Resumen Programación 3ºEVA

Tema 9: Ficheros

Ficheros Secuenciales

FICHERO: Colección de información que almacenamos en un soporte físico, registrando datos u objetos con sus correspondientes atributos

FLUJOS: Objeto que permite la comunicación entre el programa y el origen/destino de la información

FICHERO SECUENCIAL: Puede almacenar registros de cualquier longitud. Los registros se colocan unos a continuación de los otros, cada objeto se almacena con una secuencia de bytes que determina su tamaño Flujos:

A. *FLUJO DE BYTES: Los datos son escritos/leídos byte a byte*

- *FileOutputStream - MÉTODOS:*
  - *FileOutputStream(fichero); //Crea el fichero*
  - *FileOutputStream(fichero, true); //Añade datos al final*
- *FileInputStream - MÉTODOS:*
  - *FileInputStream(fichero); //Permite leer el fichero*

//Todas versiones se pueden usar con un String como parámetro

B. *FLUJO DE DATOS: Escribe/lee en el fichero datos de tipo primitivo y caracteres UTF-8 (String)*

- *DataOutputStream - MÉTODOS:*
  - *DataOutputStream(new FileOutputStream(fichero));*
    //Abre el flujo de escritura en el fichero
  - *dos.writeBoolean(); //Grava un valor booleano*
  - *dos.writeByte(); //Grava un valor byte dos.writeUTF(); //Grava un valor String ...*
- *DataInputStream - MÉTODOS:*
  - *DataInputStream(new FileInputStream(fichero));*
    //Crea el flujo de lectura en el fichero
  - *dis.readBoolean(); //Lee un valor booleano*

- dis.readByte(); //Lee un valor byte dis.readUTF(); //Lee un valor String ...

C. SERIACIÓN DE OBJETOS: En el fichero se serian/deserian objetos mediante la interfaz Serializable //import java.io.Serializable

- FLUJOS:
  - ObjectOutputStream(new FileOutputStream(fichero)); //Flujo de escritura de objetos
  - ObjectInputStream(new FileInputStream(fichero)); //Flujo de lectura de objetos
- MÉTODOS:
  - oos.writeObject(objeto); //Grava el objeto en el fichero
  - ois.readObject(objeto); //Lee el objeto del fichero
- Para que la seriación no de problemas con las cabeceras de los objetos debemos reescribir el método writeStreamHeader()

```
2  package clases;
3
4  import java.io.IOException;
5  import java.io.ObjectOutputStream;
6  import java.io.OutputStream;
7
8  /**
9   *
10  * @author darkness
11  *
12  */
13
14 public class ObjectOutputStreamSinCabecera extends ObjectOutputStream {
15
16     public ObjectOutputStreamSinCabecera (OutputStream out) throws IOException {
17
18         super(out);
19     }
20
21     @Override
22     protected void writeStreamHeader () throws IOException {
23
24         this.reset();
25     }
26 }
```

Resumen Programación 3ºEVA

D. SERIACIÓN DE OBJETOS QUE REFERENCIAN A OBJETOS: Permite guardar en un fichero una colección de datos, por ejemplo, ArrayList. Los datos se guardan/leen de forma atómica (//por esta razón no debemos reescribir el método writeStreamHeader()) accediendo siempre al ArrayList en memoria, que estará, previamente cargado del fichero

- FLUJOS:
  - ObjectOutputStream(new FileOutputStream(fichero));
    //Flujo de escritura de objetos
  - ObjectInputStream(new FileInputStream(fichero));
    //Flujo de lectura de objetos

- MÉTODOS:
  - oos.writeObject(objeto); //Grava el objeto en el fichero
  - ois.readObject(objeto); //Lee el objeto del fichero Clase File:

A. CONSTRUCTORES:

- File fichero = new File(rutaCompleta); //Fichero en su ruta
- File fichero = new File(ruta, nombre); //Fich en su ruta+nombre

B. MÉTODOS:

- fich.exists(); //Comprueba la existencia del fichero
- fich.length(); //Devuelve el tamaño del fichero
- fich.delete(); //Borra el fichero
- fich.renameTo(); //Renombra el fichero
- fich.toString(); //Devuelve la ruta del fichero cuando se crea

Resumen Programación 3ºEVA

Ficheros Aleatorios

Este tipo de ficheros se caracterizan por tener un índice que permite acceder al registro deseado, para que esto sea posible todos los registros deben ser del mismo tamaño. No permite seriación de objetos

