

# Manejo de Ficheros

## Alberto Colmenar Casas

2. Borrado recursivo: borrado que consiste en borrar el contenido del directorio primero y cuando esté vacío borrar el directorio.

3. Copia de ficheros: copia un fichero a la ubicación que tú le digas.

4. Ficheros de texto: escribe el fichero provincias con la información de un array de strings y la búsqueda recorre el fichero hasta que da con la provincia literal que pregunta el usuario.

5. Acceso directo a un fichero: sabiendo la estructura del fichero binario muestra la información de un empleado dado su número de empleado.

6. Fichero de objetos: escribe el fichero de Personas dada la información en el método y nos muestra la Persona más mayor.

7. Volcados de fichero de objetos: dado un fichero de objetos Persona pasamos la información a una lista de Persona y también escribimos el contenido de una lista Persona a un fichero.

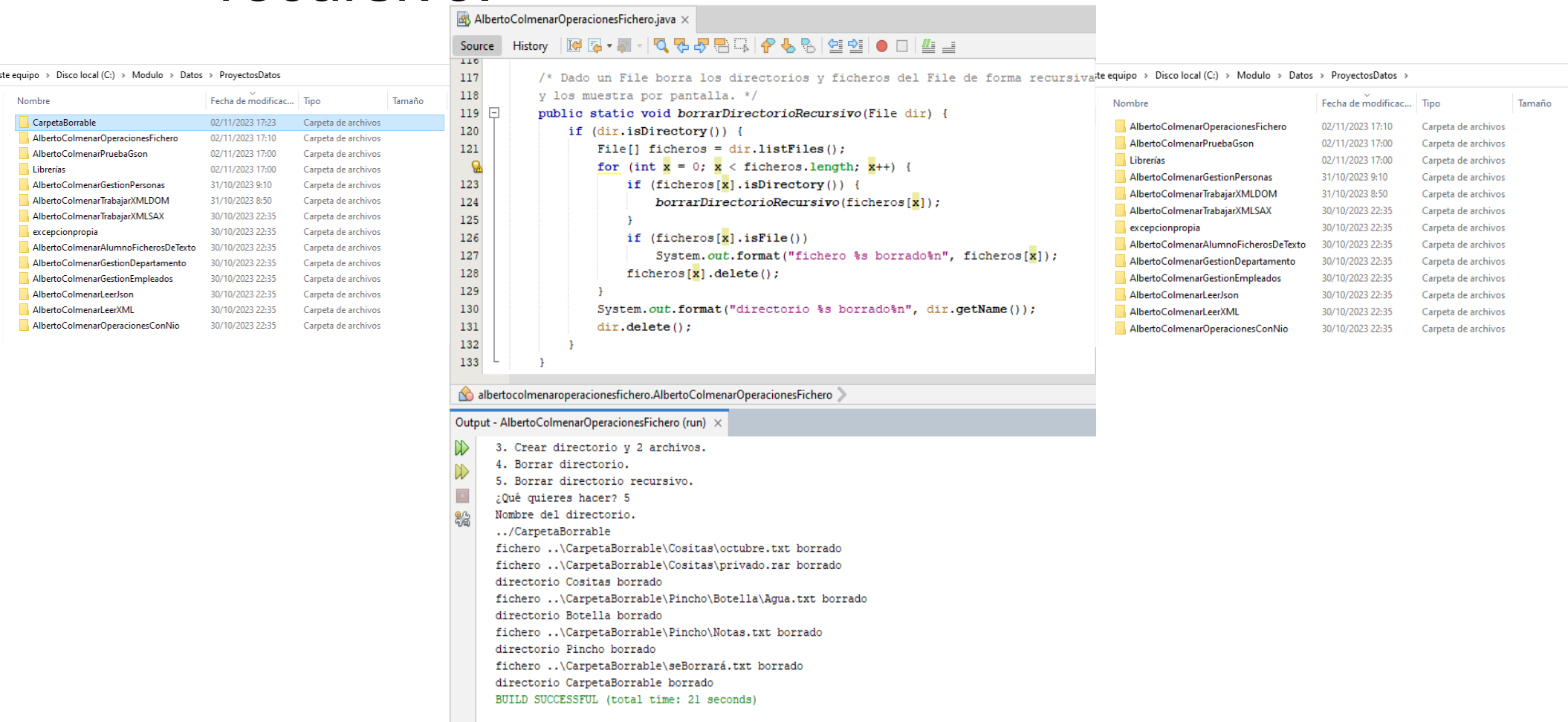
8. Fichero XML: dado Escritores.xml creamos un nuevo nodo de información en cada escritor.

9. Creación de un fichero XML con XStream: con una lista en este caso de Persona creamos un fichero XML con la librería XStream.

10. Fichero JSON: con la librería GSON de Google pasamos a JSON una lista de Alumno y viceversa.

# Practica 1.1 - Borrado recursivo.

Clase File para el directorio y el método listFiles() para hacer una lista de Files (ficheros y directorios) y recorrerlo para ir borrándolos.



