

# JAVA



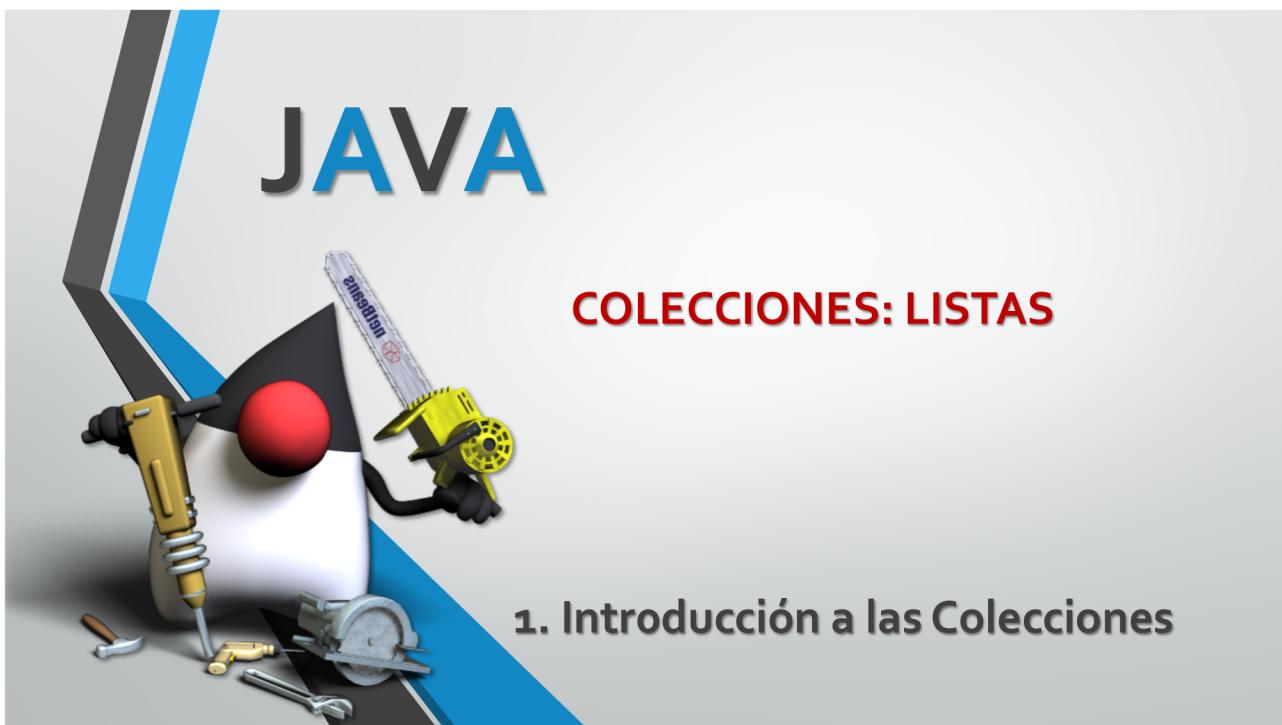
## TEMA 10: COLECCIONES: LISTAS



## ÍNDICE

1. Introducción a las Colecciones
2. Introducción a las Listas
3. Creación de listas
4. Métodos para el manejo de listas
5. Lista de listas
6. Listas de objetos definidos por el usuario
7. Recorrer listas con Iteradores
8. Ejercicios de consolidación





## 1. Introducción a las Colecciones

UNION EUROPEA  
Fondo Social Europeo  
"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.

V IES Valle del Jerte Bilingual Section IE

## 1.- Introducción a las Colecciones

- Las colecciones son una especie de arrays de tamaño dinámico. Para usarlas haremos uso del Java Collections Framework (JCF), el cual contiene un conjunto de clases e interfaces del paquete `java.util` para gestionar colecciones de objetos.
- En Java las principales interfaces que disponemos para trabajar con colecciones son: `Collection`, `Set`, `List`, `Queue` y `Map`:

Oscar Laguna García  
Ana M. Arribas Arjona

4  
PROGRAMACIÓN



## 1.- Introducción a las Colecciones

- Collection<E>: Un grupo de elementos individuales, frecuentemente con alguna regla aplicada a ellos.
- List<E>: Elementos en una secuencia particular que mantienen un orden y permite duplicados. La lista puede ser recorrida en ambas direcciones con un ListIterator. Hay 3 tipos de constructores:
  - **ArrayList<E>**: Su ventaja es que el tiempo acceso a un elemento en particular es ínfimo. Su desventaja es que para eliminar un elemento, se ha de mover toda la lista para eliminar ese "hueco".
  - **Vector<E>**: Es igual que ArrayList, pero sincronizado. Es decir, que si usamos varios hilos, no tendremos de qué preocuparnos hasta cierto punto.
  - **LinkedList<E>**: En esta, los elementos están conectados con el anterior y el posterior. La ventaja es que es fácil mover/eliminar elementos de la lista, simplemente moviendo/eliminando sus referencias hacia otros elementos. La desventaja es que para usar el elemento N de la lista, debemos realizar N movimientos a través de la lista.



