# Programación I
# Grado en Informática
# Curso 2018 / 2019

## ARRAYS
### Colecciones de elementos

### Tabla de contenidos

# 1. Introducción

Un **array** es una estructura que agrupa una colección de elementos del mismo tipo. De acuerdo a los tipos de datos existentes en Java, se pueden definir dos tipos de arrays:

- Arrays de tipos básicos. Permiten contener colecciones de elementos que representen valores específicos.
- Arrays de tipos no básicos. Permiten contener colecciones de elementos que representan entidades.

Al mismo tiempo, junto al tipo de datos representado, el manejo de los arrays viene condicionado por el número de dimensiones de los mismos. Los arrays de una dimensión se conocen como vectores, y los de dos o más dimensiones como matrices.

Junto a su concepción teórica, es necesario poner en práctica los conocimientos teóricos adquiridos a fin de garantizar la correcta utilización de las colecciones de elementos. Basándose en esta premisa, la presente práctica propone la realización de una serie de ejercicios sobre el manejo de arrays.

Antes de continuar con la resolución de los ejercicios se recomienda leer con atención este enunciado, a fin de conocer el procedimiento de evaluación, la duración y objetivos de la práctica, así como el material proporcionado para su correcta realización.

## Objetivos

El objetivo principal de la práctica es poner en uso los conocimientos sobre variables adquiridos en clase de teoría. Para ello se deberán resolver en lenguaje JAVA una serie de ejercicios sobre sentencias condicionales y de control.

Junto al objetivo general, la práctica tiene una serie de objetivos específicos. A saber:

- Gestión de memoria.
- Copia de arrays por referencia.
- Copia de arrays por valor.
- Uso de argumentos por la línea de comandos.

## 2. Ejercicios

La realización de la práctica implicará la resolución de una serie de ejercicios de programación. Los ejercicios se presentan agrupados por contenidos.

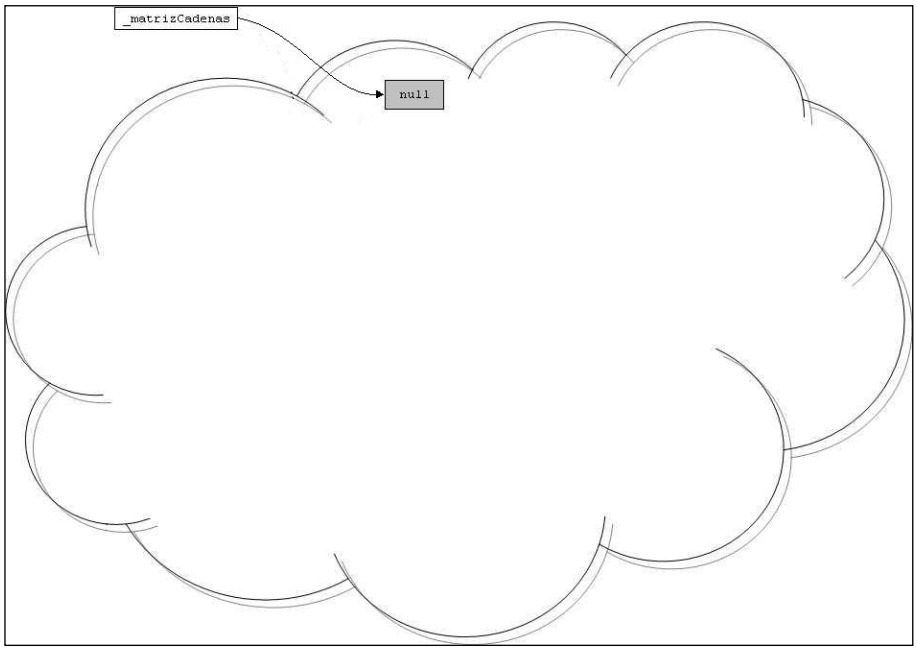
### Gestión de memoria

A continuación, se proponen una serie de ejercicios encaminados a conocer la forma en la cual se utiliza la memoria al trabajar con arrays. Para ello, se deberán completar los distintos métodos de acuerdo a la figura recogida en cada ejercicio.