The screenshot displays an IDE with a Java file named `AlbertoColmenarOperacionesFichero.java`. The code implements a recursive method `borrarDirectorioRecursivo` to delete files and directories. The output window shows the execution results, including the paths of files and directories deleted.

```
117  /* Dado un File borra los directorios y ficheros del File de forma recursiva
118  y los muestra por pantalla. */
119  public static void borrarDirectorioRecursivo(File dir) {
120      if (dir.isDirectory()) {
121          File[] ficheros = dir.listFiles();
122          for (int x = 0; x < ficheros.length; x++) {
123              if (ficheros[x].isDirectory()) {
124                  borrarDirectorioRecursivo(ficheros[x]);
125              }
126              if (ficheros[x].isFile())
127                  System.out.format("fichero %s borrado\n", ficheros[x]);
128              ficheros[x].delete();
129          }
130          System.out.format("directorio %s borrado\n", dir.getName());
131          dir.delete();
132      }
133  }
```

Output - AlbertoColmenarOperacionesFichero (run) x

```
3. Crear directorio y 2 archivos.
4. Borrar directorio.
5. Borrar directorio recursivo.
¿Qué quieres hacer? 5
Nombre del directorio.
../CarpetaBorrable
fichero ../CarpetaBorrable\Cositas\octubre.txt borrado
fichero ../CarpetaBorrable\Cositas\privado.rar borrado
directorio Cositas borrado
fichero ../CarpetaBorrable\Pincho\Botella\Agua.txt borrado
directorio Botella borrado
fichero ../CarpetaBorrable\Pincho\Notas.txt borrado
directorio Pincho borrado
fichero ../CarpetaBorrable\seBorrará.txt borrado
directorio CarpetaBorrable borrado
BUILD SUCCESSFUL (total time: 21 seconds)
```

# Practica 1.1 - Copia de ficheros.

Clase Paths: para crear las rutas y método get() dado la ruta como string.  
Clase Files: método exists() para comprobar que existe el fichero y copy() para copiarlo en la otra ruta.

The screenshot displays an IDE environment with three main components:

- File Explorer (Left):** Shows a project structure under 'ProyectosDatos'. The file 'AlbertoColmenarOperacionesConNio' is selected.
- Code Editor (Center):** Displays the source code for 'AlbertoColmenarOperacionesConNio.java'. The code defines a method `copiarFichero` that takes two `Path` arguments and uses `Files.exists` and `Files.copy` to copy a file. It includes error handling with `IOException`.
- Output Console (Bottom):** Shows the execution output for 'albertocolmenaroperacionesconnio.AlbertoColmenarOperacionesConNio'. The output indicates a successful build and execution, with a message 'BUILD SUCCESSFUL (total time: 15 seconds)'.

**File Explorer (Left) Details:**

Nombre	Fecha de modificac...	Tipo	Tamaño
notas	02/11/2023 17:37	Documento de te...	1 KB
AlbertoColmenarOperacionesFichero	02/11/2023 17:10	Carpeta de archivos	
AlbertoColmenarPruebaGson	02/11/2023 17:00	Carpeta de archivos	
Librerias	02/11/2023 17:00	Carpeta de archivos	
AlbertoColmenarGestionPersonas	31/10/2023 9:10	Carpeta de archivos	
AlbertoColmenarTrabajarXMLDOM	31/10/2023 8:50	Carpeta de archivos	
AlbertoColmenarTrabajarXMLSAX	30/10/2023 22:35	Carpeta de archivos	
excepcionpropia	30/10/2023 22:35	Carpeta de archivos	
AlbertoColmenarAlumnoFicherosDeTexto	30/10/2023 22:35	Carpeta de archivos	
AlbertoColmenarGestionDepartamento	30/10/2023 22:35	Carpeta de archivos	
AlbertoColmenarGestionEmpleados	30/10/2023 22:35	Carpeta de archivos	
AlbertoColmenarLeerJson	30/10/2023 22:35	Carpeta de archivos	
AlbertoColmenarLeerXML	30/10/2023 22:35	Carpeta de archivos	
AlbertoColmenarOperacionesConNio	30/10/2023 22:35	Carpeta de archivos	

**Code Editor (Center) Details:**

```
58 }
59
60 /* Dado 2 variables Path, una con el Path del fichero que quieres copiar
61 y otra con el Path donde lo quieres copiar.*/
62 public static void copiarFichero(Path ficheroACopiar, Path ficheroCopia) {
63     try {
64         if (Files.exists(ficheroACopiar)) {
65             Files.copy(ficheroACopiar, ficheroCopia);
66         } else {
67             System.out.format("El fichero %s no existe", ficheroACopiar.getFileName());
68         }
69     } catch (IOException ex) {
70         System.out.println("Error al copiar los ficheros" + ex);
71     }
72 }
```

**Output Console (Bottom) Details:**

```
2. Mover fichero.
3. Mostrar fichero.
4. Mostrar información del fichero.
¿Qué quieres hacer? 1
Nombre del fichero. ../notas.txt
Ruta destino. ../notasCopiadas.txt
BUILD SUCCESSFUL (total time: 15 seconds)
```

**File Explorer (Right):** Shows the project structure under 'ProyectosDatos'. The file 'AlbertoColmenarOperacionesConNio' is selected.

Nombre	Fecha de modificac...	Tipo	Tamaño
notas	02/11/2023 17:37	Documento de te...	1 KB
notasCopiadas	02/11/2023 17:37	Documento de te...	1 KB
AlbertoColmenarOperacionesFichero	02/11/2023 17:10	Carpeta de archivos	
AlbertoColmenarPruebaGson	02/11/2023 17:00	Carpeta de archivos	
Librerias	02/11/2023 17:00	Carpeta de archivos	
AlbertoColmenarGestionPersonas	31/10/2023 9:10	Carpeta de archivos	
AlbertoColmenarTrabajarXMLDOM	31/10/2023 8:50	Carpeta de archivos	
AlbertoColmenarTrabajarXMLSAX	30/10/2023 22:35	Carpeta de archivos	
excepcionpropia	30/10/2023 22:35	Carpeta de archivos	
AlbertoColmenarAlumnoFicherosDeTexto	30/10/2023 22:35	Carpeta de archivos	
AlbertoColmenarGestionDepartamento	30/10/2023 22:35	Carpeta de archivos	
AlbertoColmenarGestionEmpleados	30/10/2023 22:35	Carpeta de archivos	
AlbertoColmenarLeerJson	30/10/2023 22:35	Carpeta de archivos	
AlbertoColmenarLeerXML	30/10/2023 22:35	Carpeta de archivos	
AlbertoColmenarOperacionesConNio	30/10/2023 22:35	Carpeta de archivos	

