

## OBJETOS SERIALIZABLES

Hemos estudiado como se guardan los tipos primitivos en un fichero, pero por ejemplo, tenemos un objeto de tipo empleado con varios atributos (nombre, dirección, salario, departamento, etc.) y queremos guardarlo en un fichero tendríamos que guardar cada atributo que forma parte del objeto por separado, esto es engorroso si tenemos varios objetos. Por ello Java nos permite guardar objetos en ficheros binarios para poder hacerlo, el objeto tiene que implementar la interfaz `Serializable` que dispone de una serie de métodos con los que podemos guardar y leer objetos en ficheros binarios.

### Escritura de Objetos en un Fichero

Para que un objeto pueda ser almacenado en disco, es necesario que la clase a la que pertenece sea **serializable**. Esta característica la poseen todas las clases que implementan la interfaz `java.io.Serializable`.

La **serialización** es el proceso por el cual un objeto o una colección de objetos se convierten en una serie de bytes que pueden ser almacenados en un fichero y recuperado posteriormente para su uso. Se dice que el objeto u objetos tienen **persistencia**. Cuando serializamos un objeto, almacenamos la estructura de la clase y todos los objetos referenciados por esta.

La interfaz **Serializable** no contiene ningún método, basta con que una clase la implemente para que sus objetos puedan ser serializados por la máquina virtual y por lo tanto almacenada en disco.

**Ejemplo:** Crea una clase `Persona` cuyos objetos encapsulan el nombre y la edad de una persona. Estos objetos pueden ser transferidos a disco, ya que `Persona` implementa la interfaz `Serializable`.

```
package ejemplos070objetos01Persona;
import java.io.Serializable;

public class Persona implements Serializable {
    private static final long serialVersionUID = 1L;
    //atributos
    String nombre;
    int edad;
    // constructores
    public Persona(String nombre, int edad) {
        super();
        this.nombre = nombre;
        this.edad = edad;
    }
    public Persona() {
        super();
    }
    // metodos get y set
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
```

## Unidad didáctica 1 MANEJO DE FICHEROS.

---

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

El primer paso es crear un objeto `FileOutputStream` que permita añadir información al fichero o sobrescribirla, para ello utilizamos los **constructores**:

```
FileOutputStream(File fichero, boolean append);
FileOutputStream(String path, boolean append);
```

Para escribir objetos en disco el segundo paso es crear un objeto de la clase `ObjectOutputStream`, cuyos constructores son:

```
ObjectOutputStream(); // Para crear un objeto que permite escribir objetos de
                        // java sobre cualquier dispositivo.
ObjectOutputStream(OutputStream); // Crea un flujo que permite escribir
                                    // objetos de java en el objeto OutputStream que se pasa como argumento.
```

Una vez creado el objeto, la clase dispone de los mismos métodos que `DataOutputStream` más:

```
void writeObject(Object) // Para escribir el objeto en el disco. Propagan la
                           // excepción IOException.
```

```
package ejemplos07Objetos01Persona;
/*
 * este ejemplo escribe objetos del tipo Persona en un fichero en
disco
 */
```

```
import java.io.*;
```

```
public class Ej01EscrituraObjetoPersona {
    public static void main(final String[] args) {
        // creamos los objetos que nos permiten escribir
        FileOutputStream fs = null;
        ObjectOutputStream os = null;

        try{
            fs = new FileOutputStream("Personas.obj", true);
            os = new ObjectOutputStream(fs);
            // declaramos el objeto Persona usando un constructor y
escribimos en el disco
            Persona p = new Persona("Marta Perez", 32);
            os.writeObject(p);
        }
    }
}
```

## Unidad didáctica 1 MANEJO DE FICHEROS.

---

```
// escribimos pasando la creación del objeto
os.writeObject(new Persona("Ana Sánchez", 27));
p = new Persona();
p.setEdad(44);
p.setNombre("Pedro Martínez");
os.writeObject(p);

// cerramos el fichero
os.close();
}catch(FileNotFoundException fne){
    System.out.println("Error en el fichero. nO EXISTE");
}catch(IOException ioe){
    System.out.println("Error E/L");
}
System.out.println("Fin de la escritura");
}
}
```

### Lectura de Objetos de un Fichero

Cuando se recupera un objeto del disco mediante la llamada a `readObject()`, se produce la **deserialización** del objeto que consiste en la reconstrucción de éste a partir de la información recuperada.