A. CONSTRUCTOR:

- *RandomAccessFile raf = new RandomAccessFile(fichero, modo);*
- *RandomAccessFile raf = new RandomAccessFile(String, modo);*

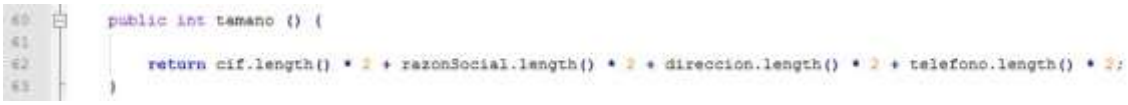
B. MODOS DE APERTURA:

- SOLO LECTURA: *r*
- LECTURA/ESCRITURA: *rw*

C. MÉTODOS:

- ESCRITURA:
  - *raf.writeUTF(); raf.writeByte(); …*
- LECTURA:
  - *raf.readUTF(); raf.readByte(); …*
- OTROS MÉTODOS:
  - *raf.getFilePointer(): Devuelve la posición actual en bytes del puntero de lectura/escritura*
  - *raf.length(): Devuelve la longitud en bytes del fichero*
  - *raf.seek(): Mueve el puntero de lectura/escritura a una posición determinada*

En este tipo de ficheros a la clase de la cual grabaremos objetos debemos añadirle un método que calcule el tamaño total de cada registro, ejemplo:



```
public int tamano () {

    return cif.length() * 2 + razonSocial.length() * 2 + direccion.length() * 2 + telefono.length() * 2;
}
```

Sumando el tamaño en bytes de cada tipo de dato:

Tipo	Tamaño (bytes)	Tipo	Tamaño (bytes)
byte	1	double	8
short	2	char	2
int	4	boolean	1
long	8	String	length * 2
float	4		

Además tendrá un atributo: public static final int tamanoReg = 140; para controlar el tamaño máximo de cada registro

Resumen Programación 3ºEVA

Métodos de trabajo con el fichero:

CREACIÓN DEL FICHERO:

```
public static int crearFichero (File fichero) throws IOException {

    RandomAccessFile raf = null;
    int numRegs = 0;

    try {

        if (fichero.exists()) {

            System.out.println("\n El fichero ya existe \n");
            numRegs = (int) Math.ceil((double) fichero.length() / (double) Empresa.tamanoReg);
            System.out.println("\n Numero de registros calculado satisfactoriamente: " + numRegs + " \n");
        } else {

            System.out.println("\n El fichero no existe, se procede a crearlo");
            raf = new RandomAccessFile(fichero, "rw");
            numRegs = 0;
            System.out.println("\n Numero de registros establecido a: " + numRegs + " \n");
        }
    }
    catch (Exception e) {
    }
    finally {

        if (raf != null) {

            raf.close();
        }
    }
    return numRegs;
}
```

GRABAR EN OBJETOS EN EL FICHERO:

```
public static boolean grabar (File fichero, Empresa emp, int pos, int numRegs) throws IOException {

    RandomAccessFile raf = null;
    boolean exito = false;

    try {

        raf = new RandomAccessFile(fichero, "rw");

        if(pos >= 0 && pos <= numRegs) {

            if(emp.tamano() + 2 + 2 > Empresa.tamanoReg) {

                System.out.println("\n Tamaño excedido, registro no grabado \n");
                exito = false;
            } else{

                raf.seek(pos * Empresa.tamanoReg);
                raf.writeUTF(emp.getCif());
                //resto datos
                exito = true;
            }
        }
    } catch(EOFException e) {
    } finally {

        if (raf != null) {

            raf.close();
        }
    }
    return exito;
}
```

Resumen Programación 3ºEVA

BORRADO DE DATOS EN EL FICHERO:

```
public static boolean actualizarFichero (File fichero, String cif, int numRegs) throws IOException {
    File ficheroTemp = null;
    RandomAccessFile raf = null;
    Empresa emp;
    boolean exito = false;
    int i, j = 0;

    try {
        raf = new RandomAccessFile(fichero, "r");
        ficheroTemp = new File ("temporal.txt");
        for (i = 0; i < numRegs; i++) {
            emp = OperacionesFichero.recorrerFichero(raf, i, numRegs);
            if (emp != null) {
                if (cif.compareToIgnoreCase(emp.getCif()) != 0) {
                    if(OperacionesFichero.grabar(ficheroTemp, emp, j, numRegs)) {
                        j++;
                    }
                } else {
                    exito = true;
                }
            }
        }
    } catch (EOFException e) {
    } finally {
        if (raf != null) {
            raf.close();
        }
        if (exito == true) {
            fichero.delete();
            if (!ficheroTemp.renameTo(fichero)) {
                throw new IOException(" Error al renombrar el fichero temporal ");
            }
        } else {
            ficheroTemp.delete();
        }
    }
    return exito;
}
```