# Practica 1.1 - Ficheros de texto. Provincias.txt.

Clase File: para crear el fichero provincias.txt  
Clase PrintWriter: para escribir en el fichero. Método print() para ir escribiendo las provincias cuando se va recorriendo la lista.

```
AlbertoColmenarAlumnoFicherosDeTexto.java x
Source History
67 // Crea el fichero provincias.txt con la lista de provincias del propio método.
68 public static void escribirProvincias() {
69     PrintWriter texto = null;
70     try {
71         String prov[]={"Albacete", "Avila", "Alicante", "Badajoz", "Barcelona",
72             "Bilbao", "Caceres", "Cádiz", "Corboba", "Huelva", "Sevilla", "Soria",
73             "Toledo", "Valencia", "Zamora", "Zaragoza"};
74         File ficheroProvincias = new File("provincias.txt");
75         texto = new PrintWriter(ficheroProvincias);
76         for(String provincia: prov) {
77             texto.print(provincia + " ");
78         }
79     } catch(IOException ex) {
80         System.out.println("Error con el fichero" + ex);
81     } finally {
82         texto.close();
83     }
84 }
85
86
```

```
albertocolmenaralumnoficherosdetexto.AlbertoColmenarAlumnoFicherosDeTexto
Output - AlbertoColmenarAlumnoFicherosDeTexto (run) x
1. Leer fichero.
2. Crear fichero provincias.
3. Buscar provincias en el fichero provincias.txt.
4. Crear alfabeto.
¿Qué quieres hacer? 2
Fichero provincias.txt creado
BUILD SUCCESSFUL (total time: 2 seconds)
```

Clase Scanner: para ir recorriendo el fichero con el método hasNext() (para el while) y next()

```
AlbertoColmenarAlumnoFicherosDeTexto.java x
Source History
86 // Búsqueda en el fichero de la provincia pasada por parámetro
87 public static void encontrarProvincia(File fichero, String provincia) {
88     try {
89         Scanner sc = new Scanner(fichero);
90         boolean encontrado = false;
91         while (sc.hasNext() && !encontrado) {
92             encontrado = sc.next().equals(provincia);
93         }
94         System.out.format("¿Se ha encontrado la provincia %s? %b\n", provincia, encontrado);
95     } catch (IOException ex) {
96         System.out.println("Error con el fichero" + ex);
97     }
98 }
99
```

```
albertocolmenaralumnoficherosdetexto.AlbertoColmenarAlumnoFicherosDeTexto
Output - AlbertoColmenarAlumnoFicherosDeTexto (run) x
2. Crear fichero provincias.
3. Buscar provincias en el fichero provincias.txt.
4. Crear alfabeto.
¿Qué quieres hacer? 3
Nombre de la provincia. Caceres
¿Se ha encontrado la provincia Caceres? true
BUILD SUCCESSFUL (total time: 8 seconds)
```

te equipo > Disco local (C:) > Modulo > Datos > ProyectosDatos > AlbertoColmenarAlumnoFicherosDeTexto				
Nombre	Fecha de modificación	Tipo	Tamaño	
build	30/10/2023 22:35	Carpeta de archivos		
nbproject	30/10/2023 22:35	Carpeta de archivos		
src	30/10/2023 22:35	Carpeta de archivos		
alfabeto	30/10/2023 22:35	Documento de te...	1 KB	
build	02/11/2023 17:49	Documento XML	4 KB	
manifest.mf	30/10/2023 22:35	Archivo MF	1 KB	

te equipo > Disco local (C:) > Modulo > Datos > ProyectosDatos > AlbertoColmenarAlumnoFicherosDeTexto >				
Nombre	Fecha de modificación	Tipo	Tamaño	
build	30/10/2023 22:35	Carpeta de archivos		
nbproject	30/10/2023 22:35	Carpeta de archivos		
src	30/10/2023 22:35	Carpeta de archivos		
alfabeto	30/10/2023 22:35	Documento de te...	1 KB	
build	02/11/2023 17:49	Documento XML	4 KB	
manifest.mf	30/10/2023 22:35	Archivo MF	1 KB	
provincias	02/11/2023 17:54	Documento de te...	1 KB	

# Práctica 1.2 - Acceso directo a un fichero.

```
AlbertoColmenarGestionEmpleados.java x
Source History
96 // Dado el fichero binario de empleados y un número de empleados muestra la información de ese empleado
97 public static void visualizarEmpleado(File fichero, int numEmpleado) {
98     RandomAccessFile raf = null;
99     try {
100         raf = new RandomAccessFile(fichero, "r");
101         int posicion = (numEmpleado - 1) * 16;
102         if (posicion < raf.length()) {
103             raf.seek(posicion);
104             int numEmp = raf.readInt();
105             int numDepart = raf.readInt();
106             double salario = raf.readDouble();
107             System.out.format("El empleado %d está en el departamento %d y cobra %.0f€\n", numEmp, numDepart, salario);
108         } else {
109             System.out.println("El empleado no existe.");
110         }
111     } catch (EOFException ex) {
112         System.out.println("Se ha completado la lectura del fichero " + ex.getMessage());
113     } catch (FileNotFoundException ex) {
114         System.out.println("No se encontró el fichero " + ex.getMessage());
115     } catch (IOException ex) {
116         System.out.println("Error al leer " + ex.getMessage());
117     }
}
```

albertocolmenargestionempleados.AlbertoColmenarGestionEmpleados >

Output - AlbertoColmenarGestionEmpleados (run) x

```
El fichero ya existe. ¿Desea sobrescribirlo? (s/n)
n
1. Consulta de los datos de un empleado.
2. Alta de un empleado.
1
Número de empleado:
2
El empleado 2 está en el departamento 65 y cobra 321€
BUILD SUCCESSFUL (total time: 8 seconds)
```