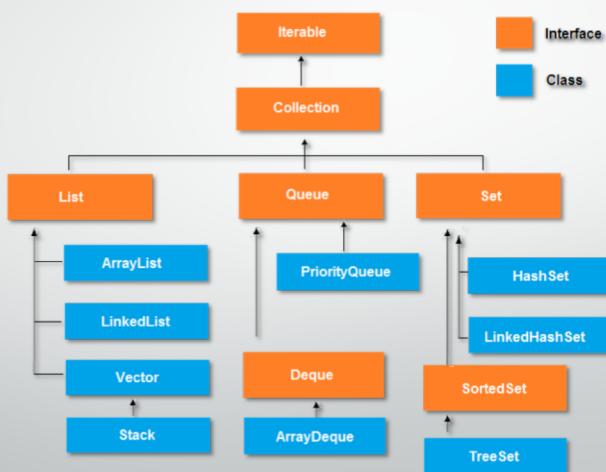
## 1.- Introducción a las Colecciones

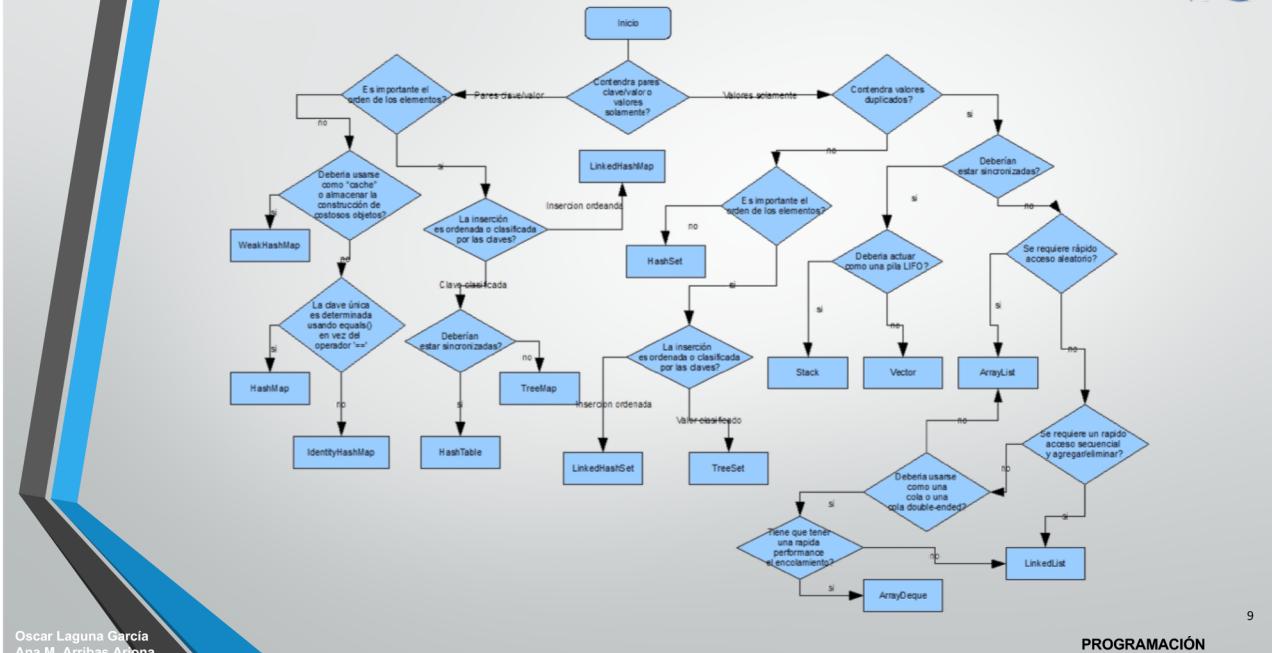
- Set<E>: No puede haber duplicados. Cada elemento debe ser único, por lo que si existe uno duplicado, no se agrega. Por regla general, cuando se redefine equals(), se debe redefinir hashCode(). Es necesario redefinir hashCode() cuando la clase definida será colocada en un HashSet. Los métodos add(o) y addAll(o) devuelven false si o ya estaba en el conjunto.
- Queue<E>: Colección ordenada con extracción por el principio e inserción por el principio (LIFO – Last Input, First Output) o por el final (FIFO – First Input, First Output). Se permiten elementos duplicados. No da excepciones cuando la cola está vacía/llena, hay métodos para interrogar, que devuelven null. Los métodos put()/take() se bloquean hasta que hay espacio en la cola/haya elementos.

## 1.- Introducción a las Colecciones

- Map<K,V>: Un grupo de pares objeto clave-valor, que no permite duplicados en sus claves. Es quizás el más sencillo, y no utiliza la interfaz Collection. Los principales métodos son: put(), get(), remove().
  - HashMap<K,V>: Se basa en una tabla hash, pero no es sincronizado.
  - HashTable<K,V>: Es sincronizado, aunque que no permite null como clave.
  - LinkedHashMap<K,V>: Extiende de HashMap y utiliza una lista doblemente enlazada para recorrerla en el orden en que se añadieron. Es ligeramente más rápida a la hora de acceder a los elementos que su superclase, pero más lenta a la hora de añadirlos.
  - TreeMap<K,V>: Se basa en una implementación de árboles en el que ordena los valores según las claves. Es la clase más lenta.