Durante este proceso, los datos miembro no serializables (los que han sido heredados de una clase no serializable) serán inicializados utilizando el constructor por defecto de su clase. Por el contrario, los datos miembro de la clase objeto serializado serán restaurados con los valores almacenados.

Para leer objetos de un fichero que han sido almacenados mediante `ObjectOutputStream`, hay que utilizar la clase `ObjectInputStream`.

El procedimiento es similar a los anteriores

El primer paso es crear un objeto `FileInputStream` que permita recuperar información del fichero, para ello utilizamos los **constructores**:

```
FileInputStream(File fichero);
FileInputStream (String path);
```

El segundo paso es a partir del objeto `FileInputStream` crear un objeto `ObjectInputStream` para realizar la escritura. El **constructor** es:

```
ObjectInputStream(); // Para crear un objeto que permite leer objetos de java
sobre cualquier dispositivo. 31
ObjectInputStream (InputStream); // Crea un flujo que permite leer objetos
de java en el objeto InputStream que se pasa como argumento.
```

La clase `OuputInputStream` proporciona métodos para leer datos desde un fichero:

Mismos métodos que `DataOutputStream` más:

## Unidad didáctica 1 MANEJO DE FICHEROS.

---

`object readObject();` // Para leer un objeto asociado al flujo y devolverlo de tipo `Object`. Hay que hacer obligatoriamente un cast a la clase a la que lo vamos a convertir.

```
Persona obj=(Persona) flujolectura.readObject();
```

Todos los métodos propagan una excepción de la clase `IOException`.

El método `readObject()` propaga la excepción `ClassNotFoundException` (Cuando se intenta hacer un cast al objeto que se ha leído de una clase que no es), y la excepción `StreamCorruptedException` (cuando se intenta hacer más de una operación de lectura a través de la clase `ObjectInputStream`).

```
package ejemplos07Objetos01Persona;
```

```
// Ejemplo que lee el primer objeto de un fichero en disco
```

```
import java.io.*;
```

```
public class LecturaObjetoPersona01 {
    public static void main(final String[] args) {
        System.out.println("Nombre \t Edad" );
        try{
            // abrimos el fichero para lectura
            FileInputStream fs = new
FileInputStream("Personas.obj");
            ObjectInputStream os = new ObjectInputStream(fs);

            // solo recupera un objeto
            // os debe realizar un casting al tipo original
            Persona p = (Persona) os.readObject();
            System.out.println(p.getNombre() +"\t" +p.getEdad());

            os.close();
        }catch(ClassNotFoundException cnf){
            System.out.println("Error la clase");
        }catch(FileNotFoundException fnfe){
            System.out.println("Error en el fichero");
        }catch(IOException ioe){
            System.out.println("Error E/L");
        }
    }
}
```

El siguiente ejemplo lee todos los objetos guardados en el fichero utilizando la excepción `EOFException` para detectar el final del fichero.

```
package ejemplos07Objetos01Persona;
```

```
//Ejemplo que lee todos los objeto de un fichero en disco
```

```
import java.io.*;
```

```
public class Ej03LecturaObjetosPersona {
    public static void main(final String[] args) throws IOException {
```

## Unidad didáctica 1 MANEJO DE FICHEROS.

---

```
//abrimos el fichero de lectura
FileInputStream fs = null;
ObjectInputStream os = null;

System.out.println("Nombre \t Edad" );
try{
    fs = new FileInputStream("Personas.obj");
    os = new ObjectInputStream(fs);
    while(true){ // lectura del fichero mientras haya
        // os debe realizar un castingal tipo original
        Persona p = (Persona)os.readObject();
        System.out.println(p.getNombre() +"\t"
+p.getEdad());
    }
} catch(ClassNotFoundException cnf){
    System.out.println("Error la clase no existe");
} catch(FileNotFoundException fnfe){
    System.out.println("Error en el fichero");
} catch(EOFException eo){
    System.out.println("Fin del fichero");
}
} catch(IOException ioe){
    System.out.println("Error");
}
os.close();
}
```

### **Problemas con la clase ObjectOutputStream**

1º) Cuando escribimos un objeto, es como si guardara el array de bytes en el interior. Si cambiamos los valores de los atributos de ese objeto y volvemos a escribirlo el **ObjetOutputStream** lo escribe nuevamente, pero con los datos antiguos. Da la impresión de que no se entera del cambio y no recalcula los bytes que va a escribir en el fichero.