BÚSQUEDA DE UN OBJETO EN EL FICHERO:

```
public static int busquedaFichero (File fichero, String cif, int numRegs) throws IOException {

    Empresa emp;
    int i;
    int cifExits = - 1;
    RandomAccessFile raf = null;

    try {

        raf = new RandomAccessFile(fichero, "r");
        for (i = 0; i < numRegs; i++) {

            emp = OperacionesFichero.recorrerFichero(raf, i, numRegs);
            if (cif.compareToIgnoreCase(emp.getCif()) == 0) {
                System.out.println("\n ----- CIF encontrado ----- \n");
                cifExits = i;
            }
        }
    } catch(EOFException e) {
    } finally {

        if (raf != null) {
            raf.close();
        }
    }

    if (cifExits == - 1) {

        System.out.println("\n ----- CIF no encontrado / CIF libre ----- \n");
    }
    return cifExits;
}
```

Resumen Programación 3ºEVA

POSICIONARSE EN UN OBJETO DETERMINADO EN EL FICHERO:

```
156     public static Empresa recorrerFichero (RandomAccessFile raf, int pos, int numRegs) throws IOException {
157
158         if (pos >= 0 && pos <= numRegs) {
159
160             raf.seek(pos * Empresa.tamanoReg);
161             return new Empresa(raf.readUTF(), raf.readUTF(), raf.readUTF(), raf.readUTF());
162         }
163         else {
164
165             return null;
166         }
167     }
168 }
```

Tema 10: Interfaz Gráfica (GUI)

INTERFAZ GRÁFICA: Conjunto de componentes gráficos que posibilitan la interacción entre el usuario y la aplicación

COMPONENTES: Objetos (jButton, jList,…) que heredan de la clase base Button, List,…

CONTAINER: Objeto que hereda de la clase Container y se encarga de alojar componentes u otros containers

FRAME: Container principal que contendrá los componentes de la GUI

LAYOUT: Componente que distribuye el espacio de un container especificando un criterio de distribución para los componentes.

TIPOS:

A. *FREE DESIGN: Distribuye los componentes de forma que cada uno mantiene una situación relativa a otro y/o al propio container*
B. *ABSOLUTELAYOUT: Distribuye los componentes sin mantener una correlación entre ello y/o con el propio container, es decir sin ninguna restricción de posicionamiento*
C. *FLOWLAYOUT: Distribuye los componentes uno al lado del otro en la parte superior del container siguiendo una determinada alineación*
D. *BORDERLAYOUT: Divide el espacio del container en 5 regiones admitiendo un único componente por región*
   - *REGIONES:*

	NORTH	
WEST	CENTER	EAST
	SOUTH	

E. *GRIDLAYOUT: Distribuye el espacio en n filas y n columnas*
   //todas las celdas serán del mismo tamaño
F. *GRIDBAGLAYOUT: Distribuye el espacio como el GridLayout, pero permitiendo a cada componente ocupar, más de una celda*

Resumen Programación 3ºEVA

Creación de tablas:

Para trabajar con tablas debemos definir una variable de tipo DefaultTableModel y dos métodos para trabajar con la tabla, uno que diseñará el modelo de la misma y otro para insertar datos, además de uno que permitirá vaciar la misma

CREAR EL MODELO DE LA TABLA:

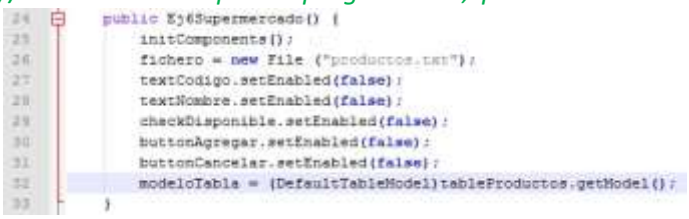
```
private DefaultTableModel crearModelo () {

    String [] columnas = new String [3];
    columnas [0] = "CIF";
    columnas [1] = "Razon Social";
    columnas [2] = "Direccion";

    DefaultTableModel modelo = new DefaultTableModel(null, columnas);

    return modelo;
}
```