Clase RandomAccessFile: para abrir el fichero en lectura y, sabiendo la estructura del fichero, usar el método seek() para colocarnos en posición para leer la información del empleado con readInt(), readDouble()

# Práctica 1.2 - Fichero de Objetos.

```
Persona.java x AlbertoColmenarGestionPersonas.java x
Source History
1 package albertocolmenargestionpersonas;
2
3 import java.io.Serializable;
4
5 /**
6  *
7  * @author DAM2A-03
8  */
9 public class Persona implements Serializable {
10
11     private String dni;
12     private String nombre;
13     private int edad;
14
15     public Persona(String dni, String nombre, int edad) {
16         this.dni = dni;
17         this.nombre = nombre;
18         this.edad = edad;
19     }
20
21     public String getDni() {
22         return dni;
23     }
24
25     public String getNombre() {
26         return nombre;
27     }
28
29     public int getEdad() {
30         return edad;
31     }
32
33     public void setDni(String dni) {
34         this.dni = dni;
35     }
36
37     public void setNombre(String nombre) {
38         this.nombre = nombre;
39     }
40
41     public void setEdad(int edad) {
42         this.edad = edad;
43     }
44
45     @Override
46     public String toString() {
47         return "Persona{" + "dni=" + dni + ", nombre=" + nombre + ", edad=" + edad + '}';
48     }
49 }
```

Clase ObjectOutputStream: método writeObject() para escribir en el fichero el objeto Persona.

```
Persona.java x AlbertoColmenarGestionPersonas.java x
Source History
108 // Crea el fichero datosPersona.obj con los datos del método
109 private static void crearFicheroPersonas(File fichero) {
110     String dnis[] = {"83487622", "13481112", "22356622", "33455614", "77733182",
111         "83765629", "22487611", "56787682", "80087600", "53077609"};
112     String nombres[] = {"Ana", "Javier", "Luisa", "Tomás", "Julio", "Pedro",
113         "Rocio", "German", "Maria", "Serafin"};
114     int edades[] = {17, 22, 19, 15, 20, 26, 25, 19, 17, 20};
115     ObjectOutputStream oos = null;
116     try {
117         oos = new ObjectOutputStream(new FileOutputStream(fichero));
118         for (int i = 0; i < dnis.length; i++) {
119             oos.writeObject(new Persona(dnis[i], nombres[i], edades[i]));
120         }
121         oos.close();
122     } catch (FileNotFoundException ex) {
123         System.out.println("No se encuentra el fichero" + ex.getMessage());
124     } catch (IOException ex) {
125         System.out.println("Error en el fichero" + ex.getMessage());
126     }
127 }
```

Uso el método volcadoALista() para leer el fichero entero y poder manejarlo con un ArrayList por comodidad. Al recorrerlo me voy quedando con la Persona con más edad

```
Persona.java x AlbertoColmenarGestionPersonas.java x
Source History
155 // Dado el fichero de personas, muestra la persona con más edad
156 private static void mostrarPersonaMayor(File fichero) {
157     ArrayList<Persona> lista = new ArrayList<>();
158     volcadoALista(fichero, lista);
159     Persona personaMayor = lista.get(0);
160     for (Persona persona : lista) {
161         if (persona.getEdad() > personaMayor.getEdad()) {
162             personaMayor = persona;
163         }
164     }
165     System.out.println(personaMayor.toString());
166 }
167
albertocolmenargestionpersonas.AlbertoColmenarGestionPersonas
Output - AlbertoColmenarGestionPersonas (run) x
run:
El fichero ya existe, ¿desea sobrescribirlo? (s/n) n
1. Consultas.
2. Mantenimiento.
3. Salida.
1
Consultas.
1. Mostrar los datos de una persona dado su DNI.
2. Mostrar los datos de la Persona con mayor edad en el fichero.
3. Listado de Personas comprendidas en un rango de edades.
4. Media de edad en el fichero.
5. Salir del submenú.
2
Persona(dni=83765629, nombre=Pedro, edad=26)
1. Consultas.
2. Mantenimiento.
3. Salida.
3
BUILD SUCCESSFUL (total time: 22 seconds)
```

# Práctica 1.2 - Volcados de Fichero de objetos.

```
Persona.java x AlbertoColmenarGestionPersonas.java x
Source History
172 // Lee el fichero de personas y escribe cada persona en el ArrayList
173 public static void volcadoALista(File f, ArrayList<Persona> lista) {
174     try {
175         ObjectInputStream ois;
176         Persona p;
177         ois = new ObjectInputStream(new FileInputStream(f));
178         while (true) {
179             p = (Persona) ois.readObject();
180             lista.add(p);
181         }
182     } catch (EOFException eof) {
183     } catch (FileNotFoundException ex) {
184         System.err.println(ex.getMessage());
185     } catch (ClassNotFoundException ex) {
186         System.err.println(ex.getMessage());
187     } catch (IOException ex) {
188         System.err.println(ex.getMessage());
189     }
190 }
```

