# ARREGLOS EN JAVA

CESSI #ArgentinaPrograma #YoProgramo

---

Leonardo Blautzik - Federico Gasior - Lucas Videla

Agosto -Diciembre de 2021

# Objetivos

- Declarar y crear arreglos de tipos primitivos, o de clases.
- Explicar por qué se inicializan los elementos de un arreglo.
- Explicar cómo inicializar los elementos de un arreglo.
- Determinar la cantidad de elementos en un arreglo.
- Crear un arreglo multidimensional.
- Copiar valores de un arreglo a otro.

# Discusión

- ¿Cuál es el propósito de un **arreglo**?

# Discusión

- ¿Cuál es el propósito de un **arreglo**?
- Los arreglos se usan para agrupar elementos del mismo tipo.

# Discusión

- ¿Cuál es el propósito de un **arreglo**?
- Los arreglos se usan para agrupar elementos del mismo tipo.
- Permiten referenciar un grupo de elementos con un nombre en común.

# La Declaración de Arreglos

Se pueden declarar arreglos de tipos primitivos o de clases.

```
char[] s; //arreglo de tipo primitivo char
Punto[] p; //arreglo donde Punto es una class
```

Se pueden declarar arreglos usando los corchetes después del nombre de la variable.

```
char s[];
Punto p[];
```

Las declaraciones no especifican el tamaño actual del arreglo.

# La Creación de Arreglos

- En Java, **un arreglo es un objeto**, incluso si está compuesto de tipos primitivos.
- La declaración no crea al objeto en sí.
- Un arreglo se crea usando la palabra reservada `new`

```
char s[] = new char[26];
```

Luego de la creación los elementos del arreglo son inicializados con el valor por defecto del tipo de datos contenido (`'\u0000'` para los char).

# Inicialización de Arreglos

Llenamos el arreglo para poder usarlo:

```
public char[] createArray() {
  char[] s;
  s = new char[26];
  for (int i=0; i<26; i++) {
    s[i] = (char) ('A' + i);
  }
  return s;
}
```

# Mapa de Memoria Stack / Heap

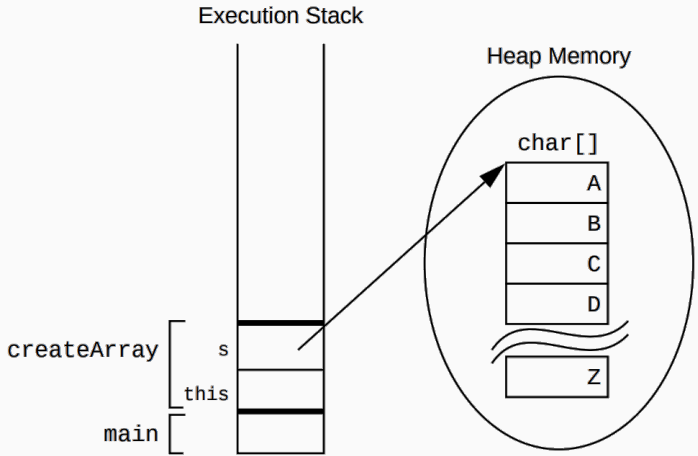


**Figure 1:** Creación e inicialización de un arreglo de char

# Un Arreglo de Puntos

Veamos ahora cómo tratar un arreglo de objetos **Punto**. La `class Punto` forma parte de nuestro paquete.

```
public class Punto {
  double x;
  double y;
  public Punto(double x, double y) {
    this.x = x;
    this.y = y;
  }
  public String toString() {
    return "[" + x + "," + y + "]";
  }
}
```

# Creamos e inicializamos el Arreglo de Puntos

```
public Punto[] createArray(int tam) {
    Punto[] p = new Punto[tam];
    for (int i = 0; i < tam; i++) {
        p[i] = new Punto(i, i + 1);
    }
    return p;
}
```