> **IMPORTANTE**
>
> El resultado mostrado en cada una de las figuras sería el obtenido **tras ejecutar el método** correspondiente. Así mismo, los ejercicios deben tomarse como acumulativos; es decir, la figura es el resultado de ejecutar el método del ejercicio y todos los anteriores del apartado.

#### Ejercicio 1

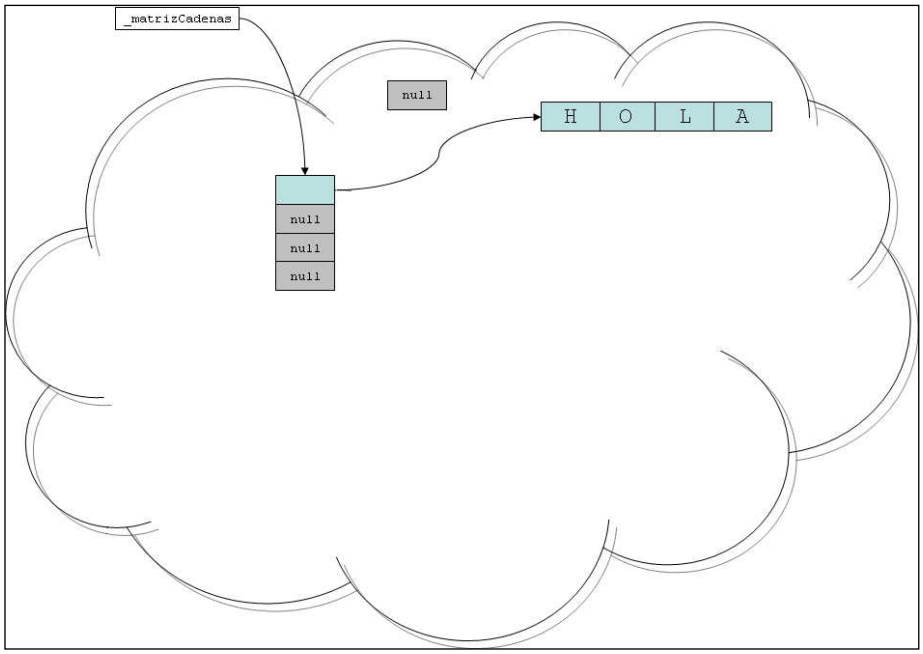
Dada la siguiente figura:




Se pide completar el método `ejercicio01()` de la clase `Apartado030101` para que, una vez ejecutado el método, la situación en memoria sea igual a la representada en la figura.

#### Ejercicio 2

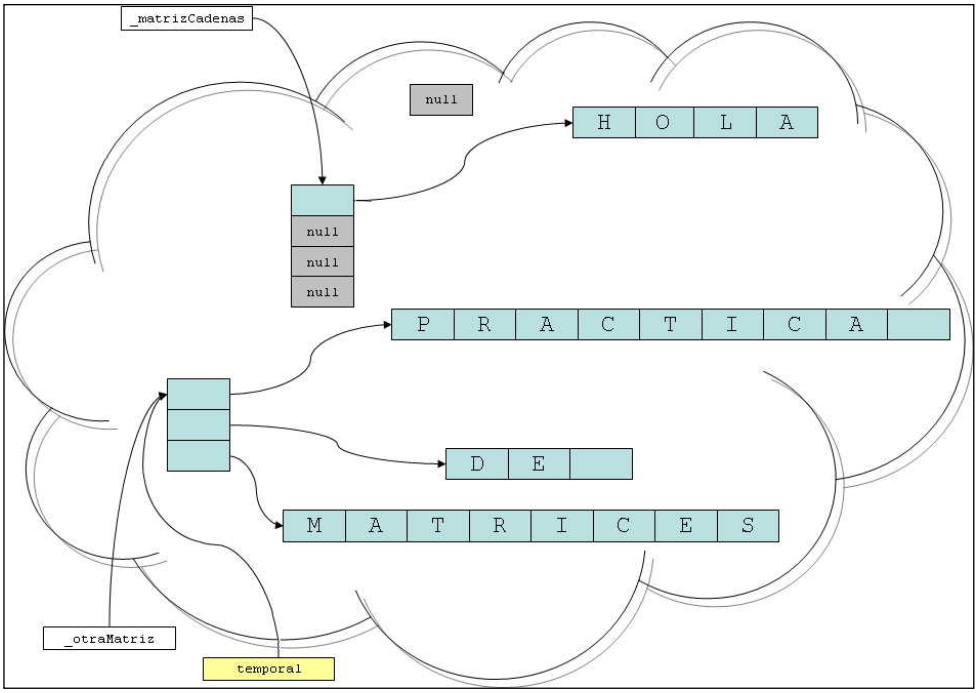
Dada la siguiente figura:




Se pide completar el método `ejercicio02()` de la clase `Apartado030101` para que, una vez ejecutado el método, la situación en memoria sea igual a la representada en la figura.

### Ejercicio 3

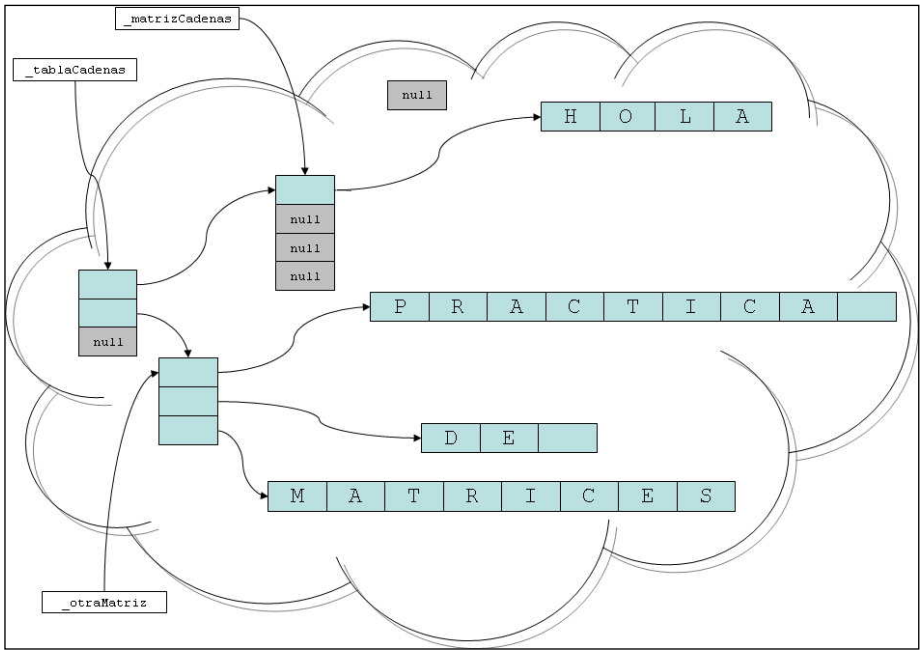
Dada la siguiente figura:




Se pide completar el método `ejercicio03()` de la clase `Apartado030101` para que, una vez ejecutado el método, la situación en memoria sea igual a la representada en la figura.

### Ejercicio 4

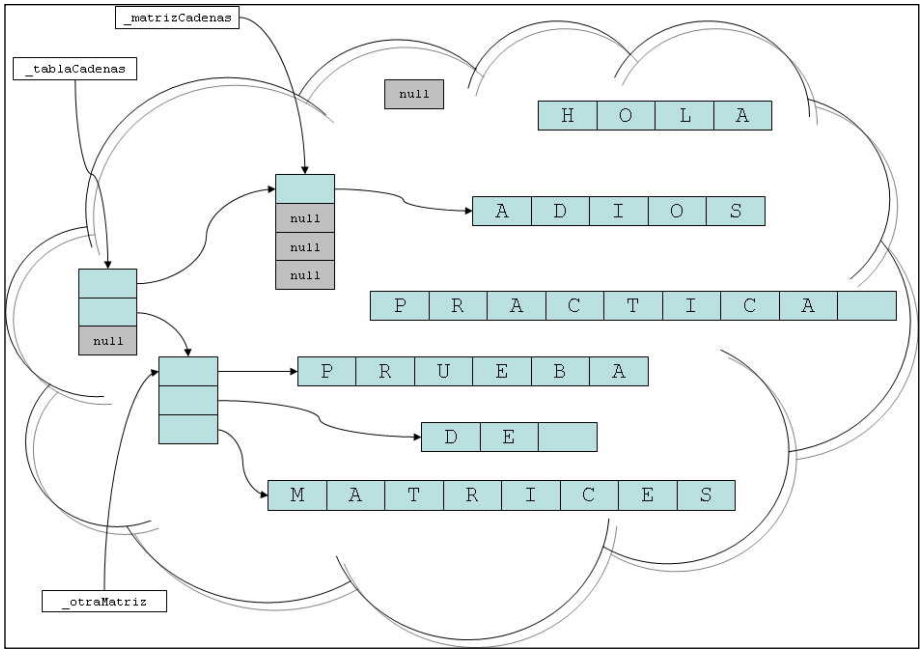
Dada la siguiente figura:




Se pide completar el método `ejercicio04()` de la clase `Apartado030101` para que, una vez ejecutado el método, la situación en memoria sea igual a la representada en la figura.

### Ejercicio 5

Dada la siguiente figura:




Se pide completar el método `ejercicio05()` de la clase `Apartado030101` para que, una vez ejecutado el método, la situación en memoria sea igual a la representada en la figura.

## Arrays de tipos básicos

Uno de los fallos habituales producidos al emplear arrays pasa por el acceso a elementos erróneos. Al utilizar arrays debe tenerse muy presente:

- Las posiciones de las arrays se numeran desde cero.
- La longitud de la array viene dada por el atributo **`length`**.
- El lugar que ocupa cada elemento sirve para identificarlo.

A continuación, se recogen una serie de ejercicios sobre arrays de tipos básicos que inciden en estos puntos.

### Ejercicio 1

Completar el método `ejercicio01()` de la clase `Apartado030102` para crear y rellenar un vector de cien posiciones que contenga los primeros cien números pares. Una vez creada, se deberá mostrar el contenido del vector por pantalla.

### Ejercicio 2

Dado el siguiente fragmento de código:

```
int[] arrayEnteros = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
System.out.println(" { ");
for(int i=1 ; i <= arrayEnteros.length ; i++) {
   System.out.print(" "+arrayEnteros[i]+",");
}
System.out.println(" } ");
```

Se pide:

- Compilar y ejecutar el método `ejercicio02()` de la clase `Apartado030102`.
- Analizar los resultados obtenidos.
- Explicar en el fichero LEEME.TXT el porqué de los resultados.
- Modificar el código a fin de evitar la excepción producida y mostrar todo el contenido del vector.

### Ejercicio 3

Completar el método `ejercicio03()` de la clase `Apartado030102` para crear una matriz que contenga la tabla de multiplicar del número 8. El contenido de la matriz será similar al siguiente:

```
[ 1] [ 2] [ 3] [ 4] [ 5] ...
[ 8] [ 8] [ 8] [ 8] [ 8] ...
[ 8] [16] [24] [32] [40] ...
```

Una vez creada, la matriz se deberá mostrar por pantalla.

## Ejercicio 4

El alumno deberá programar el código que permite mostrar por pantalla el siguiente resultado:

```
Loterías y Apuestas del Estado – BonoLoto

    Apuesta: 1
    [ 0][ 1][ 2][ 3][ 4][ 5][ 6][ 7][ 8][ 9]
    [10][11][12][13][14][15][16][17][18][19]
    [20][21][22][23][24][25][26][27][28][29]
    [30][31][32][33][34][35][36][37][38][39]
    [40][41][42][43][44][45][46][47][48][49]

    Apuesta: 2
    [ 0][ 1][ 2][ 3][ 4][ 5][ 6][ 7][ 8][ 9]
    [10][11][12][13][14][15][16][17][18][19]
    [20][21][22][23][24][25][26][27][28][29]
    [30][31][32][33][34][35][36][37][38][39]
    [40][41][42][43][44][45][46][47][48][49]

    Apuesta: 3
    [ 0][ 1][ 2][ 3][ 4][ 5][ 6][ 7][ 8][ 9]
    [10][11][12][13][14][15][16][17][18][19]
    [20][21][22][23][24][25][26][27][28][29]
    [30][31][32][33][34][35][36][37][38][39]
    [40][41][42][43][44][45][46][47][48][49]
```

Las modificaciones se realizarán sobre el método `ejercicio04()` de la clase `Apartado030102`.