Clase ObjectInputStream:  
método readObject() para ir  
leyendo el fichero y después  
añadir la Persona a la lista

```
Persona.java x AlbertoColmenarGestionPersonas.java x
Source History
191 // Dado un ArrayList de Persona, lo recorre y lo escribe en el fichero
192 public static void volcadoAFichero(File f, ArrayList<Persona> lista) {
193     ObjectOutputStream oos = null;
194     try {
195         oos = new ObjectOutputStream(new FileOutputStream(f));
196         for (Persona persona : lista) {
197             oos.writeObject(persona);
198         }
199         oos.close();
200     } catch (FileNotFoundException ex) {
201         System.err.println(ex.getMessage());
202     } catch (IOException ex) {
203         System.err.println(ex.getMessage());
204     }
205 }
206 }
```

Clase ObjectOutputStream:  
método writeObject() para ir  
escribiendo la Persona al recorrer  
la lista

re equipo > Disco local (C:) > Modulo > Datos > ProyectosDatos > AlbertoColmenarGestionPersonas

Nombre	Fecha de modificación	Tipo	Tamaño
build	30/10/2023 22:35	Carpeta de archivos	
nbproject	30/10/2023 22:35	Carpeta de archivos	
src	30/10/2023 22:35	Carpeta de archivos	
test	31/10/2023 9:10	Carpeta de archivos	
build	31/10/2023 9:05	Documento XML	4 KB
datosPersonas	02/11/2023 19:05	3D Object	1 KB
manifest.mf	30/10/2023 22:35	Archivo MF	1 KB
Personas	02/11/2023 18:43	Firefox HTML Doc...	2 KB
Personas	02/11/2023 18:43	Documento XML	1 KB
Personas	30/10/2023 22:35	Hoja de estilos XSL	1 KB

```
Persona.java x AlbertoColmenarGestionPersonas.java x
Source History
48
49 ArrayList<Persona> lista = new ArrayList<>();
50 volcadoALista(fichero, lista);
51 listadoCompleto(lista);
52 File fichero2 = new File("datosLista.obj");
53 volcadoAFichero(fichero2, lista);
```

Ejecución de los 2  
métodos y mostrado por  
consola

```
Output - AlbertoColmenarGestionPersonas (run) x
RUN:
El fichero ya existe, ¿desea sobrescribirlo? (s/n) s
Persona(dni=83487622, nombre=Ana, edad=17)
Persona(dni=13481112, nombre=Javier, edad=22)
Persona(dni=22356622, nombre=Luisa, edad=19)
Persona(dni=33455614, nombre=Tomás, edad=15)
Persona(dni=77733182, nombre=Julio, edad=20)
Persona(dni=83765629, nombre=Pedro, edad=26)
Persona(dni=22487611, nombre=Rocio, edad=25)
Persona(dni=56787682, nombre=German, edad=19)
Persona(dni=80087600, nombre=Maria, edad=17)
Persona(dni=53077609, nombre=Serafin, edad=20)
BUILD SUCCESSFUL (total time: 2 seconds)
```

re equipo > Disco local (C:) > Modulo > Datos > ProyectosDatos > AlbertoColmenarGestionPersonas

Nombre	Fecha de modificación	Tipo	Tamaño
build	30/10/2023 22:35	Carpeta de archivos	
nbproject	30/10/2023 22:35	Carpeta de archivos	
src	30/10/2023 22:35	Carpeta de archivos	
test	31/10/2023 9:10	Carpeta de archivos	
build	31/10/2023 9:05	Documento XML	4 KB
datosLista	02/11/2023 19:09	3D Object	1 KB
datosPersonas	02/11/2023 19:09	3D Object	1 KB
manifest.mf	30/10/2023 22:35	Archivo MF	1 KB
Personas	02/11/2023 18:43	Firefox HTML Doc...	2 KB
Personas	02/11/2023 18:43	Documento XML	1 KB
Personas	30/10/2023 22:35	Hoja de estilos XSL	1 KB



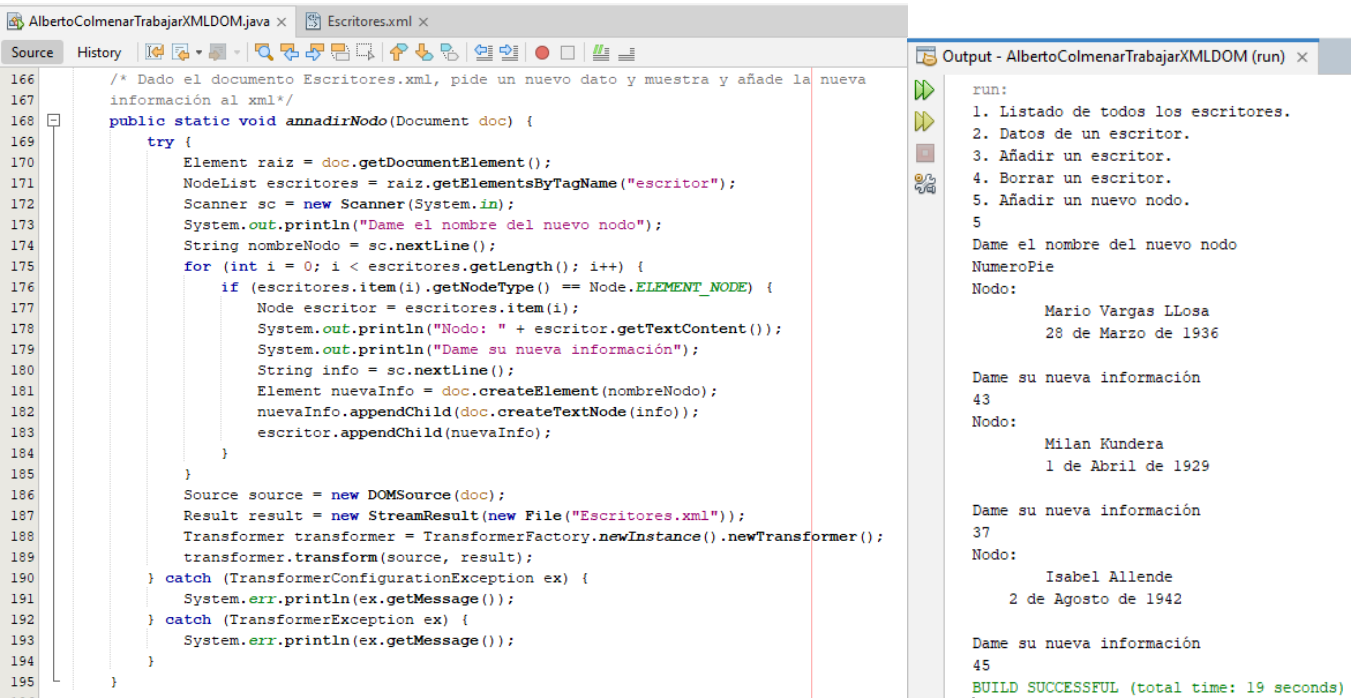
# Práctica 1.3 - Fichero XML. Añadir un nuevo item a todos los nodo del fichero.

