(numero_indice).** Remueve el elemento del índice indicado.
- **nombreDelArray.clear().** Borra todos los elementos del arraylist.
- **nombreDelArray.size().** Indica cuantos elementos tiene el arraylist.

Métodos más utilizados en Arraylist

- **Collections.sort(nombre_arraylist).** Ordena la lista, el método .sort() se importa de la librería: java.util.Collections;
- El método **.indexOf(parámetro)** recibe como parámetro el elemento que queremos buscar en la lista. En el caso que el elemento se encuentre dentro de la lista se devolverá un entero con la posición en la que se encuentra. En el caso de que no lo encuentre dentro de la lista devolverá un número negativo.
- Podemos recorrer un arraylist valiéndonos del **bucle for** o el bucle **for-each**.

Bucle for-each

Iteración de una lista

Bucle for each

- Existe el bucle "for-each" (para cada uno) , que se utiliza exclusivamente para recorrer elementos de un array, por ejemplo para buscar un elemento particular o imprimir en consola el contenido.

Sintaxis

```
for (tipo nombre_variable : nombreArray) {
  // Bloque de código;
}
```

Ejemplo bucle for each

```
//Declaración e Inicializacion del array
String[] autos = {"Volvo", "Taunus", "Toyota", "Torino",”Dodge”};
//Declaro un contador
int cont = 1;

//Bucle for-each para imprimir en consola los elementos del array

for (String i : autos) {
  System.out.println(“El elemento ”+(i+1) “del array autos es ”+ i);
  cont ++;

  }

System.out.println(“*** El programa ha finalizado ***”);
```

Desafío de clase

Desafío de clase - 1

Hacer un array que me permita la carga por teclado de 10 notas de clases, terminada la carga arrojará, las notas cargadas y el promedio de ellas.

Desafío de clase - 2

- Programar una aplicación que utilice un arraylist para cargar un listado de cosas a comprar en el supermercado.
- Permitir al usuario cargar en un buscador el producto y si el producto está en el listado que arroje un aviso “Si, tenés que comprar el producto”

Desafío de clase 3

Hacer un array que me permita la carga por teclado el tamaño del arreglo, luego cargar datos, terminada la carga arrojará, los datos cargados.

Repo de clase

https://app.codingrooms.com/w/8sZ6zg6rD4FY

Herramientas que utilizamos en clases

IDE IntelliJ o VSCode+plugins

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- Realizá los ejercicios obligatorios.

Todo en el Aula Virtual.