**IMPORTANTE**

El ejercicio admite varias soluciones; sin embargo, únicamente se tomará como válida aquella basada en el uso de **inicializadores** y **sentencias de repetición**.

## Arrays de tipos no básicos

### Ejercicio 1

Tomando como referencia la clase `Persona`, se pide programar el código que permite disponer un vector de 10 posiciones con el siguiente contenido:

```
Posición 0 => [Pedro, 913779900]
Posición 2 => null
Posición 3 => null
Posición 4 => [Juan, 915453322]
Posición 5 => null
Posición 6 => null
Posición 7 => [Luis, 629674532]
Posición 8 => null
Posición 9 => [Pepe, 913778800]
```

Las modificaciones se realizarán sobre el método `ejercicio01()` de la clase `Apartado030201`.

### Ejercicio 2

Se pide modificar el contenido de la clase `Persona` y del método `ejercicio02()` de la clase `Apartado030201` a fin de mostrar por pantalla los NOMBRES contenidos en el vector creado en el ejercicio anterior. La información se mostrará **utilizando una sentencia "for"**.

**IMPORTANTE**

Al tratarse de un vector de tipos no básicos es necesario controlar los elementos vacíos de la colección. En caso contrario, se producirán excepciones del tipo NullPointerException.

### Ejercicio 3

Tomando como referencia la clase `Persona`, se pide programar el código que permite disponer un matriz de 2X2 el siguiente contenido:

```
Posición 0,0 => [Luis, 629674532]
Posición 0,1 => null
Posición 1,0 => null
Posición 1,1 => [Pepe, 913778800]
```

La matriz deberá **crearse usando inicializadores**. Una vez creada, se mostrarán por pantalla los NOMBRES contenidos en la matriz. Las modificaciones se realizarán sobre el método `ejercicio03()` de la clase `Apartado030201`.

## Manejo avanzado de arrays

En este apartado se recogen una serie de ejercicios que ilustran el manejo de arrays y la realización de operaciones sensibles con las mismas.

### Ejercicio 1

La copia por referencia implica trabajar sobre las mismas posiciones de memoria, de tal forma que los cambios realizados en la copia afectan al array original. Dado el siguiente fragmento de código:

```
Persona[] personas = new Persona[2];
Persona[] copia;

personas[0] = new Persona("Pedro", "913779900");
personas[1] = new Persona("Juan", "915453322");

//Se realiza la copia.
copia = personas;

contador = 0;
while ((contador < copia.length) && (copia[contador] != null))
{
    System.out.println(copia[contador]);
    contador++;
}

personas[0] = new Persona("Luis", "610010101");

contador = 0;
while ((contador < copia.length) && (copia[contador] != null))
{
    System.out.println(copia[contador]);
    contador++;
}
```

Se pide:

- Compilar y ejecutar el método `ejercicio01()` de la clase `Apartado030202`.
- Analizar los resultados obtenidos.
- Explicar en el fichero LEEME.TXT el porqué de los resultados.

### Ejercicio 2

La copia por valor implica duplicar la información en posiciones de memoria distintas. En este caso, los cambios realizados en la copia no afectan al array original. Dado el siguiente fragmento de código:

```
Persona[] personas = new Persona[2];
Persona[] copia;

personas[0] = new Persona("Pedro", "913779900");
personas[1] = new Persona("Juan", "915453322");

//Se realiza la copia de la array.
copia = new Persona[personas.length];
contador = 0;
while ((contador < personas.length) &&
       (personas[contador] != null)) {
  copia[contador] = new Persona(personas[contador]);
  contador++;
}

contador = 0;
while ((contador < copia.length) && (copia[contador] != null))
{
  System.out.println(copia[contador]);
  contador++;
}

personas[0] = new Persona("Luis", "610010101");

contador = 0;
while ((contador < copia.length) && (copia[contador] != null))
{
  System.out.println(copia[contador]);
  contador++;
}
```

Se pide:

- Compilar y ejecutar el método `ejercicio02()` de la clase `Apartado030202`.
- Analizar los resultados obtenidos.
- Explicar en el fichero LEEME.TXT el porqué de los resultados.