## 1.- Introducción a las Colecciones



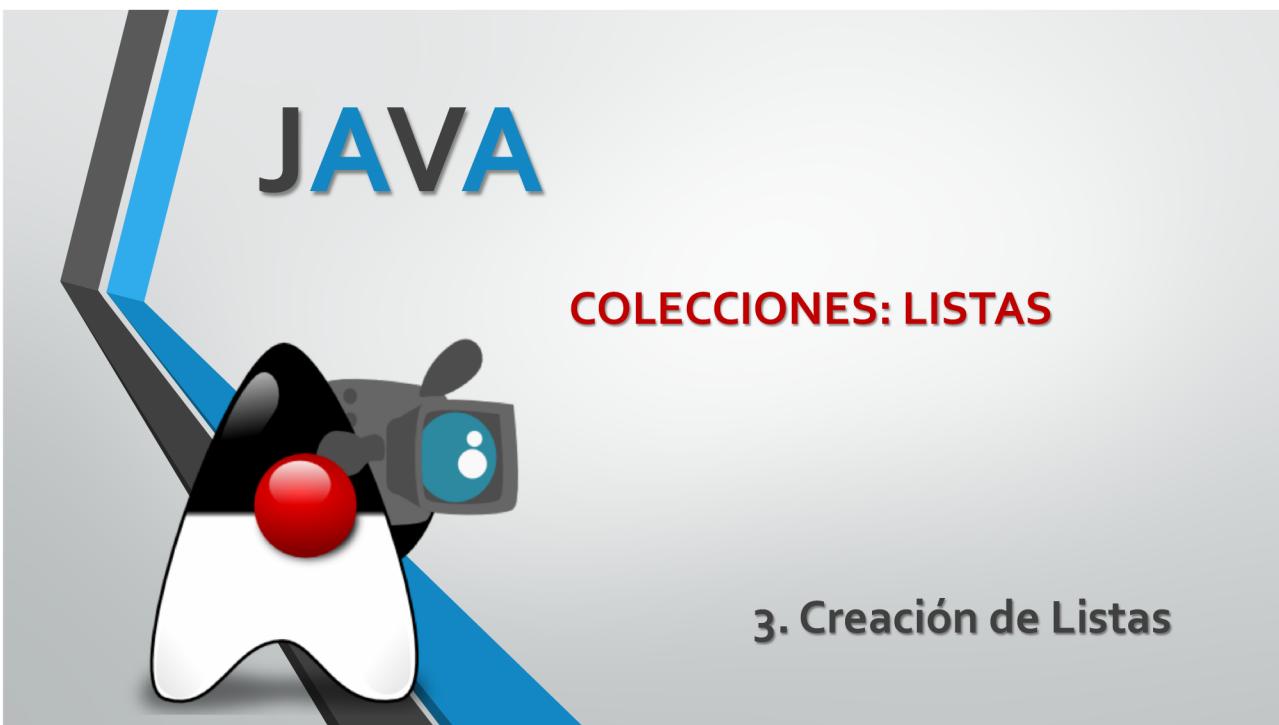


## 2.- Introducción a las Listas

- La biblioteca de JAVA nos proporciona una clase genérica llamada **ArrayList** (definida en el paquete `java.util`) que nos facilita la creación y manipulación de listas **de objetos**.
- A diferencia de los Arrays primitivos, las Listas tienen la ventaja de que el numero de elementos que almacenan lo hacen de forma dinámica, es decir, que no es necesario declarar su tamaño de antemano, como pasaba con los Arrays.

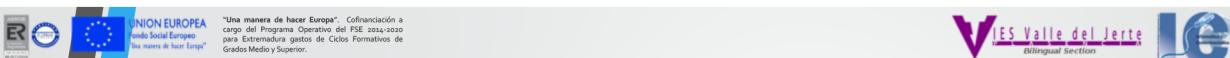
## 2.- Introducción a las Listas

- En el manejo de Listas hay una clara ventaja de JAVA frente a otros lenguajes de programación. Las operaciones que nos permitirán añadir, eliminar y modificar elementos se hacen de forma transparente para el programador; simplemente bastará con llamar al método correspondiente que realiza la operación que deseemos.



## COLECCIONES: LISTAS

### 3. Creación de Listas



UNIÓN EUROPEA  
Fondo Social Europeo  
"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.

V IES Valle del Jerte Bilingual Section IE

### 3.- Creación de Listas

- Para definir una Lista de objetos utilizamos la siguiente sintaxis:  
**ArrayList <tipo> nombre = new ArrayList <tipo>();**  
donde tipo deberá ser una clase.
- Como las listas solo pueden contener objetos, si queremos declarar una lista de enteros, tendremos que usar las clases envoltorio, por que lo haremos de la siguiente manera:  
**ArrayList <Integer> lista = new ArrayList <Integer>();**

Oscar Laguna García  
Ana M. Arribas Arjona

14  
PROGRAMACIÓN

### 3.- Creación de Listas

- A partir de la versión 7 de JAVA se permite que en el lado derecho de la asignación no se tengan que escribir el tipo, de los objetos que forman la lista, de nuevo.

**ArrayList <tipo> nombre = new ArrayList <tipo>();**

Por ejemplo: **ArrayList <String> lista = new ArrayList <>();**

### 3.- Creación de Listas

- Recuerda el que paquete java.lang nos proporciona las clases envoltorio: Byte, Character, Short, Integer, Long, Float, Double y Boolean, que encapsulan cada uno de los tipos primitivos.
- Tanto cuando insertamos un elemento a una lista, como cuando lo obtenemos, JAVA encapsula directamente un tipo primitivo (como int) a un objeto de la clase que lo encapsula (como Integer). Ejemplo:

```
int x = 123;  
lista.add(x); //añade el elemento x a la lista. Gracias al Autoboxing ya  
//no es necesario hacerlo así: lista.add(new Integer(x));
```

### 3.- Creación de Listas

- No es recomendable y está mal visto, pero también podemos definir una colección ArrayList sin indicar el tipo:

**ArrayList nombre = new ArrayList();**

- Una Lista declarada así ocupa más espacio en memoria, pero puede contener objetos de cualquier tipo, e incluso puede contener objetos de tipos distintos mezclados.
- Veamos un ejemplo:

### 3.- Creación de Listas

```
ArrayList listaVariada = new ArrayList();
listaVariada.add("Lenguaje");
listaVariada.add(3);
listaVariada.add('a');
listaVariada.add(23.5);
```

- Los elementos del ArrayList *listaVariada* serían: "Lenguaje", 2, 'a', y 23.5.



## 4. Métodos para el manejo de Listas

UNIÓN EUROPEA  
Fondo Social Europeo  
"Una manera de hacer Europa"  
Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.