Clase Scanner: para los datos introducidos por el usuario.

Clase NodeList: getElementsByTagName() para coger a todos los escritores e ir recorriéndolos.

Para luego crear el nuevo nodo Element y su información (createElement(), createTextNode()) y añadirlo en el escritor (appendChild()).

Al final Source y Result para escribir el fichero Escritores.xml de nuevo.

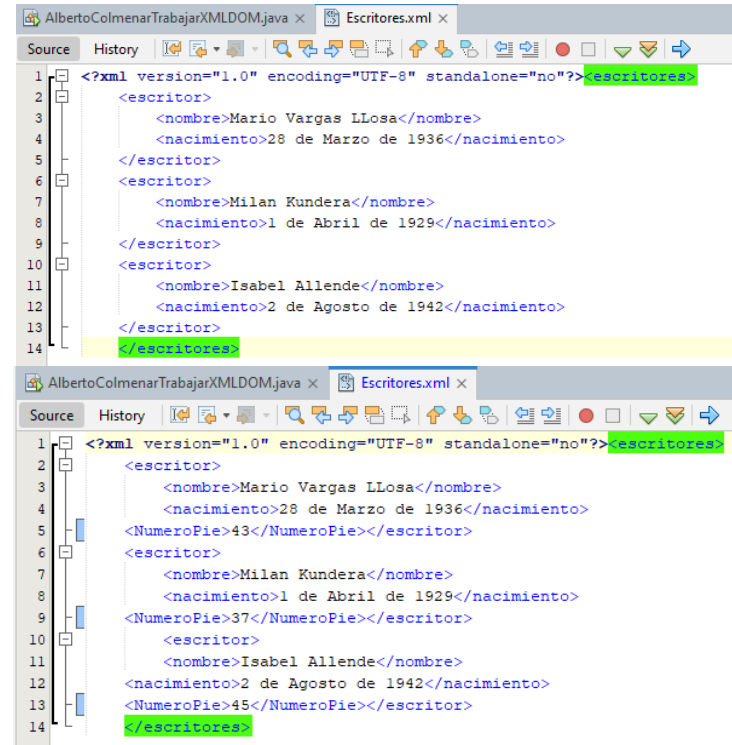


The screenshot shows an IDE with two tabs: 'AlbertoColmenarTrabajarXMLDOM.java' and 'Escritores.xml'. The Java code in the first tab is as follows:

```
166 /* Dado el documento Escritores.xml, pide un nuevo dato y muestra y añade la nueva
167 información al xml*/
168 public static void annadirNodo(Document doc) {
169     try {
170         Element raiz = doc.getDocumentElement();
171         NodeList escritores = raiz.getElementsByTagName("escritor");
172         Scanner sc = new Scanner(System.in);
173         System.out.println("Dame el nombre del nuevo nodo");
174         String nombreNodo = sc.nextLine();
175         for (int i = 0; i < escritores.getLength(); i++) {
176             if (escritores.item(i).getNodeType() == Node.ELEMENT_NODE) {
177                 Node escritor = escritores.item(i);
178                 System.out.println("Nodo: " + escritor.getTextContent());
179                 System.out.println("Dame su nueva información");
180                 String info = sc.nextLine();
181                 Element nuevaInfo = doc.createElement(nombreNodo);
182                 nuevaInfo.appendChild(doc.createTextNode(info));
183                 escritor.appendChild(nuevaInfo);
184             }
185         }
186         Source source = new DOMSource(doc);
187         Result result = new StreamResult(new File("Escritores.xml"));
188         Transformer transformer = TransformerFactory.newInstance().newTransformer();
189         transformer.transform(source, result);
190     } catch (TransformerConfigurationException ex) {
191         System.err.println(ex.getMessage());
192     } catch (TransformerException ex) {
193         System.err.println(ex.getMessage());
194     }
195 }
```

The second tab shows the output of the program:

```
Output - AlbertoColmenarTrabajarXMLDOM (run) x
run:
1. Listado de todos los escritores.
2. Datos de un escritor.
3. Añadir un escritor.
4. Borrar un escritor.
5. Añadir un nuevo nodo.
Dame el nombre del nuevo nodo
NumeroPie
Nodo:
    Mario Vargas Llosa
    28 de Marzo de 1936
Dame su nueva información
43
Nodo:
    Milan Kundera
    1 de Abril de 1929
Dame su nueva información
37
Nodo:
    Isabel Allende
    2 de Agosto de 1942
Dame su nueva información
45
BUILD SUCCESSFUL (total time: 19 seconds)
```



