

JAVA



TEMA 07: ARRAYS DE OBJETOS

ÍNDICE

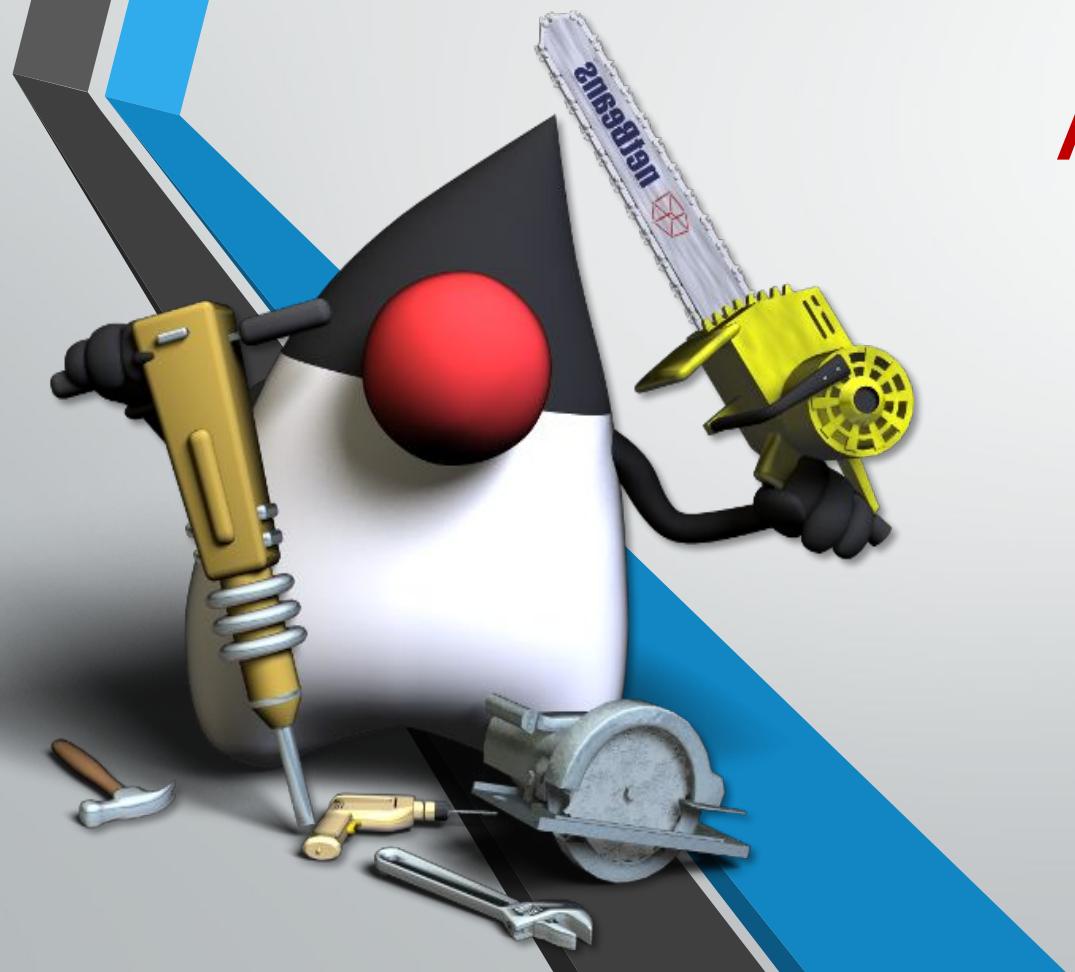
- 1. Arrays de objetos**
- 2. Recorrido de arrays con for-each**
- 3. La clase Arrays**
- 4. Ejercicios de consolidación**



JAVA

ARRAYS DE OBJETOS

1. Arrays de objetos



1.- Arrays de objetos

- Un array, al igual que puede contener datos simples (int, float, char, double...), también puede contener objetos.
- Definición + Instanciación:

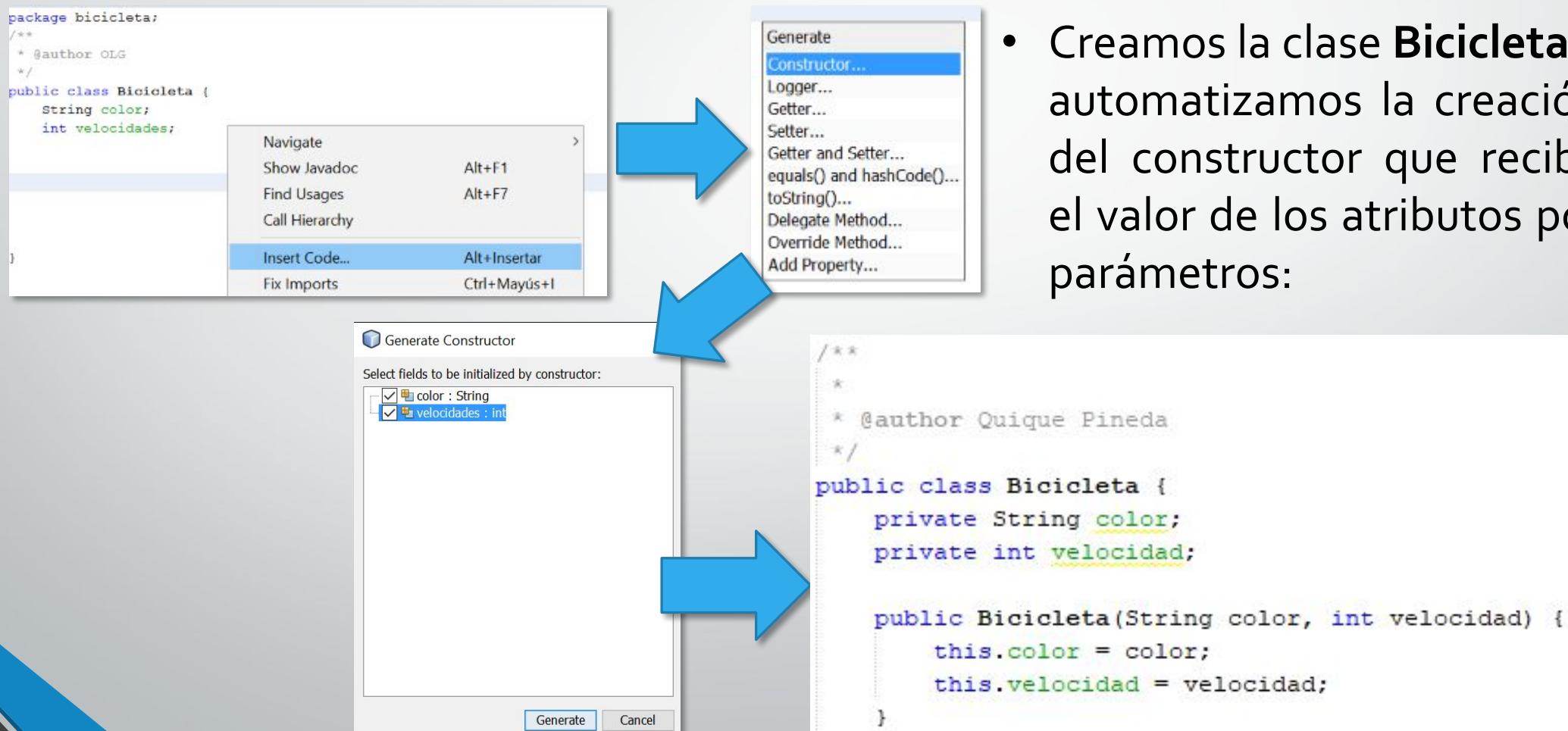
NombreClase[] nombrearray = new NombreClase[tamañoarray];