# Mapa de Memoria Stack / Heap para el Arreglo de Puntos

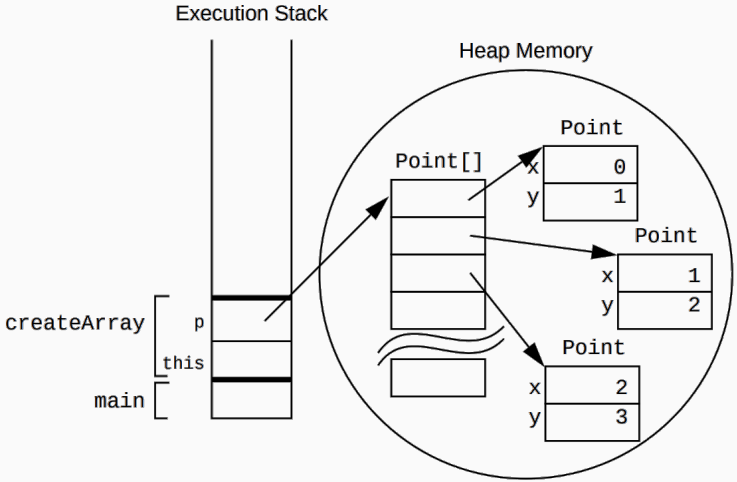


**Figure 2:** Creación e inicialización de un arreglo de Puntos

# Otras formas de inicializar un Arreglo

```
String names[];
names = new String[3];
names[0] = "Georgianna";
names[1] = "Jen";
names[2] = "Simon";

//Solo al momento de declalar el arreglo.
String names[] = {"Georgianna", "Jen", "Simon" };
```

# La inicialización de Arreglos

Cuando se crea un Arreglo se inicializa cada elemento. En el caso del arreglo de tipo char de la secciión anterior, cada valor se inicializa con el caracter nulo (`\u0000`).

En el caso del arreglo `p` cada valor se inicializa con el valor `null` indicando que no refiere aún a ningún objeto `Punto`. Después de la asignación `p[0] = new Punto();` es que el primer elemento del arreglo refiere a un objeto `Punto` real.

---

Nota:

Inicializar todas las variables, incluyendo los elementos del arreglo, es esencial para la seguridad del sistema. No deben quedar variables en un estado no inicializado.

---

# Los límites del Arreglo

- Los índices de los arreglos comienzan en 0.
- La cantidad de elementos del objeto arreglo es almacenado como un atributo del mismo, el atributo `length`.
- Si se intenta acceder a un elemento con un índice fuera de los límites de l arreglo se lanzará una Exception en tiempo de ejecución.

# Iterando sobre un arreglo usando el atributo length

```
for (int i = 0; i < enteros.length; i++) {
    System.out.println(enteros[i]);
}
```

Uso del atributo length para iterar sobre el arreglo enteros.

# La iteración foreach

```
for (int cadaElemento : elementos) {
    System.out.println(cadaElemento);
}
```

Uso de la iteración for mejorada (foreach) para recorrer el arreglo.
El compilador se encarga de la iteración.

# Copiar los elementos de un arreglo a otro

Sea el siguiente fragmento de código:

```
Punto [] puntos = new Punto[10];
for (int i = 0; i < puntos.length; i++) {
    puntos[i] = new Punto(i,i+1);
}

Punto [] otrosPuntos = puntos;

Punto [] otrosPuntos = Arrays.copyOfRange(puntos, 0, puntos.length);
for (Punto punto : otrosPuntos) {
    System.out.println(punto);
}
```

# Arreglos Multidimensionales

## Arreglos bidimensionales(matriz)

En java los arreglos bidimensionales se deben interpretar como Arrays de arrays:

```
int dosDim [][] = new int [4][];
dosDim[0] = new int[5];
dosDim[1] = new int[5];
dosDim[2] = new int[3]; //Cada fila puede tener distinta cantidad de columnas
dosDim[3] = new int[6];

int dosDim [][] = new int [][4]; //ilegal
```

# Declaración, Construcción y mapa de memoria de un arreglo bidimensional irregular

```
int dosDim [][] = new int [4][];
dosDim[0] = new int[4];
dosDim[1] = new int[4];
dosDim[2] = new int[3];
dosDim[3] = new int[6];
```

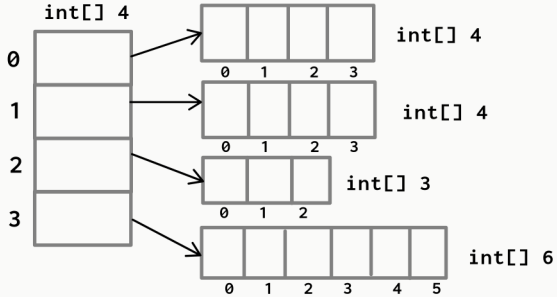


**Figure 3:** Arreglo bidimesional irregular

# Un Arreglo Tridimensional

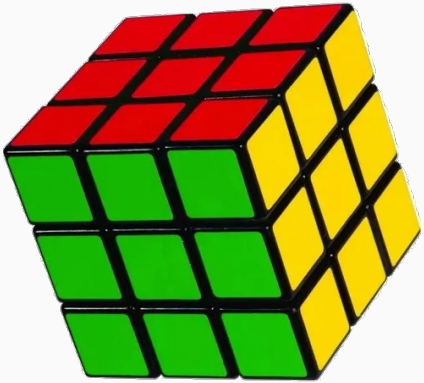

**Figure 4:** Cubo de Rubik

Pero no en Java!!