The screenshot shows the 'Escritores.xml' file in an IDE. The XML content is as follows:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>escritores
2
3 <escritor>
4     <nombre>Mario Vargas Llosa</nombre>
5     <nacimiento>28 de Marzo de 1936</nacimiento>
6 </escritor>
7 <escritor>
8     <nombre>Milan Kundera</nombre>
9     <nacimiento>1 de Abril de 1929</nacimiento>
10 </escritor>
11 <escritor>
12     <nombre>Isabel Allende</nombre>
13     <nacimiento>2 de Agosto de 1942</nacimiento>
14 </escritor>
15 </escritores>
```

The second screenshot shows the same XML file after modification, with the following content:

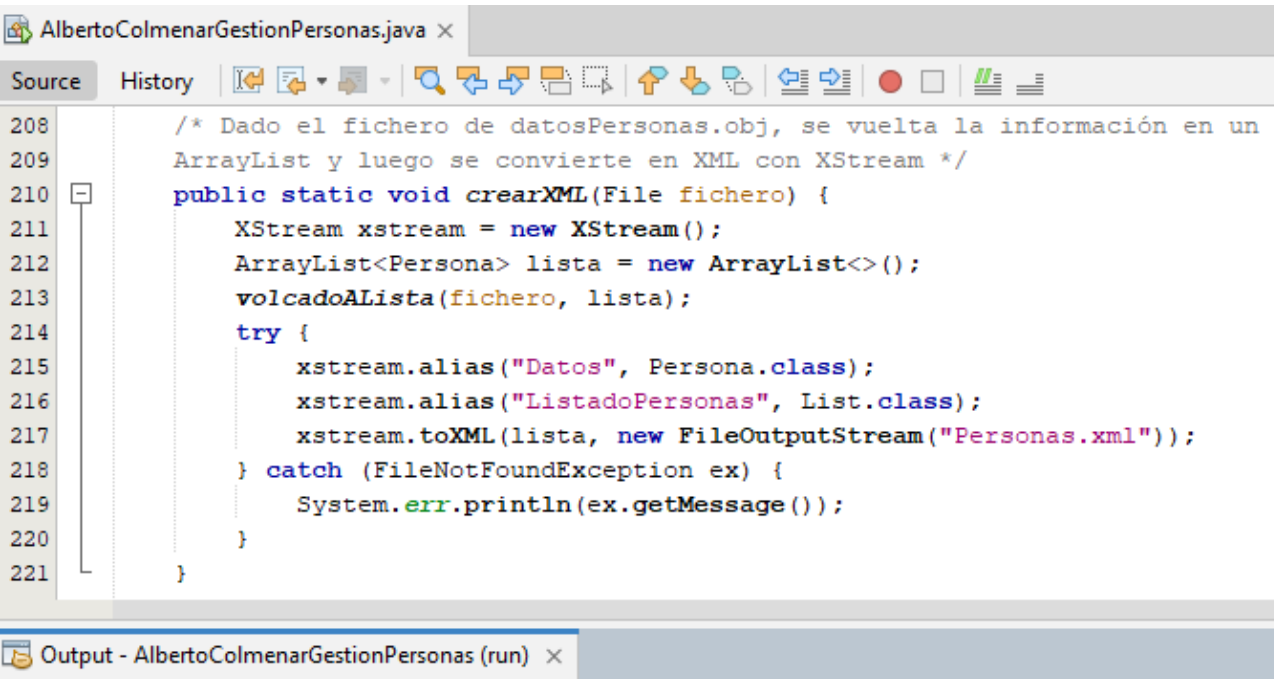
```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>escritores
2
3 <escritor>
4     <nombre>Mario Vargas Llosa</nombre>
5     <nacimiento>28 de Marzo de 1936</nacimiento>
6     <NumeroPie>43</NumeroPie></escritor>
7 <escritor>
8     <nombre>Milan Kundera</nombre>
9     <nacimiento>1 de Abril de 1929</nacimiento>
10     <NumeroPie>37</NumeroPie></escritor>
11 <escritor>
12     <nombre>Isabel Allende</nombre>
13     <nacimiento>2 de Agosto de 1942</nacimiento>
14     <NumeroPie>45</NumeroPie></escritor>
15 </escritores>
```



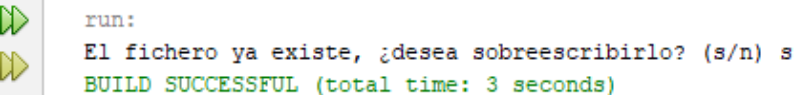
# Práctica 1.3 - Creación de un fichero XML con XStream.

Clase Xstream: método alias() para que no nos escriba el paquete de la clase Persona y la clase List y toXML() para pasar la lista a XML.

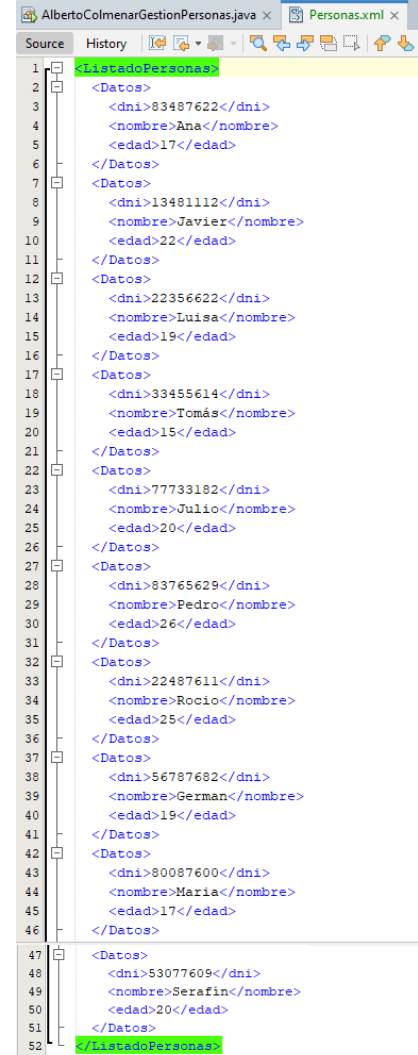
Usamos de nuevo volcadoALista() para pasar la información del fichero a un ArrayList de Persona