- Creación de un objeto en una posición del array:

nombrearray [posiciónarray] = new NombreClase();

- Veamos un sencillo ejemplo:

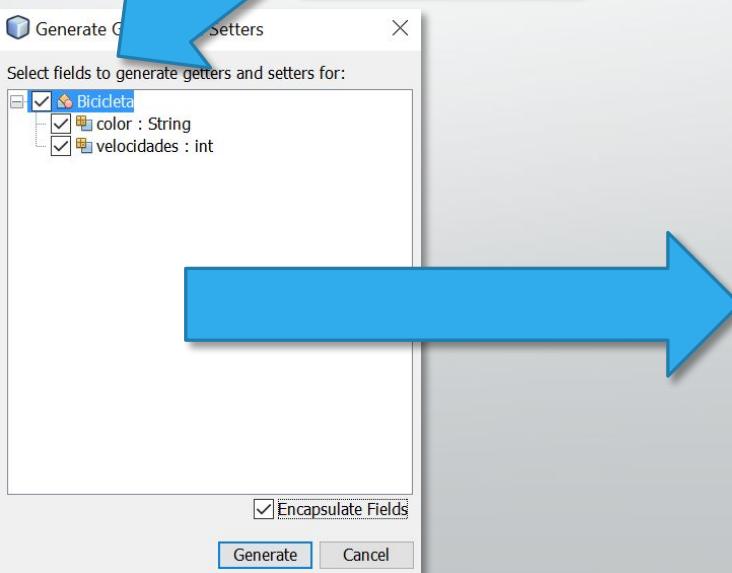
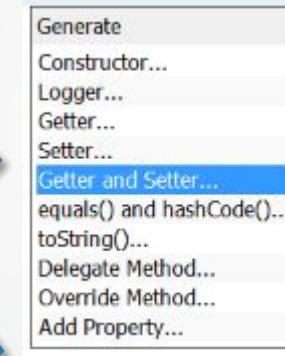
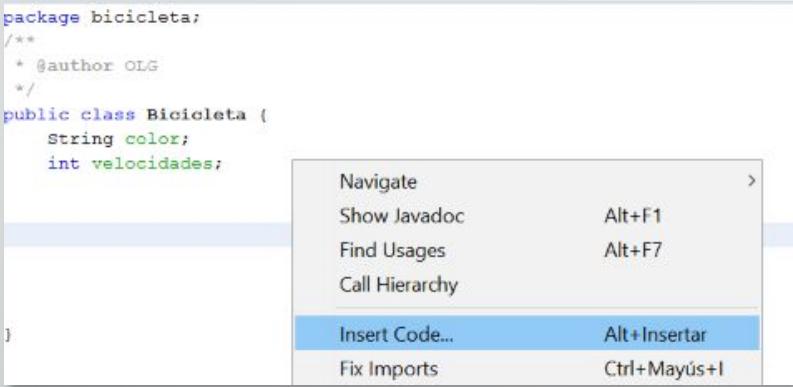
1.- Arrays de objetos



- Creamos la clase **Bicicleta** y automatizamos la creación del constructor que recibe el valor de los atributos por parámetros:

1.- Arrays de objetos

```
package bicicleta;
/**
 * @author OLG
 */
public class Bicicleta {
    String color;
    int velocidades;
}
```



```
* @author Quique Pineda
*/
public class Bicicleta {
    private String color;
    private int velocidad;

    public Bicicleta() {
        this.color = "";
        this.velocidad = 0;
    }

    public Bicicleta(String color, int velocidad) {
        this.color = color;
        this.velocidad = velocidad;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public int getVelocidad() {
        return velocidad;
    }

    public void setVelocidad(int velocidad) {
        this.velocidad = velocidad;
    }
}
```

- Automatizamos la creación de los métodos get y de los set (obtener y establecer), así como la encapsulación de los atributos:

1.- Arrays de objetos

- Por último creamos la clase **Test** donde creamos un array de 10 posiciones.
 - Rellenamos la posición 3 y 6 con dos objetos de la clase Bicicleta y las mostramos por pantalla:

```
ta.java x Test.java x
History | 
package bicicleta;
/**
 * @author OLG
 */
public class Test {
    public static void main(String[] args) {
        Bicicleta[] almacen = new Bicicleta[10]; /*Creo un array llamado almacen con 10 objetos de la clase Bicicleta*/
        almacen [3] = new Bicicleta ("rojo",21); /* Creo un objeto de la clase Bicicleta con el constructor por parametros, con los atributos "rojo" y 21, y lo guardo en la posicion [3] del array almacen */
        almacen [6] = new Bicicleta(); /* Creo un objeto de la clase Bicicleta con el constructor por defecto y lo guardo en la posicion [6] del array almacen */
        almacen[6].setColor("Azul"); // Asignamos el color "Azul" a la bicicleta de la posicion [6]
        almacen[6].setVelocidades(18); // Asignamos las velocidades "18" a la bicicleta de la posicion [6]
        System.out.println(almacen[3].getColor()); /*Mostramos el atributo color del objeto situado en la posicion[3] del array almacen*/
        System.out.println(almacen[3].getVelocidades()); /*Mostramos el atributo velocidades del objeto situado en la posicion[3] del array almacen*/
        System.out.println(almacen[6].getColor());
        System.out.println(almacen[6].getVelocidades());
    }
}
```

EJERCICIOS

- **Ejercicio 01.- (OPTATIVO)** Realiza un programa en JAVA en el que le pidas al usuario las notas de las 6 asignaturas del Ciclo de DAM y te calcule la nota media del curso.
- Cada una de las asignaturas serán un objeto que se encuentran en un array de 6 posiciones, y cuyos atributos serán el nombre y la nota.
- Crea un constructor con el que asigne directamente el nombre de la asignatura y la nota al crear el objeto. El nombre de la asignatura se lo asignaremos nosotros, en cambio, el atributo nota, será el usuario quien la introduzca mediante un método.
- Crea otro método que reciba el array con las 6 notas y devuelva la nota media (return)
- Ejemplo de ejecución:

Por favor, introduzca la nota de Programación: 6,5

Introduzca la nota de Lenguajes de Marcas: 7,5

Introduzca la nota de Bases de Datos: 7,5

Introduzca la nota de Entornos de Desarrollo: 8

Introduzca la nota de Sistemas Informáticos: 6,5

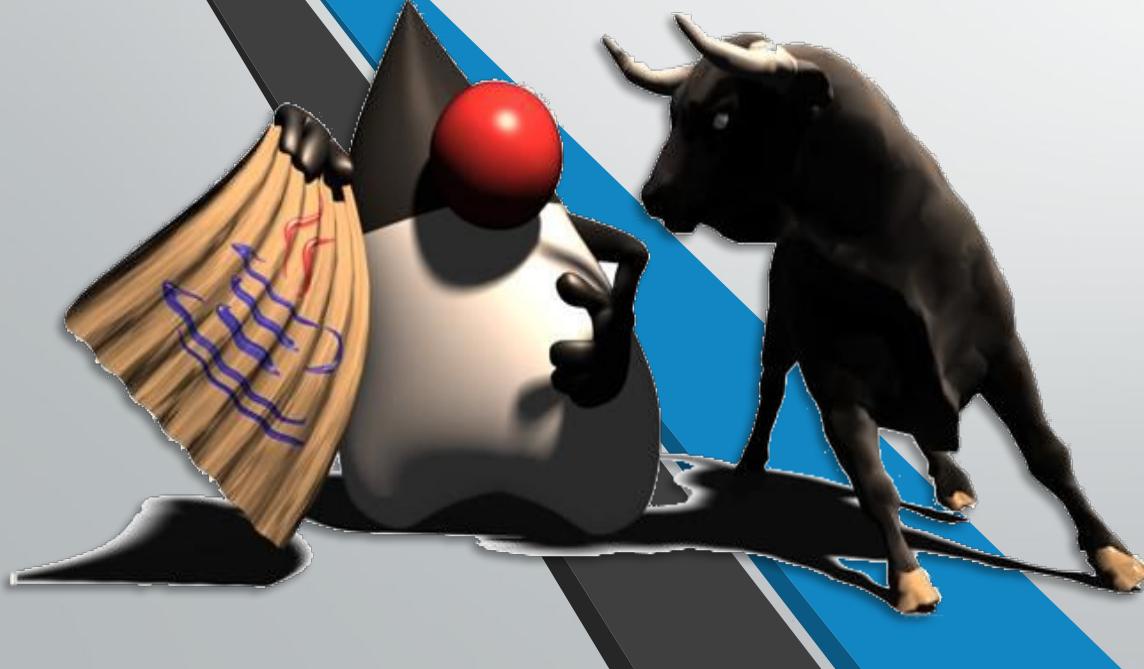
Por último, introduzca la nota de Formación y Orientación Laboral: 6

Su nota media del curso es de: 7

JAVA

ARRAYS DE OBJETOS

2. Recorrido de arrays
con for-each



2.- Recorrido de arrays con for-each

- A partir de la versión 5 de JAVA, se incorpora una variante del bucle for, conocida como for-each, que facilita el recorrido de los Arrays ya que no necesita un índice para acceder a las distintas posiciones.

- La palabra reservada sigue siendo for:

```
for (tipo nombre_variable: nombre_del_array)  
    ... instrucciones ...  
}
```

- Veamos un ejemplo:

2.- Recorrido de arrays con for-each

- Supongamos que tenemos el siguiente array:

```
int [] vector = { 4, 6 , 30 , 15 };
```

- La forma tradicional para mostrar por pantalla los elementos almacenados es:

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

2.- Recorrido de arrays con for-each

- Utilizando la nueva instrucción for-each, la misma operación se realizará de la siguiente forma:

```
for (int n : vector) {  
    System.out.println ( n );  
}
```

Fíjate que, sin acceder de forma explícita a las posiciones del array, el contenido de cada una de éstas es copiada automáticamente a la variable auxiliar *n* al principio de cada iteración.