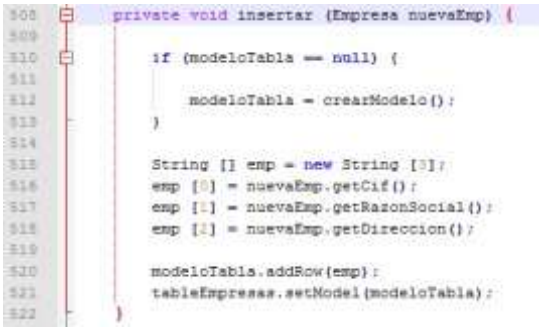
//Modelo creado por el programador, podríamos usar Object[]



```
public Ej6Supermercado() {
    initComponents();
    fichero = new File ("productos.txt");
    textCodigo.setEnabled(false);
    textNombre.setEnabled(false);
    checkDisponible.setEnabled(false);
    buttonAgregar.setEnabled(false);
    buttonCancelar.setEnabled(false);
    modeloTabla = (DefaultTableModel)tableProductos.getModel();
}
```

//Modelo creado mediante GUI

INSERTAR:



```
private void insertar (Empresa nuevaEmp) {

    if (modeloTabla == null) {

        modeloTabla = crearModelo();
    }

    String [] emp = new String [3];
    emp [0] = nuevaEmp.getCif();
    emp [1] = nuevaEmp.getRazonSocial();
    emp [2] = nuevaEmp.getDireccion();

    modeloTabla.addRow(emp);
    tableEmpresas.setModel(modeloTabla);
}
```

//Si el modelo fue creado mediante GUI no pondremos el if

VACIAR LA TABLA:

//Modelo creado por el programador

```
400         if (tableEmpresas.getRowCount() > 0) {
401
402             modeloTabla = crearModelo();
403             tableEmpresas.setModel(modeloTabla);
404         }
```

```
386         if (tableProductos.getRowCount() > 0) {
387             int totalTuplas = tableProductos.getRowCount();
388
389             for (i = 0; totalTuplas > i; i++) {
390
391                 modeloTabla.removeRow(0);
392             }
393         }
```

//Modelo creado mediante GUI

Creación de listas:

Para trabajar con listas debemos definir una variable de tipo DefaultListModel y usaremos los siguientes 3 métodos:

CREAR MODELO:

En el constructor de la clase usaremos el generado mediante GUI pondremos la siguiente línea de código:

```
modeloLista = new DefaultListModel();
```

INSERTAR:

```
modeloLista.addElement(objeto);

jList.setModel(modeloLista);
```

VACIAR LA LISTA:

```
modeloLista.clear();

jList.setModel(modeloLista);
```

Creación de ComboBox:

Para trabajar con ComboBox debemos definir una variable de tipo DefaultComboBoxModel y usaremos los siguientes 3 métodos:

CREAR MODELO E INSERTAR:

```
private DefaultComboBoxModel crearModeloCmboxPersonas (File fichero) {

    DefaultComboBoxModel modelo = new DefaultComboBoxModel();
    ObjectInputStream ois = null;
    Persona pers;

    try {

        ois = new ObjectInputStream(new FileInputStream(fichero));

        modelo.addElement(" Selecciona una persona de la lista ");
        do {

            pers = (Persona) ois.readObject();
            modelo.addElement(pers.getNombre());
        } while (true);
    } catch (EOFException e) {
    } catch (IOException | ClassNotFoundException e) {
    } finally {

        if (ois != null) {

            try {
                ois.close();

            } catch (IOException e) {
            }
        }
    }

    return modelo;
}
```

//Modelo creado por el programador con inserción de datos

VACIAR LA LISTA:

modeloComboBoxLista = crearModeloComboBoxLista(fichero); //Método anterior

comboBoxPeliculaMod.setModel(modeloComboBoxLista);