VIES Valle del Jerte Bilingual Section IE

## 4.- Métodos para el manejo de Listas

- Añadir un elemento: Para añadir a un elemento al final de la lista disponemos del método **add(elemento)**. Por ejemplo:

```
package listaenteros;
import java.util.ArrayList;
/*
 * @author Oscar Laguna García
 */
public class ListaEnteros {

    public static void main(String[] args) {
        ArrayList <Integer> lista = new ArrayList <Integer>();
        int x,y;
        int i;
        x= 12;
        y=56;
        lista.add(x); //Añade el elemento x en la primera posición de la lista
        lista.add(y); //Añade el elemento x en la segunda posición de la lista
        for(i=0;i<lista.size();i++){ //size nos dice el número de los elementos de la lista
            System.out.println("Elemento " + i + ": " + lista.get(i));
        }
    }
}
```

- ListaEnteros (run) X

run:  
Elemento 0: 12  
Elemento 1: 56  
BUILD SUCCESSFUL (total time: 0 seconds)

Oscar Laguna García  
Ana M. Arribas Arjona

20

PROGRAMACIÓN

## 4.- Métodos para el manejo de Listas

- **Insertar un elemento:** Para insertar un elemento en una posición concreta de la lista disponemos del método **add (posición, elemento)**. Ejemplo:

```

package listaenteros;
import java.util.ArrayList;
/*
 * @author Oscar Laguna García
 */
public class ListaEnteros {
    public static void main(String[] args) {
        ArrayList <Integer> lista = new ArrayList <Integer>();
        int x,y,z;
        int i;
        x=12;
        y=55;
        z=33;
        lista.add(x); //Añade el elemento x en la primera posición de la lista
        lista.add(y); //Añade el elemento x en la segunda posición de la lista
        lista.add(i, z);
        for(i=0;i<lista.size();i++){ // .size nos dice el número de los elementos de la lista
            System.out.println("Elemento " + i + ": " + lista.get(i));
        }
    }
}
  
```

21

PROGRAMACIÓN

## 4.- Métodos para el manejo de Listas

- **Tamaño:** Para conocer el tamaño de una lista hay que invocar al método **size ()**.
- Para saber si una lista está vacía, utilizamos el método **isEmpty**. Este método devuelve true si la lista está vacía y false en caso contrario.
- Extraer: Para obtener el valor de una posición de la lista utilizamos el método **get (posicion)**

22

PROGRAMACIÓN

## 4.- Métodos para el manejo de Listas

- Eliminar elementos:

- El método **clear** permite eliminar todos los elementos de una lista.
- El método **remove** permite eliminar un elemento en concreto de una lista, o una posición de la lista.

- Ejemplos:

```
lista.remove (i) // elimina el elemento de índice i
```

```
lista.remove (new Integer(x)); // elimina la primera ocurrencia  
// de la lista en la que el elemento tiene el valor x
```

```
lista.clear ( ); //elimina todos los elementos
```

## 4.- Métodos para el manejo de Listas

```
package listaenteros;  
import java.util.ArrayList;  
/**  
 * @author Oscar Laguna García  
 */  
public class ListaEnteros {  
  
    public static void main(String[] args) {  
        ArrayList <Integer> lista = new ArrayList <Integer>();  
        int x=11,y=22,z=33;  
        lista.add(x);  
        lista.add(y);  
        lista.add(z);  
        System.out.println(lista); // [11, 22, 33]  
        lista.remove(new Integer(22));  
        System.out.println(lista); // [11, 33]  
        lista.remove(1);  
        System.out.println(lista); // [11]  
        lista.clear();  
        System.out.println(lista); // []  
    }  
}
```

## 4.- Métodos para el manejo de Listas

- Modificar un elemento: Para modificar un elemento determinado de la lista reemplazándolo por otro disponemos del método **set (posición, elemento)**. Ejemplo:

```
package listacadenas;
import java.util.ArrayList;
/**
 * @author Oscar Laguna García
 */
public class ListaCadenas {

    public static void main(String[] args) {
        ArrayList <String> lista = new ArrayList <String>();
        String x,y;
        int i;
        x="hola";
        y="adios";
        lista.add(x);
        lista.add(y);
        System.out.println(lista); // [hola, adios]
        lista.set(0,"¿Qué tal?");
        System.out.println(lista); // [¿Qué tal?, adios]
    }
}
```

25

PROGRAMACIÓN

## 4.- Métodos para el manejo de Listas

- Buscar un elemento: El método **contains (elemento)** devuelve true si la lista contiene el elemento especificado.
- Los métodos **indexOf (elemento)** y **lastIndexOf (elemento)** devuelven, respectivamente, la posición de la primera y de la última ocurrencia del elemento en la lista. Si no encuentran el elemento devolverán -1. Ejemplo:

26

PROGRAMACIÓN



UNION EUROPEA  
Fondo Social Europeo  
Una manera de hacer Europa

"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## 4.- Métodos para el manejo de Listas

```
package listaenteros;
import java.util.ArrayList;
/**
 * @author Oscar Laguna García
 */
public class ListaEnteros {

    public static void main(String[] args) {
        ArrayList <Integer> lista = new ArrayList <Integer>();
        int x=11,y=22,z=33,a=22,b=44,c=55;
        lista.add(x);
        lista.add(y);
        lista.add(z);
        lista.add(a);
        lista.add(b);
        lista.add(c);
        System.out.println(lista); // [11, 22, 33, 22, 44, 55]
        if (lista.contains(22)){
            System.out.println("La lista contiene el elemento 22");
            System.out.println("La primera ocurrencia esta en la posición: "+lista.indexOf(22));
            System.out.println("La última ocurrencia esta en la posición: "+lista.lastIndexOf(22));
        }
    }
}
```

Oscar Laguna García  
Ana M. Arribas Arjona

27

PROGRAMACIÓN



UNION EUROPEA  
Fondo Social Europeo  
Una manera de hacer Europa

"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## 4.- Métodos para el manejo de Listas