2.- Recorrido de arrays con for-each

- Si, por ejemplo, quisiésemos sumar los elementos de un array sería así:

```
int suma=0;  
for (int n : vector) {  
    suma = suma + n; //suma+=n;  
}
```

En vez de:

```
for (int i = 0 ; i < vector.length ; i++){  
    suma = suma + vector [ i ]; //suma+=vector[i];  
}
```

2.- Recorrido de arrays con for-each

- Ahora supongamos que tenemos el siguiente array bidimensional:

```
int [][] array = {{4, 6, 9}, {13, 15, 21}};
```

- La forma tradicional para mostrar por pantalla los elementos almacenados es:

```
for ( int i=0; i<vector.length; i++){ //for (int i=0;i<2;i++)  
    for ( int j=0; j<vector[ i ].length; j++){ //for (int j=0;j<3;j++)  
        System.out.println ( vector [ i ][ j ] );  
    }
```

2.- Recorrido de arrays con for-each

- Utilizando la nueva instrucción for-each, la misma operación se realizará de la siguiente forma:

```
for ( int [ ] fila : array) {  
  
    for ( int columna : fila) {  
  
        System.out.println(columna);  
  
    }  
  
}
```

EJERCICIOS

- **Ejercicio 02.- (OPTATIVO)** Escribir un programa en JAVA que contenga un método que rellena un array unidimensional (vector) de enteros ALEATORIOS entre el 1 y el 6, y luego, otro método, lo muestre por pantalla, utilizando el bucle for-each.
- El programa también tendrá un método donde el usuario elegirá la longitud del array entre 1 y 10. Este método también controla que el usuario nos introduzca un número entero y no una letra (mediante excepciones). En caso de que el usuario introduzca el dato incorrecto se lo volveremos a pedir las veces que hagan falta.

JAVA

ARRAYS DE OBJETOS

3. La clase Arrays

3.- La clase Arrays

- La API de JAVA nos proporciona la clase ya definida Arrays (en el paquete `java.util`) para realizar operaciones con Arrays, como llenar el array, buscar si un dato está en el array, y ordenar los datos de un array.
- Para ello contiene 4 métodos estáticos:
 - `fill`
 - `equals`
 - `sort`
 - `binarySearch`

3.- La clase Arrays

- **Método fill:** Este método rellena un array con un valor.
- Ejemplo:

```
int [ ] vector = new int [50];
```

```
Arrays.fill ( vector , 2 ); /* Rellena todas las posiciones  
del vector con el número 2 */
```

```
Arrays.fill ( vector , 5 , 8 , 66 ); /* Rellena desde el  
índice 5 hasta el 7 con el valor 66 */
```

3.- La clase Arrays

```
package arrayfill;
import java.util.Arrays;
/**
 * @author Oscar Laguna García
 */
public class ArrayFill {

    public static void mostrarArray(int [] valores){
        int i;
        for (i=0; i<valores.length;i++){
            System.out.print(" ["+i+"] ->" +valores[i]);
        }
        System.out.print(" \n");
    }

    public static void main(String[] args) {
        int [ ] valores = new int [10];
        Arrays.fill ( valores , 2 ); /* Rellena todas las posiciones del array valores con el número 2 */
        mostrarArray(valores);
        Arrays.fill ( valores , 5 , 8 , 0 ); /* Rellena desde el indice 5 hasta el 7 con el valor 0*/
        mostrarArray(valores);
    }
}
-ArrayFill (run)
run:
[0]->2 [1]->2 [2]->2 [3]->2 [4]->2 [5]->2 [6]->2 [7]->2 [8]->2 [9]->2
[0]->2 [1]->2 [2]->2 [3]->2 [4]->2 [5]->0 [6]->0 [7]->0 [8]->2 [9]->2
BUILD SUCCESSFUL (total time: 0 seconds)
```

3.- La clase Arrays

- **Método toString:** Este método convierte un array en una cadena. Nos viene muy bien para mostrar un array por pantalla de forma sencilla y sin necesidad de recorrerlo.
- Ejemplo:

```
int [ ] vector = { 7 , 8 , 2 , 0 , 1 , 6 , 5 };
```

```
System.out.println(Arrays.toString( vector ));
```

/* Muestra todo el contenido del array por pantalla */

3.- La clase Arrays

The screenshot shows an IDE interface with two main panes. The top pane is titled 'ArraytoString.java' and contains Java code for creating and printing arrays. The bottom pane is titled 'Output - ArraytoString (run)' and shows the console output of the program's execution.

```
package arraytoString;
import java.util.Arrays;

public class ArraytoString {

    public static void main(String[] args) {
        int [] arrayUnidimensional = { 9 , 7 , 8 , 3 , 0 , 1 , 4 } ;
        int [ ] [ ] arrayBidimensional = { {7 , 8 , 2} , {5 , 5 , 5} , {0 , 1 , 6} } ;
        System.out.print("Array unidimensional: ");
        System.out.println(Arrays.toString(arrayUnidimensional));

        System.out.println("Array Bidimensional: ");
        for (int i = 0; i < arrayBidimensional.length; i++) {
            System.out.println(Arrays.toString(arrayBidimensional[i]));
        }
    }
}
```

Output - ArraytoString (run) ×

```
run:
Array unidimensional: [9, 7, 8, 3, 0, 1, 4]
Array Bidimensional:
[7, 8, 2]
[5, 5, 5]
[0, 1, 6]
BUILD SUCCESSFUL (total time: 0 seconds)
```

3.- La clase Arrays

- **Método equals:** Este método devuelve true si dos arrays son iguales.
- 2 Arrays son iguales si tienen:
 - Mismo tipo
 - Mismo tamaño
 - Contiene los mismos valores
- Devolverá false en caso contrario.