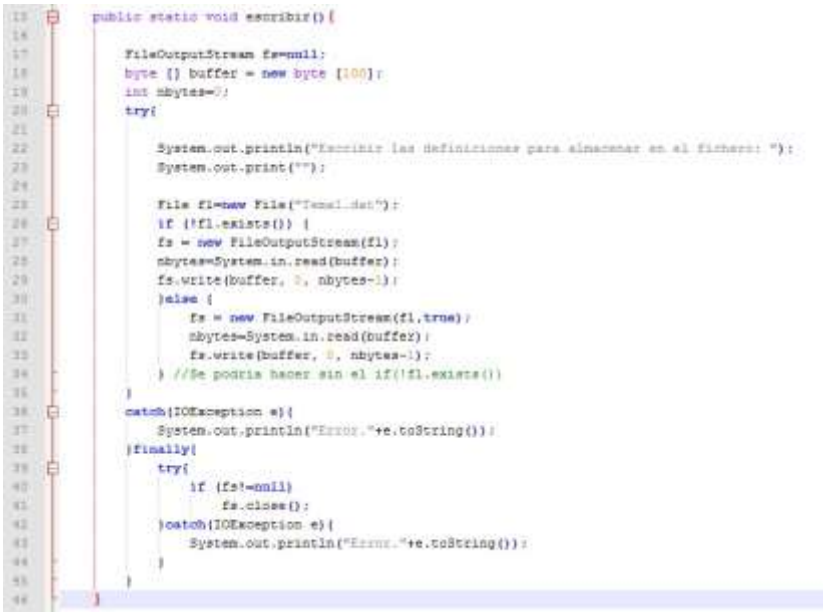
Otros métodos de trabajo con GUI:

A. *setEnabled(boolean); //Activa/Desactiva el componente*
B. *getText(String); //Recupera el texto de un jLabel/jTextField*
C. *setText(String); //Cambia el texto de un jLabel/jTextField*
D. *isSelected(); //Devuelve un booleano para un jRadioButton/jCheckBox seleccionado o no*
E. *getSelectedItem(); //Devuelve un objeto*
F. *getSelectIndex(); //Devuelve el índice del ítem seleccionado*
G. *getRowCount(); //Devuelve el total de filas*
H. *removeRow(index); //Elimina la fila seleccionada*

Capturas de Código:

FLUJO DE BYTES:

//Altas



```
public static void escribir(){

    FileOutputStream fs=null;
    byte [] buffer = new byte [100];
    int nbytes=0;
    try{

        System.out.println("Escribir las definiciones para almacenar en el fichero: ");
        System.out.print("");

        File fl=new File("Temal.dat");
        if (!fl.exists()) {
        fs = new FileOutputStream(fl);
        nbytes=System.in.read(buffer);
        fs.write(buffer, 0, nbytes-1);
        }else {
            fs = new FileOutputStream(fl,true);
            nbytes=System.in.read(buffer);
            fs.write(buffer, 0, nbytes-1);
        } //Se podria hacer sin el if(!fl.exists())
    }
    catch(IOException e){
        System.out.println("Error."+e.toString());
    }finally{
        try{
            if (fs!=null)
                fs.close();
        }catch(IOException e){
            System.out.println("Error."+e.toString());
        }
    }
}
```

```
public static void BajasLogicas() throws FileNotFoundException, IOException{
    DataOutputStream dos=null;
    DataInputStream dis=null;
    File fichero=null;
    String nombreF, nombre=null;
    char marca;
    boolean b=true;

try{
    fichero=new File("datos.dat");
    dis=new DataInputStream(new BufferedInputStream(new FileInputStream(fichero)));
    dos=new DataOutputStream(new BufferedOutputStream(new FileOutputStream(fichero)));
do{
    marca=dis.readChar();
    if(marca==''){
        nombreF=dis.readUTF();
        if(nombre.compareToIgnoreCase(nombreF)==0){
            marca='*';
            dos.writeChar(marca);
            b=false;
        }
    } else{
        System.out.println("registro ya dado debaja lógica");
        b=false;
    }
}while(b);
    }catch(IOException e){
            System.out.println("Error: "+ e.getMessage());
    }finally{
        if(dos != null)
            dos.close();
        if(dis != null)
            dis.close();
```

//Bajas

//Lectura

```
11  public class Leer {
12      public static void leer(){
13          FileInputStream fe=null;
14          byte [] buffer = new byte [100];
15          int nbytes;
16
17          try{
18
19              fe = new FileInputStream("tema1.dat");
20              nbytes=fe.read(buffer, 0, 100);
21              String str=new String(buffer, 0, nbytes);
22              System.out.println(str);
23
24          }catch(IOException e){
25              System.out.println("Error."+e.toString());
26          }finally{
27              try{
28                  if (fe!=null)
29                      fe.close();
30               }catch(IOException e){
31                  System.out.println("Error."+e.toString());
32              }
33          }
34      }
35  }
```