Ejemplo: Si escribimos así, con un solo new

```
Persona p = new Persona(); // un único new fuera del bucle
for(int i = 0; i < 5; i++){
    p.setPersona(i); /* cambiamos los datos de p, pero no
creamos un nuevo objeto con new */
    oos.writeObject(p);
}
oos.close(); // se cierra al terminar
```

El siguiente ejemplo introduce 5 personas y a la hora de leerlos solo lee el primer objeto, es porque no se ha instanciado una nueva persona.

```
package ejemplos07Objetos01Persona;
// Ejemplo que no cambia el valor del objeto por estar instanciado una
única vez
```

## Unidad didáctica 1 MANEJO DE FICHEROS.

---

```
import java.io.*;
import aaIntroducirDatos.IntroducirDatos;

public class Ej04EscrituraLecturaVariosObjetos {

    static FileOutputStream fs = null;
    static ObjectOutputStream os = null;

    public static void main(final String[] args) {
        escrituraObjetos();
        lecturaObjetos();
    }

    private static void escrituraObjetos() {
        // creamos los objetos que nos permiten escribir

        try{
            fs = new FileOutputStream("Personas01.txt");
            os = new ObjectOutputStream(fs);
            //instanciamos la clase Persona una vez fuera del
            bucle

            Persona p = new Persona();
            for (int i= 0; i<5; i++){
                //introducimos los datos por teclado
                // p = new Persona();

                p.setNombre(IntroducirDatos.introducirDatos("Introduce el nombre
del empleado: "));

                p.setEdad(Integer.parseInt(IntroducirDatos.introducirDatos("Intro
duce la edad del empleado: ")));
                //escribimos el objeto p en el disco
                os.writeObject(p);
            }
            os.close();
        }catch(FileNotFoundException fne){
            System.out.println("Error en el fichero");
        }catch(IOException ioe){
            System.out.println("Error E/L");
        }
        System.out.println("Fin del método escribir");
    }

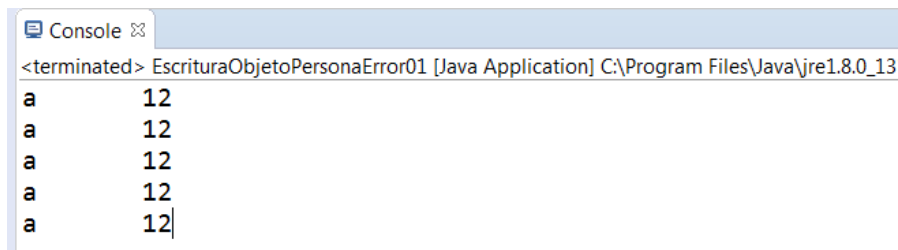
    private static void lecturaObjetos() {
        // creamos los objetos que nos permiten leer del disco
        objetos

        //abrimos el fichero de lectura
        FileInputStream fs = null;
        ObjectInputStream os = null;
```

## Unidad didáctica 1 MANEJO DE FICHEROS.

```
System.out.println("Nombre \t Edad" );
try{
    fs = new FileInputStream("Personas01.txt");
    os = new ObjectOutputStream(fs);
    while(true){ // lectura del fichero mientras haya
        // os debe realizar un castingal tipo original
        Persona p = (Persona)os.readObject();
        System.out.println(p.getNombre() +"\t"
+p.getEdad());
    }
}catch(ClassNotFoundException cnf){
    System.out.println("Error la clase");
}catch(FileNotFoundException fnfe){
    System.out.println("Error en el fichero");
}catch(IOException ioe){
}
}
```

Con el método de lectura obtenemos 5 veces el primer objeto que hemos introducido por teclado.



```
<terminated> EscrituraObjetoPersonaError01 [Java Application] C:\Program Files\Java\jre1.8.0_13
a      12
a      12
a      12
a      12
a      12
```