Ejemplo: `Arrays.equals (array1 , array2) ;`

3.- La clase Arrays

```
package arrayequals;
import java.util.Arrays;
/**
 * @author Oscar Laguna García
 */
public class ArrayEquals {

    public static void main(String[] args) {
        int [] array1 = {10,15,20,25,30};
        int [] array2 = {10,15,20,25,30};

        if (Arrays.equals(array1,array2)){
            System.out.println("Los arrays son iguales");
        }
        else{
            System.out.println("Los arrays NO son iguales");
        }
    }
}

- ArrayEquals (run)

run:
Los arrays son iguales
BUILD SUCCESSFUL (total time: 0 seconds)
```

3.- La clase Arrays

- **Método sort**: Este método ordena un array en orden ascendente.

- Ejemplo:

```
int [] vector = { 7 , 8 , 2 , 0 , 1 , 6 , 5 };
```

```
Arrays.sort ( vector ); /* Ordena todas las posiciones del  
array de forma ascendente */
```

```
Arrays.sort ( vector , 2 , 5 ); /* Ordena los valores desde el  
índice 2 hasta el índice 4 */
```

3.- La clase Arrays

```
package arraysort;
import java.util.Arrays;
/**
 * @author Oscar Laguna Garcia
 */
public class ArraySort {

    public static void mostrarArray(int [] valores){
        int i;
        for (i=0; i<valores.length;i++){
            System.out.print(" ["+i+"]->" +valores[i]);
        }
        System.out.print("\n");
    }

    public static void main(String[] args) {
        int [ ] valores1 = {7,8,2,0,1,6,5};
        int [ ] valores2 = {7,8,2,0,1,6,5};

        Arrays.sort ( valores1 ); /* Ordena todas las posiciones del array valores1 de forma ascendente */
        mostrarArray(valores1);
        Arrays.sort ( valores2 , 2 , 5 ); /* Ordena las posiciones del array valores2 desde el indice 2 hasta el indice 4 */
        mostrarArray(valores2);
    }
}

- ArraySort (run) ✘
run:
[0]->0 [1]->1 [2]->2 [3]->5 [4]->6 [5]->7 [6]->8
[0]->7 [1]->8 [2]->0 [3]->1 [4]->2 [5]->6 [6]->5
BUILD SUCCESSFUL (total time: 0 seconds)
```

3.- La clase Arrays

- **Método binarySearch:** Este método busca un valor en un array ordenado y devuelve el índice que ocupa el elemento.
- Ejemplo:

```
int [ ] vector = { 7 , 8 , 2 , 0 , 1 , 6 , 5 } ;
```

```
Arrays.sort ( vector ); /* Ordena todas las posiciones del  
array de forma ascendente */
```

```
System.out.println ( Arrays.binarySearch ( vector , 6 ) );/* Busca el  
número 6 en el array y devuelve el índice donde está. Si no  
lo encuentra, devuelve un número negativo */
```

3.- La clase Arrays

```
package arraybinarysearch;
import java.util.Arrays;
/**
 * @author Oscar Laguna Garcia
 */
public class ArrayBinarySearch {
    public static void mostrarArray(int [] valores){
        int i;
        for (i=0; i<valores.length;i++){
            System.out.print(" ["+i+"]->" +valores[i]);
        }
        System.out.print(" \n");
    }
    public static void main(String[] args) {
        int posicion;
        int [ ] valores = {7,8,2,0,1,6,5};
        Arrays.sort (valores); /* Ordena todas las posiciones del array de forma ascendente */
        mostrarArray(valores);
        posicion= Arrays.binarySearch(valores,7); /* Busca el número 7 en el array y devuelve el indice donde
                                                    está. Si no lo encuentra, devuelve un número negativo */
        if(posicion >=0){
            System.out.println ("El número 7 se encuentra en la posicion "+posicion);
        }
        else{
            System.out.println ("Elemento no encontrado");
        }
    }
}
arraybinarysearch.ArrayBinarySearch >>
-ArrayBinarySearch (run) ✘
run:
[0]->0 [1]->1 [2]->2 [3]->5 [4]->6 [5]->7 [6]->8
El número 7 se encuentra en la posicion 5
```

3.- La clase Arrays

- Todos estos métodos también se pueden utilizar con arrays de objetos, por ejemplo:
 - El método `Arrays.fill()` rellena un array, total o parcialmente con referencias a un objeto dado.
 - En caso de utilizar el método `Arrays.sort()`, la clase debe implementar la interfaz `Comparable` que nos permitirá ordenar los objetos por un atributo numérico o alfabético.
 - Veamos un ejemplo del uso de cada método:

3.- La clase Arrays

Arrays.fill()

```
package empleado;
/**
 * @author OLG
 */
public class Empleado {

    private String departamento;

    public Empleado() {
    }

    public Empleado(String departamento) {
        this.departamento = departamento;
    }

    public String getDepartamento() {
        return departamento;
    }

    public void setDepartamento(String departamento) {
        this.departamento = departamento;
    }
}
```

```
package empleado;
import java.util.Arrays;
import java.util.Scanner;
/**
 * @author OLG
 */
public class Test {

    public static String pedirDepartamento() {
        Scanner entrada = new Scanner(System.in);
        System.out.println("Introduzca el departamento: ");
        return entrada.nextLine();
    }

    public static void visualizarArray(Empleado[] aula) {
        for (int i = 0; i < aula.length; i++) {
            System.out.print("[ " + i + " ]-->depto:" + aula[i].getDepartamento() + " " );
        }
    }

    public static void main(String[] args) {

        Empleado[] empresa = new Empleado[10]; /*Creo un array llamado empresa con 10 objetos de la clase Empleado*/

        Arrays.fill(empresa, new Empleado(pedirDepartamento()));
        visualizarArray(empresa);
    }
}
```