### Ejercicio 3

El paso de parámetros en Java es siempre por valor; es decir, se realiza una copia de la variable al argumento recibido, de manera que se trabaja sobre posiciones de memoria distintas. Dados los métodos `pasarParametros()` y `recibirParametros()` de la clase `PruebaCodigo`, se pide:

- Compilar y ejecutar el método `ejercicio03()` de la clase `Apartado030202`.
- Analizar los resultados obtenidos.
- Explicar en el fichero LEEME.TXT el porqué de los resultados.

### Ejercicio 4

El paso de parámetros en Java es siempre por valor; es decir, se realiza una copia de la variable al argumento recibido, de manera que se trabaja sobre posiciones de memoria distintas. Dados métodos `pasarParametrosErroneo()` y `recibirParametrosErroneo()` de la clase `PruebaCodigo`, se pide:

Se pide:

- Compilar y ejecutar el método `ejercicio04()` de la clase `Apartado030202`.
- Analizar los resultados obtenidos.
- Explicar en el fichero LEEME.TXT el porqué de los resultados.

### Ejercicio 5

Modificar las clases `Banco`, `CuentaBancaria` y `Titular` de acuerdo al siguiente enunciado:

> *El banco "MTP Money S.A." gestiona 50 cuentas bancarias, cada una de las cuales tendrá como mínimo un titular, considerado como titular principal. Al mismo tiempo, la cuenta puede tener varios titulares hasta un máximo de 5. Un titular viene identificado por su nombre, edad, NIF y crédito.*
>
> *Para sacar dinero de la cuenta el titular principal debe ser mayor de edad, y la cuenta debe tener crédito suficiente para cubrir la cantidad deseada.*

El código programado se probará utilizando el método `ejercicio05()` de la clase `Apartado030202`.

# 3. Material proporcionado

Para la realización de la práctica se proporciona, junto a este enunciado, un fichero comprimido con el material necesario para su correcta realización. Una vez descomprimido el fichero, se creará un directorio `/p-arr` con la siguiente estructura de trabajo:

- **src.** Ficheros fuente de la aplicación. Contiene el programa principal de prueba.
- **classes.** Ficheros compilados. Contiene los *bytecode* de los ficheros fuente.
- **doc.** Documentación javadoc.
- **etc.** Ficheros de configuración. Contiene el esbozo del fichero LEEME.TXT a completar.

## Definición de ejercicios

Dentro del directorio `/src`, se encuentra la definición de los métodos que deberán completarse de acuerdo a los enunciados previamente expuestos. La definición de los métodos recoge la signatura completa, incluyendo comentarios Javadoc que reproducen el contenido del enunciado correspondiente. Al realizar la implementación de los métodos se deberá respetar la información contenida en estos comentarios.

## Casos de prueba

Para probar el resultado de la práctica, se proporciona una clase de prueba. Esta clase ejecuta los distintos ejercicios en función de los parámetros recibidos por la línea de comandos. Se proporciona un `build.xml` que solicitará, una vez que la compilación tenga éxito, la entrada por consola de dos parámetros que corresponden con:

[*apartado*] Apartado de la práctica al cual pertenece el ejercicio (solo el número, ej: 030101).

[*ejercicio*] Número de ejercicio (solo el número, ej. 01).

### Resolución de errores

Al ejecutar la clase de prueba pueden producirse excepciones originadas por el código del alumno. Estos errores serán emitidos por la clase de prueba de la siguiente manera:

```
Resultado de la ejecución

java.lang.reflect.InvocationTargetException
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native
Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown
Source)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
        at java.lang.reflect.Method.invoke(Unknown Source)

        at PruebaPractica.main(PruebaPractica.java:36)
Caused by: java.lang.NullPointerException
        at Apartado030201.ejercicio02(Apartado030201.java:75)
        ... 5 more
```

En este caso, a fin de saber la causa del error, el alumno debe fijarse en la parte final del mensaje emitido por pantalla: a partir de la línea que indique **`PruebaPractica.main.`** Así, en el ejemplo mostrado, el error se origina en la línea 75 de la clase `Apartado030201`. Se trata de un error producido al ejecutar un objeto declarado pero no creado; es decir, que apunta a "null".