- Copiar listas: Una lista puede ser copiada en otra invocando a su método **clone()**:

```
package listaenteros;
import java.util.ArrayList;
/**
 * @author Oscar Laguna García
 */
public class ListaEnteros {

    public static void main(String[] args) {
        ArrayList <Integer> lista = new ArrayList <Integer>();
        int x=11,y=22,z=33;
        lista.add(x);
        lista.add(y);
        lista.add(z);
        System.out.println(lista); // [11, 22, 33]
        ArrayList lista2 = (ArrayList) lista.clone(); // Clonamos lista en lista2
        System.out.println(lista2); // [11, 22, 33]
    }
}
```

Oscar Laguna García  
Ana M. Arribas Arjona

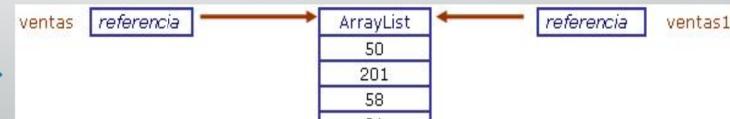
28

PROGRAMACIÓN

## 4.- Métodos para el manejo de Listas

- **Nota Importante:** Si utilizamos el símbolo igual (=) en vez del método **clone ()**, no copiamos una lista en otra sino que creamos un alias, esto es, otro nombre con el cual también acceder al ArrayList. Ejemplo:

```
ArrayList<Integer> ventas = new ArrayList<Integer>();  
ArrayList<Integer> ventas1 = new ArrayList<Integer>();  
ventas.add(50);  
ventas.add(201);  
ventas.add(58);  
...  
ventas1 = ventas;
```



## 4.- Métodos para el manejo de Listas

- **Volcar una lista a un array:** Para pasar una lista a un array utilizaremos el método **toArray ()**:

```
package listaenteros;  
import java.util.ArrayList;  
/*  
 * @author Oscar Laguna García  
 */  
public class ListaEnteros {  
  
    public static void main(String[] args) {  
        ArrayList<Integer> lista = new ArrayList<Integer>();  
        int x=11,y=22,z=33;  
        lista.add(x);  
        lista.add(y);  
        lista.add(z);  
        System.out.println(lista); // [11, 22, 33]  
        Object [] array = lista.toArray();  
        for (int i=0;i<array.length;i++){  
            System.out.println(array[i]); // 11, 22, 33  
        }  
    }  
}
```

## 4.- Métodos para el manejo de Listas

- También podemos pasar listas a los métodos (por referencia), al igual que una lista podría ser devuelta por un método mediante la instrucción return.
- Un ejemplo:

```
1 package listas;
2 import java.util.ArrayList;
3 /**
4  * @author OLG
5 */
6 public class Listas {
7
8     public static void rellenarLista(ArrayList<Integer> lista) {
9         for (int i = 0; i < 10; i++) {
10             lista.add(i);
11         }
12     }
13
14     public static void mostrarLista(ArrayList<Integer> lista) {
15         for (int i = 0; i < 10; i++) {
16             System.out.println(lista.get(i));
17         }
18     }
19
20     public static void main(String[] args) {
21         ArrayList<Integer> lista = new ArrayList<Integer>();
22         rellenarLista(lista);
23         mostrarLista(lista);
24     }
25 }
```

31

PROGRAMACIÓN

## EJERCICIOS

- **Ejercicio 01.- (OPTATIVO)** Crea una lista en la que almacenes nombres de personas y luego la muestres por pantalla.
- Para ello, crea un método para llenar la lista, en el que le vayas pidiendo al usuario el nombre de la persona a introducir y si desea introducir otro nombre.
- Crea otro método para mostrar la lista por pantalla.

32

PROGRAMACIÓN



"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

- **Ejercicio 02.- (OPTATIVO)** Crea un programa que calcule el mayor, el menor y la suma de todos los elementos de una lista que contenga números enteros positivos.
- El programa permitirá, mediante un método, que el usuario introduzca valores hasta que introduzca un valor negativo.
- Otro método visualizará los elementos de la lista
- Por último, se mostrará el mayor, el menor y la suma de los elementos (implementado en 3 métodos independientes)



"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

- **Ejercicio 03.- (OPTATIVO)** Realiza un programa que lea tantos números enteros como desee el usuario (le irás preguntando si desea introducir más números) y los introduzca en una lista.
- A continuación, muestras la lista, luego intercambias los números que se encuentren en la 2<sup>a</sup> y 4<sup>a</sup> posición, y por último muestras de nuevo la lista por pantalla.
- Utiliza al menos 3 métodos: uno para introducir los datos, otro para mostrar los datos y otro para intercambiar los datos.



UNION EUROPEA  
Fondo Social Europeo  
"Una manera de hacer Europa"

"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

- **Ejercicio 04.- (OPTATIVO)** Escribe un programa que contenga un método que acepte como parámetro una lista de números enteros mayores que 0, pudiendo contener elementos duplicados. Este método debe sustituir cada valor repetido por 0.
- Otro método se encargará de mostrar la lista antes y después de ser modificada.
- También necesitarás otro método para llenar la lista de enteros. Este método le irá pidiendo números al usuario hasta que este introduzca un número negativo.

Ejemplo: 2 7 8 4 5 8 7 1 8 → 2 0 0 4 5 0 0 1 0



UNION EUROPEA  
Fondo Social Europeo  
"Una manera de hacer Europa"

"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

- **Ejercicio 05.- (OPTATIVO)** Realiza un programa en JAVA que lea por teclado números enteros (los que quiera el usuario) y los introduzca en una lista.
- Tras mostrar la lista por pantalla, calcularás cual es el mayor número par y también el menor número impar de la lista, los muestras por pantalla, intercambias sus posiciones en la lista y muestras el nuevo la lista por pantalla.
- Crea todos los métodos que creas conveniente; cuantos más, mejor.