Resumen Programación 3ºEVA

FLUJO DE DATOS:

//Altas

```
11  public class EscribirDatos {
12      public static void escribirDatos(File fichero) throws IOException{
13      BufferedReader lee = new BufferedReader(new InputStreamReader(System.in));
14      char resp;
15      DataOutputStream dos=null;
16      String nombre;
17      String direccion;
18      int telefono;
19
20      try{
21          dos=new DataOutputStream(new FileOutputStream(fichero,true));
22          do{
23              System.out.print("Escribe el nombre: ");
24              nombre = lee.readLine();
25              System.out.print("Escribe la direccion: ");
26              direccion=lee.readLine();
27              System.out.print("Escribe el telefono: ");
28              telefono=Integer.parseInt(lee.readLine());
29              dos.writeUTF( nombre);
30              dos.writeUTF(direccion);
31              dos.writeInt(telefono);
32
33              System.out.print("Quiere escribir otro registro? (s/n): ");
34                  resp=lee.readLine().charAt(0);
35          }while(resp == 's');
36      }catch(IOException e){
37          System.out.println("Error: "+ e.getMessage());
38      }finally{
39          if(dos != null)
40              dos.close();
41          }
42      }
43  }
```

//Bajas

Resumen Programación 3ºEVA

```java
public static void Borrar(File fichero,File ficheroTemp) throws FileNotFoundException, IOException{
    BufferedReader lee = new BufferedReader(new InputStreamReader(System.in));
    DataOutputStream dos=null;
    DataInputStream dis=null;
    String nombreF, nombre, direccion;
    int telefono;
    boolean b=true;

    try{
        System.out.print("Escribe el nombre del registro a borrar: ");
        nombre=lee.readLine();
        dis=new DataInputStream(new FileInputStream(fichero));
        dos=new DataOutputStream(new FileOutputStream(ficheroTemp));
        do{
            nombreF=dis.readUTF();
            direccion=dis.readUTF();
            telefono=dis.readInt();
            if(nombre.compareToIgnoreCase(nombreF)!=0){
                dos.writeUTF(nombreF);
                dos.writeUTF(direccion);
                dos.writeInt(telefono);
            }
          }while(b); // Mientras b sea verdadero significa que no llego el fin de fichero
    }catch(EOFException e){
        System.out.println("Fin de fichero");
    }finally{
        if(dos != null)
            dos.close();
        if(dis != null) {
            dis.close();
        }
        fichero.delete();
        ficheroTemp.renameTo(fichero);
```

```java
public static void leerDatos(File fichero) throws FileNotFoundException, IOException{
    DataInputStream dis = null;
    String nombre;
    String direccion;
    int telefono;
    try{

        dis=new DataInputStream(new FileInputStream(fichero));
        do{
            nombre=dis.readUTF();
            direccion=dis.readUTF();
            telefono=dis.readInt();

            System.out.println(nombre);
            System.out.println(direccion);
            System.out.println(telefono);
            System.out.println();

        }while(true);
    }catch (EOFException e){
        System.out.println("Fin de la lectura del archivo");
    }finally{
        if(dis != null)
            dis.close();
    }
}
```

//Lectura

Resumen Programación 3ºEVA

FICHEROS RANDOM:

```
28  public static void visualizaFichero (File fichero, int numRegs) throws IOException {
29
30      RandomAccessFile raf = null;
31      Empresa emp;
32      int i, b = 0;
33      String cabecera = "";
34      String linea = "";
35
36      try {
37
38          raf = new RandomAccessFile(fichero, "r");
39          for (i = 0; i < numRegs; i++) {
40
41              if (b%5 == 0) {
42
43                  System.out.print(linea + cabecera + linea);
44              }
45
46              emp = OperacionesFichero.recorrerFichero(raf, i, numRegs);
47              System.out.printf("\n  %-5s \t \t %-20s \t \t %-20s \t \t %-9s ", emp.getCif(), emp.getRazonSocial(), emp.getDireccion(), emp.getTelefono());
48              b++;
49          }
50
51          System.out.print(linea);
52      } catch (EOFException e) {
53      } finally {
54
55          if (raf != null) {
56
57              raf.close();
58          }
59      }
60  }
```