Arrays.sort()

```
package alumn0;
/**
 * @author OLG
 */
public class Alumn0 implements Comparable<Alumn0> {

    private int numero;

    Alumn0() {
    }

    public Alumn0(int numero) {
        this.numero = numero;
    }

    public int getNumero() {
        return numero;
    }

    public void setNumero(int numero) {
        this.numero = numero;
    }

    @Override
    public int compareTo(Alumn0 alum) {
        int resultado;
        if (this.numero > alum.getNumero()) {
            resultado = 1;
        } else {
            if (this.numero < alum.getNumero()) {
                resultado = -1;
            } else {
                resultado = 0;
            }
        }
        return resultado;
    }
}
```

```
package alumn0;
import java.util.Arrays;
/**
 * @author OLG
 */
public class Test {

    public static void rellenarArray(Alumn0[] aula) {
        double aleatorio;
        int aleatorioEntero;
        for (int i = 0; i < aula.length; i++) {
            aula[i]=new Alumn0();
            aleatorio=Math.random()*9+1;
            aleatorio=Math.floor(aleatorio);
            aleatorioEntero=(int)aleatorio;
            aula[i].setNumero(aleatorioEntero);
        }
    }

    public static void visualizarArray(Alumn0[] aula) {
        for (int i = 0; i < aula.length; i++) {
            System.out.println("[ "+i+" ]-->" + aula[i].getNumero());
        }
    }

    public static void main(String [] args) {
        Alumn0[] aula = new Alumn0[10];
        rellenarArray(aula);
        visualizarArray(aula);
        Arrays.sort(aula);
        visualizarArray(aula);
    }
}
```

3.- La clase Arrays

EJERCICIOS

- **Ejercicio 03.- (OPTATIVO)** Construye un array de 30 números enteros de nombre vectorEnteros. Mediante el uso de la clase Arrays, hacer que el array almacene en las 10 primeras posiciones 10 ceros, en las 10 siguientes 10 unos, y en las 10 últimas 10 cincos. Por último, muestra el contenido del array.

EJERCICIOS

- **Ejercicio 04.- (OBLIGATORIO)** Construye un array de 6 caracteres, de nombre arrayChar. El array almacenará en la primera mitad de las posiciones el carácter a, y en la segunda mitad el carácter b, mediante el uso de la clase Arrays. Por último, muestra el contenido del array.

JAVA

ARRAYS DE OBJETOS

4. Ejercicios de consolidación

EJERCICIOS

- **Ejercicio 05.- (OPTATIVO)** Construye un array de 10 números enteros, de nombre arrayEnteros. Hacer que el array almacene en su posiciones los dígitos del 9 al 0 para mostrar después su contenido. Ordenar el array ascendentemente, mediante los métodos de la clase **Arrays**, y mostrar de nuevo su contenido.

EJERCICIOS

- **Ejercicio 06.- (OPTATIVO)** Desarrolle una aplicación en Java que determine el sueldo bruto para cada uno de los empleados de una empresa. La empresa paga la tarifa normal en las primeras 40 horas de trabajo de cada empleado, y paga tarifa y media en todas las horas trabajadas que excedan de 40.
- El programa creará los objetos que quiera el usuario (uno para cada empleado) los meterá en un array y se le pedirá al usuario que rellene la información para cada empleado.
- Por cada empleado se almacenará su nombre, el número de horas que trabajó, y la tarifa que cobra por una hora de trabajo.
- Para finalizar el programa debe determinar y mostrar el sueldo bruto de cada empleado.
- Ejemplo de ejecución: Pepe trabajó 42 horas, cobra 20 euros la hora por lo que le corresponde un sueldo de 860 euros.

EJERCICIOS

¿Cuántos empleados desea introducir?

2

- EMPLEADO 1 -

Introduzca el nombre del empleado: Pepe

¿Cuántas horas trabajó este mes? 42

¿Cuál es su tarifa por hora de trabajo? 20

EMPLEADO 1 ALMACENADO CON ÉXITO –

- EMPLEADO 2 -

Introduzca el nombre del empleado: María

¿Cuántas horas trabajó este mes?: 44

¿Cuál es su tarifa por hora de trabajo?: 15

EMPLEADO 2 ALMACENADO CON ÉXITO –

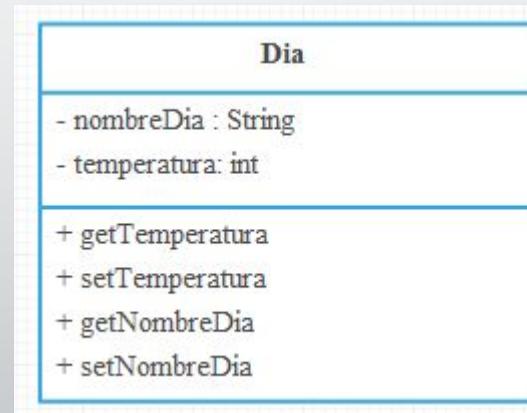
SUELDO BRUTO DE LOS EMPLEADOS

Pepe trabajó 42 horas, cobra 20 euros la hora por lo que le corresponde un sueldo de 860 euros.

Maria trabajó 44 horas, cobra 15 euros la hora, por lo que le corresponde un sueldo de 690 euros.

EJERCICIOS

- **Ejercicio 07.- (OPTATIVO)** Diseña programa que almacene las temperaturas medias de un mes. Para ello crearemos un array unidimensional de 30 posiciones relleno con objetos de la clase Día. La clase Día contiene los siguientes atributos:



EJERCICIOS

- Hasta que el usuario pulse 5, mostrar un menú que nos permita:
 1. Rellenar de forma aleatoria las temperaturas. Además el día 1 del mes no tiene porqué ser un Lunes, será un día aleatorio de la semana.
 2. Mostrar las temperaturas. Ejemplo: *Jueves día 1: 40 grados, Viernes día 2: 35 grados, Sábado día 3: 38 grados...*
 3. Visualizar la temperatura media del mes.
 4. Día o días más calurosos del mes. Ejemplo: *El día o días más calurosos fueron:*
 - *El Jueves día 1 con 40 grados.*
 - *El Sábado día 18 con 40 grados.*
 5. Salir del programa.
- *Fíjate que necesitarás otro array con el nombre de los días de la semana.*