"Una manera de hacer Europa". Colaboración a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medios y Superior.



## EJERCICIOS

- **Ejercicio 06.- (OBLIGATORIO)** Realiza un programa en el que utilices una lista de enteros y que muestre un menú en el que se le ofrezcan al usuario las siguientes opciones:
  1. Introducir las ventas de coches de cada uno de los 12 meses del año.
  2. Mostrar las ventas introducidas en el punto anterior.
  3. Mostrar las ventas introducidas al revés.
  4. Que muestre la suma total de ventas del año.
  5. Que muestre las ventas totales de los meses cuyo nombre contenga la letra a.
  6. Que muestre el nombre del mes (o meses) con más ventas.
  7. Salir del programa.
- Necesitarás, un array unidimensional (o una lista) con el nombre de los meses del año.
- Controlaremos que el usuario elija una opción del menú entre 1 y 7. Hasta que el usuario no pulse 7 no saldremos del programa.

Oscar Laguna García  
Ana M. Arribas Arjona

37

PROGRAMACIÓN

# JAVA

## COLECCIONES: LISTAS

### 5. Lista de Listas



## 5.- Lista de listas

- Una Lista es unidimensional pero, al igual que los arrays podían tener varias dimensiones, aquí podemos crear listas de dos o más dimensiones anidando ArrayLists.
- Para crear una lista de listas crearemos un ArrayList cuyos elementos sean a su vez ArrayList. Esto se puede extender sucesivamente y obtener listas de más dimensiones.
- Ejemplo:

```
ArrayList<ArrayList<Integer>> lista = new ArrayList<ArrayList<Integer>>();
```

## 5.- Lista de listas

```
package listadelistas;
import java.util.Scanner;
import java.util.ArrayList;
/**
 * @author Oscar Laguna García
 */
public class ListaDeListas {

    public static void rellenarNotas(ArrayList<ArrayList<Integer>> lista){
        Scanner entrada = new Scanner(System.in);
        int numalumnos = 3; //número de alumnos
        int i, nota, cuentaNota;
        System.out.println("Introduzca la notas. Nota menor que 0 para acabar");
        for(i=0;i<numAlumnos;i++){
            lista.add(new ArrayList<Integer>()); //para cada alumno se añade una nueva fila vacía
            cuentaNota = 1;//Para mostrar el número de nota
            System.out.println("Alumno " + (i+1) + ": ");
            System.out.print("Nota " + cuentaNota + ": ");
            nota = entrada.nextInt();
            while(nota>=0){
                lista.get(i).add(nota); //añade la nota en la posición i de la lista
                cuentaNota++;
                System.out.print("Nota " + cuentaNota + ": ");
                nota = entrada.nextInt();
            }
        }
    }
}
```

## 5.- Lista de listas

```
public static void mostrarNotas(ArrayList<ArrayList<Integer>> lista){  
    int i,j;  
    System.out.println("Notas de alumnos");  
    for(i=0;i<lista.size();i++){ //para cada alumno (para cada fila)  
        System.out.print("Alumno " + i + ": ");  
        for(j=0;j<lista.get(i).size();j++){ //recorre todas la columnas de la fila  
            System.out.print(lista.get(i).get(j) + " "); //se obtiene el elemento i,j  
        }  
        System.out.println();  
    }  
}  
  
public static void main(String args[]){  
    ArrayList<ArrayList<Integer>> lista = new ArrayList<ArrayList<Integer>>();  
    llenarNotas(lista);  
    mostrarNotas(lista);  
}
```

## EJERCICIOS

- **Ejercicio 07.- (OBLIGATORIO)** Diseña programa que almacene, en una lista de listas de enteros (2 dimensiones), las temperaturas medias de un mes que introduzca un usuario. Para hacerlo más sencillo vamos a suponer que el mes tiene 28 días y está formado por 4 semanas de 7 días. Hasta que el usuario pulse 5, mostrar un menú que nos permita:
  1. Rellenar las temperaturas.
  2. Mostrar las temperaturas.
  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 de la Semana 3 con 40 grados.*
    - *El Sábado de la Semana 4 con 40 grados.*
  5. Salir del programa.
- *Fíjate que necesitarás un array (o una lista) con el nombre de los días de la semana.*



6.- Lista de objetos definidos por el usuario

- Hemos visto como una lista almacena objetos de las clases Integer, Character, String...
- Por lo tanto, no habría ningún problema en que almacene objetos de clases definidas por el usuario.
- Veamos un sencillo ejemplo:

Oscar Laguna García  
Ana M. Arribas Arjona

UNION EUROPEA  
Fondo Social Europeo  
"Una manera de hacer Europa"  
Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.

VIES Valle del Jerte  
Bilingual Section

IE

44

PROGRAMACIÓN

## 6.- Lista de objetos definidos por el usuario

45

PROGRAMACIÓN

## EJERCICIOS

- **Ejercicio 08.- (OPTATIVO)** Crea una lista en la que almacenes objetos Alumno que tengan como atributos el nombre del alumno y el curso al que pertenecen. Para ello le pedirás los datos al usuario, que introducirá los alumnos que él desee.
  - Para finalizar, muestra la lista de alumnos por pantalla.
  - Al menos, utiliza un método para llenar la lista (en el que le vayas pidiendo al usuario el nombre y la edad de la persona a introducir) y otro método para mostrar la lista por pantalla.