//Visualizar

Resumen Programación 3ºEVA

//Altas

```
public static int altas (File fichero, int numRegs) throws IOException, ClassNotFoundException {

    char decision;
    Empresa emp;
    String cif, razonSocial, direccion, telefono;
    int error, cifExitst, i = 0;
    boolean exito;

    Scanner lee = new Scanner (System.in);

    do {

        //peticion de datos
        emp = new Empresa(cif, razonSocial, direccion, telefono);
        exito = OperacionesFichero.grabar(fichero, emp, numRegs, numRegs);

        if (exito == true) {

            i++;
            numRegs++;
            System.out.println("\n Numero de registros incrementado en 1 \n");
        }

        decision = Menu.menu2();
    }
    while (decision != 'N');

    System.out.println("\n Altas finalizadas \n" +
                       "\n Numero de registros establecido a: " + numRegs + "\n");
    Visualizar.visualizaSalidaOpciones();

    return numRegs;
}
```

```
public static int bajas (File fichero, int numRegs) throws IOException {

    String cif;
    boolean exito;
    char decision;

    Scanner lee = new Scanner (System.in);

    do {

        Visualizar.visualizaFichero(fichero, numRegs);

        System.out.println("\n Introduce el CIF de la empresa a dar de baja: ");
        cif = lee.nextLine();
        exito = OperacionesFichero.actualizarFichero(fichero, cif, numRegs);

        if (exito == true) {

            System.out.println("\n Empresa eliminada con exito \n");
            numRegs--;
            System.out.println("\n Numero de registros reducido en 1 \n");
        }
        else {

            System.out.println("\n Empresa no encontrada, no se han realizado cambios \n");
        }

        decision = Menu.menu2();
    }
    while (decision != 'N');

    System.out.println("\n Bajas finalizadas \n" +
                       "\n Numero de registros establecido a: " + numRegs + "\n");
    Visualizar.visualizaSalidaOpciones();

    return numRegs;
}
```

//Bajas

Resumen Programación 3ºEVA

```java
public static void modificar (File fichero, int numRegs) throws IOException {

    String cif;
    int cifExits;
    boolean exito;
    char decision;

    Scanner lee = new Scanner (System.in);

    do {

        Visualizar.visualizaFichero(fichero, numRegs);

        System.out.println("\n Introduce el CIF de la empresa a dar de baja: ");
        cif = lee.nextLine();
        cifExits = OperacionesFichero.busquedaFichero(fichero, cif, numRegs);
        if (cifExits != -1) {
            Modificaciones.modificarEmpresa(fichero, cifExits, numRegs);
        } else {
            System.out.println("\n Empresa no encontrada, no se han realizado cambios \n");
        }
        decision = Menu.menu2();
    } while (decision != 'N');
    Visualizar.visualizaSalidaOpciones();
}
```

```java
//Modificaciones
public static void modificarEmpresa (File fichero, int cifExists, int numRegs) throws IOException {

    RandomAccessFile raf = null;
    int eleccion;
    Empresa emp = null;
    String razonSocial, direccion, telefono;
    int error;
    Scanner lee = new Scanner (System.in);

    try {
        raf = new RandomAccessFile(fichero, "r");
        emp = OperacionesFichero.recorrerFichero(raf, cifExists, numRegs);
    } catch (EOFException e) {
    } finally {
        if (raf != null) {
            raf.close();
        }
    }
    eleccion = Menu.menu3();
    switch (eleccion) {
        case 0:
            Visualizar.visualizaSalidaOpciones();
            break;
        case 1:
            do {
                System.out.println("\n Razon social actual: " + emp.getRazonSocial() + "\n Introduce la nueva razon social de la empresa: ");
                razonSocial = lee.nextLine();
                if (razonSocial.compareToIgnoreCase(emp.getRazonSocial()) == 0) {
                    Visualizar.visualizarErrorDatos();
                }
            } while (razonSocial.compareToIgnoreCase(emp.getRazonSocial()) == 1);
            emp = new Empresa(emp.getCif(), razonSocial, emp.getDireccion(), emp.getTelefono());
            OperacionesFichero.grabar(fichero, emp, cifExists, numRegs);
            break;
        //resto modificaciones
        default:
            Visualizar.visualizaError3();
            break;
    }
}
```