EJERCICIOS

- **Ejercicio 08.- (OBLIGATORIO)** Realiza un programa en JAVA en el que se creen los objetos que desee el usuario y que los vaya metiendo en un array. Cada objeto contiene como atributos el nombre de un producto de una tienda de deportes, su precio y su stock.
- Un método estático le mostrará al usuario un menú de Administración, que le preguntará al usuario cuantos productos desea introducir en la tienda, y luego, mediante métodos estáticos, le irá pidiendo el nombre, el precio y el stock de cada uno.
- Realiza un método estático que le muestre al usuario un menú de Venta para que elija que producto comprar, y otro método estático que le pregunte cuantas unidades desea de él. Luego, en otro método estático se le preguntará si desea comprar otro producto o salir. Por último se le mostrará el importe total de la compra.
- Realiza otro método para actualizar el valor del stock de un producto cuando el usuario lo compre. En caso de que el usuario pida más unidades de las que quedan se le avisará por pantalla del error, se le comunicarán las unidades restantes y le preguntará si desea comprar otro producto.
- Ejemplo de ejecución:

EJERCICIOS

-- Bienvenido a mi Tienda de Deportes --

Acceso al menú de Administración:

¿Cuántos productos desea dar de alta?

2

- PRODUCTO 1 -

Introduzca el nombre del producto: Raquetas

Introduzca el precio del producto: 80

Introduzca el stock del producto: 10

PRODUCTO 1 ALMACENADO CON ÉXITO -

- PRODUCTO 2 -

Introduzca el nombre del producto: Canastas

Introduzca el precio del producto: 140

Introduzca el stock del producto: 3

- PRODUCTO 2 ALMACENADO CON ÉXITO -

Acceso al menú de Ventas:

Pulse 1 para comprar Raquetas cuyo precio es de 80 euros y el stock es de 10 unidades.
Pulse 2 para comprar Canastas cuyo precio es de 140 euros y el stock es de 3 unidades.

1

Ha elegido comprar Raquetas

¿Cuántas unidades desea?

8

Venta Realizada con éxito

¿Desea comprar otro producto?

si

Pulse 1 para comprar Raquetas cuyo precio es de 80 euros y el stock es de 2 unidades.
Pulse 2 para comprar Canastas cuyo precio es de 140 euros y el stock es de 3 unidades.

2

Ha elegido comprar Canastas

¿Cuántas unidades desea?

5

Lo sentimos, solo tenemos disponibles 3 unidades

¿Desea comprar otro producto?

si

Pulse 1 para comprar Raquetas cuyo precio es de 80 euros y el stock es de 2 unidades.
Pulse 2 para comprar Canastas cuyo precio es de 140 euros y el stock es de 3 unidades.

2

Ha elegido comprar Canastas

¿Cuántas unidades desea?

3

Venta Realizada con éxito

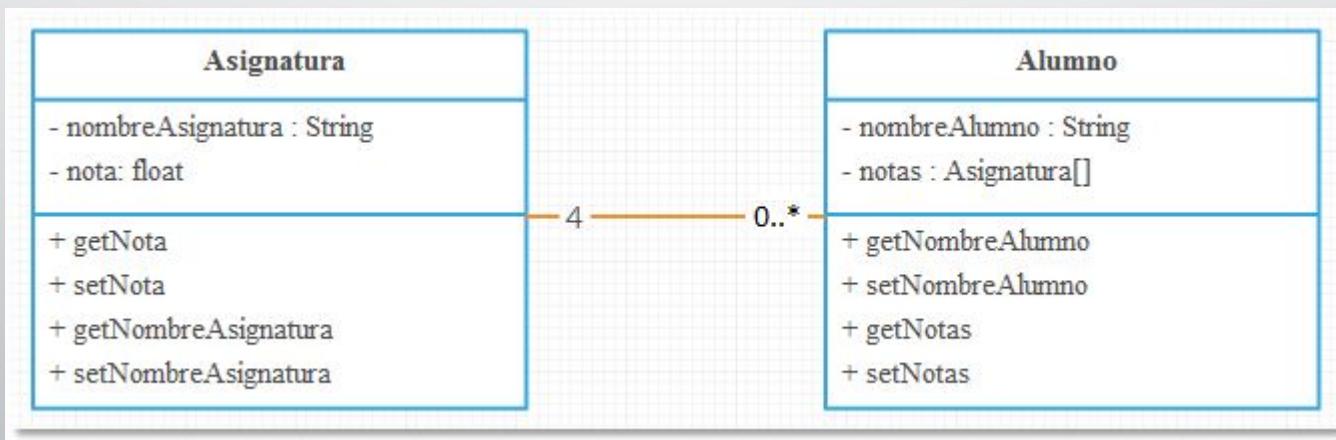
¿Desea comprar otro producto?

no

El total de su compra asciende a 1060 Euros
Muchas gracias. Vuelva cuando quiera.

EJERCICIOS

- **Ejercicio 09.- (OBLIGATORIO)** Tenemos un aula de 3 alumnos (Pepe, Juan y Marta), con 4 asignaturas cada uno (Lengua, Mates, Historia y Física).
- Para guardar la información utilizaremos un array de 3 posiciones que contiene objetos de la clase Alumno. La clase Alumno tendrá un atributo que será el nombre del alumno de tipo String, y un array de la clase Asignatura. La clase Asignatura tendrá un atributo de tipo String que será el nombre de la asignatura y otro atributo llamado nota de tipo float.



EJERCICIOS

- Realiza un programa que le dé al usuario las siguientes opciones:
 1. Rellenar las notas de los alumnos.
 2. Mostrar las notas introducidas en el punto anterior.
 3. Que nos diga que alumno es el mejor de la clase. (nota media más alta) . (Necesitarás utilizar otro array unidimensional con los nombres de los alumnos)
 4. Que nos diga el alumno con más suspensos.
 5. Que nos diga cual es la asignatura más difícil. (nota media más baja) . (Necesitarás utilizar otro array unidimensional con los nombres de las asignaturas)
 6. Salir del programa.
- Hasta que el usuario no pulse 6 no saldremos del programa y se volverá a mostrar el menú.