Esto puede evitarse de tres formas:

- Haciendo un **new** de cada objeto que queramos escribir, sin reaprovechar instancias. Esto es lo que se ha hecho en el código anterior.
- Usar el método **writeUnshared()** en vez de **writeObject()**. Este método funcionará bien si cambiamos **TODOS** los atributos de la clase **Persona**. Si no modificamos uno de los atributos, obtendremos resultados extraños en ese atributo.
- Llamando al método **reset()** de **ObjectOutputStream** después de escribir cada objeto. Aunque funciona, no parece demasiado ortodoxo.

La solución más idónea es instanciar siempre el objeto que queramos escribir

```
package ejemplos07Objetos01Persona;
// Ejemplo que cambia el valor del objeto por estar instanciado una
única vez
```

```
import java.io.*;
import aaIntroducirDatos.IntroducirDatos;

public class Ej04EscrituraLecturaVariosObjetos {
```

## Unidad didáctica 1 MANEJO DE FICHEROS.

---

```
static FileOutputStream fs = null;
static ObjectOutputStream os = null;

public static void main(final String[] args) {
    escrituraObjetos();
    lecturaObjetos();
}

private static void escrituraObjetos() {
    // creamos los objetos que nos permiten escribir

    try{
        fs = new FileOutputStream("Personas01.txt");
        os = new ObjectOutputStream(fs);
        //instanciamos la clase Persona una vez fuera del
        bucle

        for (int i= 0; i<5; i++){
            //introducimos los datos por teclado
            Persona p = new Persona();

            p.setNombre(IntroducirDatos.introducirDatos("Introduce el nombre
del empleado: "));

            p.setEdad(Integer.parseInt(IntroducirDatos.introducirDatos("Intro
duce la edad del empleado: ")));
            //escribimos el objeto p en el disco
            os.writeObject(p);
        }
        os.close();
    }catch(FileNotFoundException fne){
        System.out.println("Error en el fichero");
    }catch(IOException ioe){
        System.out.println("Error E/L");
    }
    System.out.println("Fin del método escribir");
}

private static void lecturaObjetos() {
    // creamos los objetos que nos permiten leer del disco
    objetos

    //abrimos el fichero de lectura
    FileInputStream fs = null;
    ObjectInputStream os = null;
    System.out.println("Nombre \t Edad" );
    try{
        fs = new FileInputStream("Personas01.txt");
        os = new ObjectInputStream(fs);
        while(true){ // lectura del fichero mientras haya
        objetos
```



## Unidad didáctica 1 MANEJO DE FICHEROS.

```
// os debe realizar un casting tipo original
Persona p = (Persona)os.readObject();
System.out.println(p.getNombre() + "\t"
+p.getEdad());
    }
} catch (ClassNotFoundException cnf){
    System.out.println("Error la clase");
} catch (FileNotFoundException fnfe){
    System.out.println("Error en el fichero");
} catch (IOException ioe){
}
}
```

2º) Un segundo problema es que al instanciar el objeto **ObjectOutputStream**, escribe unos bytes de cabecera en el fichero, antes incluso de que escribamos nada. Como el **ObjectInputStream** lee correctamente estos bytes de cabecera, aparentemente no pasa nada y ni siquiera nos enteramos que existen.

El problema se presenta si escribimos unos datos en el fichero y lo cerramos. Luego volvemos a abrirlo para añadir datos, creando un nuevo **ObjectOutputStream** así

```
/* El true del final indica que se abre el fichero para añadir
datos al final del fichero */
```

```
ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(fichero,true));
```

Esto escribe una nueva cabecera justo al final del fichero. Luego se irán añadiendo los objetos que vayamos escribiendo. El fichero contendrá lo del dibujo, con dos cabeceras.

Contenido del fichero							
Primera sesión con el fichero				Segunda sesión con el fichero			
Cabecera	Persona1	Persona2	Persona3	Cabecera	Persona4	Person5	Persona6

Cuando leemos el fichero, lo primero será crear el **ObjectInputStream**, este leerá la cabecera del principio y luego se pone a leer objetos. Cuando llegamos a la segunda cabecera que se añadió al abrir por segunda vez el fichero para añadirle datos, obtendremos un error **StreamCorruptedException** y no podremos leer más objetos.