46

PROGRAMACIÓN



"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

- **Ejercicio 09.- (OBLIGATORIO)** Diseña programa que almacene las temperaturas medias de un mes. Para ello crearemos una lista de 31 posiciones rellena con objetos de la clase Día. La clase Día contiene los siguientes atributos:

Dia
- nombreDia : String
- temperatura: int
+ getTemperatura
+ setTemperatura
+ getNombreDia
+ setNombreDia



"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## 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 un array con el nombre de los días de la semana.*



"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

- **Ejercicio 10.- (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 una lista 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 la lista 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*

Oscar Laguna García  
Ana M. Arribas Arjona

PROGRAMACIÓN

49



"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

- **Ejercicio 11.- (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 una lista 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:

Oscar Laguna García  
Ana M. Arribas Arjona

PROGRAMACIÓN

50



"Una manera de hacer Europa". Colaboración a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

\*\*\* Se procede a llenar la lista de empleados: \*\*\*

- 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 –

¿Desea introducir otro empleado?

**SI**

- EMPLEADO 2 -

Introduzca el nombre del empleado: **Maria**

¿Cuántas horas trabajó este mes?: **44**

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

EMPLEADO 2 ALMACENADO CON ÉXITO –

¿Desea introducir otro empleado?

**NO**

\*\*\* 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**.

Oscar Laguna García  
Ana M. Arribas Arjona

51

PROGRAMACIÓN

# JAVA

## COLECCIONES: LISTAS



### 7. Recorrer listas con Iteradores

## 7.- Recorrer listas con Iteradores

- Además de utilizar bucles for, otra forma de trabajar con los ArrayList son los “Iteradores” (Iterator).
- Un iterador no es más que una referencia a una colección, y los utilizaremos para recorrer los ArrayList y poder trabajar con ellos.
- Los Iteradores tienen tres métodos que son:
  - hasNext(): Comprueba que siguen quedando elementos en el iterador.
  - next(): Devuelve el siguiente elemento del iterador.
  - remove(): Elimina un elemento del iterador. ¡Ojo!, si eliminas un elemento del iterador lo estás eliminando de la lista también, ya que el iterador hace referencia a la lista.

## 7.- Recorrer listas con Iteradores

```
Cliente.java
package cliente;
/*
 * Author OLG
 */
public class Cliente {

    private String nombre;
    private int edad;

    public Cliente() {
    }

    public Cliente(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }
}
```

```
Test.java
import cliente.Cliente;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
/*
 * Author OLG
 */
public class Test {

    public static String pedirNombre() {
        Scanner entrada = new Scanner(System.in);
        System.out.println("Introduce el nombre del empleado");
        return entrada.nextLine();
    }

    public static int pedirEdad() {
        Scanner entrada = new Scanner(System.in);
        System.out.println("Introduce su edad");
        return entrada.nextInt();
    }

    public static void main(String[] args) {
        ArrayList<Cliente> clientes = new ArrayList<Cliente>();
        clientes.add(new Cliente(pedirNombre(), pedirEdad()));
        System.out.println("Has añadido otro cliente?");
        while (entrada.nextLine().equalsIgnoreCase("si")) {
            System.out.println("Datos del cliente:");
            clientes.add(new Cliente(pedirNombre(), pedirEdad()));
            System.out.println("Has añadido otro cliente?");
        }
    }
}
```

Mismo ejemplo que antes, pero ahora el método mostrarClientes lo hago con un iterador llamado it.

```
public static void mostrarClientes(ArrayList<Cliente> clientes) {
    Iterator<Cliente> it = clientes.iterator();
    System.out.println("Procedemos a visualizar la lista de clientes");
    while (it.hasNext()) {
        Cliente aux = it.next();
        System.out.println("Nombre: " + aux.getNombre());
        System.out.println("Edad: " + aux.getEdad());
        // Utilizariamos it.remove(); si quisiersemos eliminar el elemento
    }
}

public static void main(String[] args) {
    ArrayList<Cliente> clientes = new ArrayList();
    rellenarClientes(clientes);
    mostrarClientes(clientes);
}
```



UNION EUROPEA  
Fondo Social Europeo  
"Una manera de hacer Europa"

"Una manera de hacer Europa". Colaboración a  
cargo del Programa Operativo del FSE 2014-2020  
para Extremadura gastos de Ciclos Formativos de  
Grados Medio y Superior.



## EJERCICIOS

- **Ejercicio 12.- (OBLIGATORIO)** Crea una lista en la que almacenes nombres de personas y luego, UTILIZANDO ITERADORES, la muestres por pantalla.
- Para ello, crea un método para llenar la lista, en el que le vayas pidiendo al usuario el nombre de la persona a introducir y si desea introducir otro nombre.
- Crea otro método para mostrar la lista por pantalla.

Oscar Laguna García  
Ana M. Arribas Arjona

55

PROGRAMACIÓN

# JAVA

## COLECCIONES: LISTAS



### 8. Ejercicios de consolidación



"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

- **Ejercicio 13.- (OBLIGATORIO)** Realiza un programa en el almacenes una lista de objetos de la clase VENTA cuyos atributos son 2:
  - Nombre de Mes.
  - Ventas de coches del mes.
- Se mostrará un menú en el que se le ofrezcan al usuario las siguientes opciones:
  1. Introducir las ventas de coches de cada uno de los meses del año .
  2. Mostrar las ventas introducidas en el punto anterior.
  3. Que muestre la suma total de ventas de coches del año.
  4. Que muestre las ventas totales de los meses que empiezan por la letra A. (Utiliza el método correspondiente de la clase String)
  5. 5.- Que muestre el nombre del mes con más ventas.
  6. 6.- Salir del programa.
- Controlaremos que el usuario elija una opción del menú entre 1 y 6.
- Hasta que el usuario no pulse 6 no saldremos del programa.