EJERCICIOS

- **Ejercicio 10.- (OPTATIVO)** Construye un array de 10 números enteros aleatorios y muestra su contenido mediante el método **toString** de la clase **Arrays**. Después, ordena el array descendente (de mayor a menor), **mediante el método de la burbuja**, y muestra de nuevo su contenido mediante el método **toString** de la clase **Arrays**.

EJERCICIOS

- **Ejercicio 11.- (OBLIGATORIO)** Crea una clase *Mueble* con *precio* y *descripción* como atributos. Crea constructor por defecto y parametrizado, así como *getters* y *setters*.
- Una vez creada la clase, crea en el main dos objetos de la clase mueble (crea uno de ellos con el constructor por defecto y el otro con el parametrizado) e implementa varias sentencias a tu gusto: cambia la descripción utilizando el setters, muestra la información de los objetos por pantalla, ...

EJERCICIOS

- **Ejercicio 12.- (OBLIGATORIO)** Amplía el ejercicio anterior. Crea un vector de muebles (de 4 celdas) y muestra al usuario las siguientes opciones:
 - Rellenar muebles (todos los muebles del vector).
 - Muestra muebles.
 - Muestra muebles por precio (Pediremos un precio al usuario y mostraremos todos los muebles con igual o menor precio).

EJERCICIOS

- **Ejercicio 13.- (OBLIGATORIO)** Diseña la clase *Alumno*, cuyos atributos serán *nombre*, *edad* y *notaMedia*. Crea constructores por defecto, parametrizado, *getters* y *setters*.
- En el main, crea un par de objetos de la clase Alumno.
- Completa el main a tu gusto.

EJERCICIOS

- **Ejercicio 14.- (OBLIGATORIO)** Amplía el ejercicio anterior. Crea un vector de 5 alumno y muestra al usuario las siguientes opciones:
 - Rellenar un alumno (Pediremos posición e insertaremos al nuevo alumno en esa posición. Si la posición está ya rellena, volveremos a pedir una nueva posición las veces que sean necesarias).
 - Muestra vector de alumnos.(Si una celda todavía no está rellena, no mostraremos nada de esa celda)
 - Muestra alumnos con nota media por encima de una nota dada.
 - Muestra cuántos alumnos hay con nota media suspensa.
 - Buscar alumno (pediremos nombre y mostraremos si está matriculado o si no está matriculado).

EJERCICIOS

- **Ejercicio 15.- (OBLIGATORIO)** La Asociación "Nos gusta el cine" nos ha pedido que desarrollemos un software antes de final de mes. En un principio desarrollaremos una primera versión del software, donde la Asociación quiere almacenar las películas que proyectan (utilizaremos un vector de películas) junto con los socios que asisten a las proyecciones.
- De cada película almacenaremos su título, coste de la licencia y un vector de socios, donde almacenaremos los socios que acuden a la proyección (En esa primera versión, vamos a suponer que el aforo de la sala de proyección es de 4)
- De los socios almacenaremos los siguientes datos: nombre y precio abonado (al ser una Asociación, los socios aportan "la voluntad").

EJERCICIOS

- El menú del software tendrá las siguientes opciones:
 1. Rellenar las películas junto con los socios que han acudido a verla.
 2. Mostrar las películas y los socios que la han visto.
 3. Mostrar película más rentable (el beneficio neto de una película será la venta de entradas menos el coste de la licencia).
 4. Mostrar película menos rentable
 5. Pedir el nombre de una película y mostrar el beneficio neto y toda la información de los socios que han asistido a verla.
 6. Contar el número de socios que han abonado una cantidad mayor a la pedida por pantalla.
 7. Salir del programa.

EJERCICIOS

- Hasta que no pulsemos 7 no saldremos del programa y se volverá a mostrar el menú.
- NOTA IMPORTANTE: El beneficio neto de una película será la venta de entradas menos el coste de la licencia.
- Y RECUERDA: El programa principal debe ser un "esquema" de lo que hace un programa.

EJERCICIOS

- **Ejercicio 16.- (OBLIGATORIO)** Añade al ejercicio 15 las siguientes opciones:
 - Mostrar clientes que empiecen por una letra introducida por teclado.
 - Mostrar películas que empiecen por un texto indicado (Dicho texto se le pedirá al usuario)
 - Modificar las 'a' por 'e' para la última película.
 - Convertir los títulos de todas las películas a mayúsculas.

EJERCICIOS

- **Ejercicio 17.- (OBLIGATORIO)** Implementa en JAVA el juego del 3 en Raya orientado a objetos. Se puede hacer de muchas maneras, esta es la que propongo yo:
 - Una clase Tablero, que contendrá un array de 3x3 de objetos de la clase Ficha.
 - Una clase Casilla, que contendrá un atributo **ficha** de tipo String ("O" o "X") y otro atributo de tipo booleano llamado **ocupada**.
 - Un método, de la clase Tablero, se encargará de dibujar el tablero con las fichas que ya hayan sido colocadas.
 - Método estático para pedirle al jugador la posición donde colocar la ficha.
 - Método estático que permitirá que el ordenador coloque una ficha al azar.
 - Método de la clase Tablero comprobará si el jugador que acaba de mover ha ganado la partida.
 - Por último, un método de la clase Tablero comprobará si la partida a terminado en empate.
 - Ejemplo de ejecución:

EJERCICIOS

- JUGADOR1 -

Elige una fila: 2

Elige una columna: 2

		O	

- TURNO DE LA CPU-

Fila: 1

Columna 3

			X	
		O		

- JUGADOR1 -

Elige una fila: 1

Elige una columna: 2

		O		X	
		O			

- TURNO DE LA CPU-

Fila: 3

Columna 1

		O		X	
		O			
	X				

- JUGADOR1 -

Elige una fila: 3

Elige una columna: 2

		O		X	
		O			
	X				

JUGADOR 1
HAS
GANADO!!!!