```
Source History
208 /* Dado el fichero de datosPersonas.obj, se vuelta la información en un
209 ArrayList y luego se convierte en XML con XStream */
210 public static void crearXML(File fichero) {
211     XStream xstream = new XStream();
212     ArrayList<Persona> lista = new ArrayList<>();
213     volcadoALista(fichero, lista);
214     try {
215         xstream.alias("Datos", Persona.class);
216         xstream.alias("ListadoPersonas", List.class);
217         xstream.toXML(lista, new FileOutputStream("Personas.xml"));
218     } catch (FileNotFoundException ex) {
219         System.err.println(ex.getMessage());
220     }
221 }
```



```
Output - AlbertoColmenarGestionPersonas (run) x
run:
El fichero ya existe, ¿desea sobreescribirlo? (s/n) s
BUILD SUCCESSFUL (total time: 3 seconds)
```



```
Source History
1 <ListadoPersonas>
2 <Datos>
3 <dni>83487622</dni>
4 <nombre>Ana</nombre>
5 <edad>17</edad>
6 </Datos>
7 <Datos>
8 <dni>13481112</dni>
9 <nombre>Javier</nombre>
10 <edad>22</edad>
11 </Datos>
12 <Datos>
13 <dni>22356622</dni>
14 <nombre>Luisa</nombre>
15 <edad>19</edad>
16 </Datos>
17 <Datos>
18 <dni>33455614</dni>
19 <nombre>Tomás</nombre>
20 <edad>15</edad>
21 </Datos>
22 <Datos>
23 <dni>77733182</dni>
24 <nombre>Julio</nombre>
25 <edad>20</edad>
26 </Datos>
27 <Datos>
28 <dni>83765629</dni>
29 <nombre>Pedro</nombre>
30 <edad>26</edad>
31 </Datos>
32 <Datos>
33 <dni>22487611</dni>
34 <nombre>Rocio</nombre>
35 <edad>25</edad>
36 </Datos>
37 <Datos>
38 <dni>56787682</dni>
39 <nombre>German</nombre>
40 <edad>19</edad>
41 </Datos>
42 <Datos>
43 <dni>80087600</dni>
44 <nombre>Maria</nombre>
45 <edad>17</edad>
46 </Datos>
47 <Datos>
48 <dni>53077609</dni>
49 <nombre>Serafin</nombre>
50 <edad>20</edad>
51 </Datos>
52 </ListadoPersonas>
```

# Práctica 1.4 - Fichero JSON. Lectura de un fichero JSON con GSON.

```
AlbertoColmenarPruebaGson.java x
Source History

50 // Usando Gson y la lista de Alumnos del método retorna el JSON de dicha lista
51 public static String mostrarAlumnosJson() {
52     Gson gson = new Gson();
53     ArrayList<Alumno> listaAlumnos = new ArrayList();
54     listaAlumnos.add(new Alumno("Miriam", 23));
55     listaAlumnos.add(new Alumno("Luis", 12));
56     listaAlumnos.add(new Alumno("Lucia", 75));
57     listaAlumnos.add(new Alumno("Julian", 51));
58     return gson.toJson(listaAlumnos);
59 }
60
61 // Dado un JSON de Alumno lo muestra por la entrada estándar
62 public static void mostrarFicheroJson(File ficheroJson) {
63     try {
64         Gson gson = new Gson();
65         Type tipoListaCliente = new TypeToken<List<Alumno>>().getType();
66         List<Alumno> listaAlumno = gson.fromJson(new FileReader(ficheroJson), tipoListaCliente);
67         for (Alumno alumno : listaAlumno) {
68             System.out.println(alumno);
69         }
70     } catch (FileNotFoundException ex) {
71         System.err.println(ex.getMessage());
72     }
73 }
74 }
```

```
Output - AlbertoColmenarPruebaGson (run) x
run:
Alumno: nombre = Miriam, nota = 23
Alumno: nombre = Luis, nota = 12
Alumno: nombre = Lucia, nota = 75
Alumno: nombre = Julian, nota = 51
BUILD SUCCESSFUL (total time: 0 seconds)
```

Clase Gson: método toJson() dado un arrayList nos devuelve un String en formato JSON.  
fromJson() dado el fichero y un Type que sea una lista de Alumno para que nos devuelva en este caso una lista de Alumno.

```
AlbertoColmenarPruebaGson.java x
Source History

36
37
38 String jsonAlumnos = mostrarAlumnosJson();
39 PrintWriter ficheroJsonAlumnos = new PrintWriter(new FileWriter("listaAlumnos.json"));
40 ficheroJsonAlumnos.print(jsonAlumnos);
41 ficheroJsonAlumnos.close();
42 File ficheroJson = new File("listaAlumnos.json");
43 mostrarFicheroJson(ficheroJson);
```