"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

- **Ejercicio 14.- (OPTATIVO)** Realiza un programa en el almacenes una lista de objetos de la clase ALUMNO cuyos atributos son 2:
  - Nombre del Alumno. (String)
  - asignaturas (Array de la Clase Asignatura)
- A su vez, la clase Asignatura está formada por:
  - Nombre de la asignatura. (String)
  - Nota de la Asignatura (float)
- Sabiendo que tenemos varios alumnos (los que desee el usuario) con 3 asignaturas cada uno (Lengua, Mates y Física), realiza un programa que le dé al usuario las siguientes opciones:
  1. Introducir un nuevo alumno junto a sus notas. (uno solo, si se quieren introducir mas volveríamos a entrar aquí)
  2. Mostrar los alumnos introducidos hasta el momento (junto a sus notas).
  3. Que nos diga que alumno es el mejor de la clase. (nota media más alta) .
  4. Que nos diga cual es la asignatura más difícil. (mayor número de suspensos).
  5. Salir del programa.

Controlaremos que el usuario elija una opción del menú entre 1 y 5. Hasta que el usuario no pulse 5 no saldremos del programa.



"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

- **Ejercicio 15.- (OPTATIVO)** Realiza un programa en JAVA que esté formado por dos clases:
- Clase Empresa:
  - nombreEmpresa de tipo cadena.
  - Lista de empleados de tipo Empleados.
- Clase Empleados:
  - nombreEmpleado de tipo cadena.
  - sueldo de tipo entero.
- Además, crearás un menú que le permita al usuario las siguientes opciones:
  1. Añadir empresas junto a sus empleados a una lista.
  2. Mostrar las empresas de la lista junto a sus empleados
  3. Mostrar los empleados cuyo nombre contenga la letra 'A'
  4. Mostrar todos los empleados ordenados por el sueldo que cobran
  5. Salir del programa.

Ejemplo de ejecución:

Oscar Laguna García  
Ana M. Arribas Arjona

PROGRAMACIÓN

59



"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.

## EJERCICIOS

Pulse 1 para introducir empresas y sus empleados.

Pulse 2 para mostrar empresas y sus empleados.

Pulse 3 para mostrar a todos los empleados cuyo nombre contenga la letra 'A'

Pulse 4 para mostrar a todos los empleados ordenados por su sueldo

Pulse 5 para salir.

1

Nombre de la empresa: **Mc Donalds**

Nombre del empleado 1: **Juan**

Sueldo del empleado 1: **650**

¿Desea añadir más empleados? **Si**

Nombre del empleado 2: **Marta**

Sueldo del empleado 2: 660

¿Desea añadir más empleados? **No**

¿Desea añadir más empresas? **Si**

Pulse 1 para introducir empresas y sus empleados.

Pulse 2 para mostrar empresas y sus empleados.

Pulse 3 para mostrar a todos los empleados cuyo nombre contenga la letra 'A'

Pulse 4 para mostrar a todos los empleados ordenados por su sueldo

Pulse 5 para salir.

2



\*\* Empresa: Mc Donalds \*\*

Empleado 1: Juan

Sueldo:650

Empleado 2: Ana

Sueldo:660

\*\* Empresa: Burger King \*\*

Empleado 1: Pedro

Sueldo:670

Empleado 2: Sonia

Sueldo:680

Pulse 1 para introducir empresas y sus empleados.

Pulse 2 para mostrar empresas y sus empleados.

Pulse 3 para mostrar a todos los empleados cuyo nombre contenga la letra 'A'

Pulse 4 para mostrar a todos los empleados ordenados por su sueldo

Pulse 5 para salir.

3

Juan - Ana - Sonia

Pulse 1 para introducir empresas y sus empleados.

Pulse 2 para mostrar empresas y sus empleados.

Pulse 3 para mostrar a todos los empleados cuyo nombre contenga la letra 'A'

Pulse 4 para mostrar a todos los empleados ordenados por su sueldo

Pulse 5 para salir.

4

Sonia-680

Pedro-670

Ana-660

Juan-650

Pulse 1 para introducir empresas y sus empleados.

Pulse 2 para mostrar empresas y sus empleados.

Pulse 3 para mostrar a todos los empleados cuyo nombre contenga la letra 'A'

Pulse 4 para mostrar a todos los empleados ordenados por su sueldo

Pulse 5 para salir.

5

PROGRAMACIÓN

60

Oscar Laguna García  
Ana M. Arribas Arjona



"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

- **Ejercicio 16.- (OPTATIVO)** Realizar un programa en JAVA en el realices la gestión de una pequeña tienda de deportes. Para ello manejarás objetos que tendrán 3 atributos:
  - Nombre del producto: de tipo String.
  - Precio: de tipo float.
  - Stock: de tipo int.
- Se le mostrarán al usuario un menú con 3 opciones:
  1. MENU DE ADMINISTRACIÓN
  2. MENÚ DE COMPRA
  3. SALIR

Oscar Laguna García  
Ana M. Arribas Arjona

61

PROGRAMACIÓN



"Una manera de hacer Europa". Cofinanciación a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



## EJERCICIOS

- **MENU DE ADMINISTRACIÓN:** Se visualizará el siguiente submenú:
  1. Introducir productos en la lista: Pediremos los datos de un producto al usuario y lo introducimos en la lista.
  2. Visualizar la lista de productos.
  3. Eliminar productos de la lista: Pediremos el nombre del producto y lo eliminaremos.
  4. Volver al menú principal.

Oscar Laguna García  
Ana M. Arribas Arjona

62

PROGRAMACIÓN



UNION EUROPEA  
Fondo Social Europeo  
"Una manera de hacer Europa"

"Una manera de hacer Europa". Colaboración a cargo del Programa Operativo del FSE 2014-2020 para Extremadura gastos de Ciclos Formativos de Grados Medio y Superior.



# EJERCICIOS

## • MENU DE COMPRA:

### 1. Comprar productos:

- Mostramos una lista con los productos a comprar.
- El usuario elegirá que producto comprar y luego le preguntaremos cuantas unidades desea de él. Luego se le preguntará si desea comprar otro producto o salir.
- Por último, se le mostrará el importe total de la compra.
- Date cuenta de que necesitarás 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.

### 2. Volver al menú principal.