Una solución es evidente, no usar más que un solo **ObjectOuptutStream** para escribir todo el fichero. Sin embargo, esto no es siempre posible.

Una solución es hacernos nuestro propio **ObjectOutputStream**, heredando del original y redefiniendo el método **writeStreamHeader()**, vacío, para que no haga nada.

```
/* Redefinición del método de escribir la cabecera para que no
haga nada. */
```

```
protected void writeStreamHeader() throws IOException
{
}
```



## Unidad didáctica 1 MANEJO DE FICHEROS.

---

Si el fichero se sobrescribe nunca se generará el problema de tener varias cabeceras.

```
package ejemplos09ObjetosAlumnos;
```

```
import java.io.*;
```

```
// ejemplo que nos permite escribir y leer objetos Alumnos  
// el fichero se sobrescribe no hay problemas de cabeceras
```

```
public class LecturaEscrituraObjetos01 {  
  
    public static void main(String[] args) {  
        escrituraObjetos();  
        lecturaObjetos();  
    }  
  
    static void escrituraObjetos(){  
        FileOutputStream fo = null;  
        ObjectOutputStream oo = null;  
  
        try{  
            fo = new FileOutputStream("Alumnos.dat");  
            oo = new ObjectOutputStream(fo);  
  
            //creo los objetos a escribir  
            Alumnos alumno1 = new Alumnos("María", "Vigo", 25 );  
            Alumnos alumno2 = new Alumnos("Juan", "Madrid", 32 );  
            Alumnos alumno3 = new Alumnos("Marta", "Eibar", 22 );  
  
            // escribimos en el fichero  
            oo.writeObject(alumno1);  
            oo.writeObject(alumno2);  
            oo.writeObject(alumno3);  
  
        }catch(FileNotFoundException fn){  
            System.out.println("Error. Fichero no existe");  
        }catch(IOException io){  
            System.out.println("Error de escritura");  
        }  
    }  
} // fin metodo escrituraObjetos  
  
static void lecturaObjetos(){  
    FileInputStream fi = null;  
    ObjectInputStream oi = null;  
  
    try{  
        fi = new FileInputStream("Alumnos.dat");  
        oi = new ObjectInputStream(fi);
```

## Unidad didáctica 1 MANEJO DE FICHEROS.

---

```
        while(true){
            //creo el objeto donde voy a guardar los datos
leidos del disco
            Alumnos alumno = (Alumnos) oi.readObject();
            System.out.println(alumno);
        }

    }catch(ClassNotFoundException cn){
        System.out.println("Error en la clase alumnos");
    }catch(FileNotFoundException fn){
        System.out.println("Error. Fichero no existe");
    }catch(EOFException eof){
        System.out.println("Fin de la lectura");
    }catch(IOException io){
        System.out.println("Error de lectura/escritura");
    }
} // fin metodo lecturaObjetos
}
```

Si el fichero no se sobrescribe y le añadimos nuevos objetos se generará un error al leer ya que se han escrito varias cabeceras.

```
package ejemplos090objetosAlumnos;
```

```
import java.io.*;
```

```
public class LecturaEscrituraObjetos02 {
    public static void main(String[] args) {
        escrituraObjetos();
        lecturaObjetos();
    }
}
```

```
static void escrituraObjetos(){
    FileOutputStream fo = null;
    ObjectOutputStream oo = null;

    try{
        fo = new FileOutputStream("Alumnos01.dat", true);
        oo = new ObjectOutputStream(fo);

        //creo los objetos a escribir
        Alumnos alumno1 = new Alumnos("María", "Vigo", 25 );
        Alumnos alumno2 = new Alumnos("Juan", "Madrid", 32 );
        Alumnos alumno3 = new Alumnos("Marta", "Eibar", 22 );

        // escribimos en el fichero
        oo.writeObject(alumno1);
        oo.writeObject(alumno2);
        oo.writeObject(alumno3);
    }
}
```

## Unidad didáctica 1 MANEJO DE FICHEROS.

---

```
    }catch(FileNotFoundException fn){
        System.out.println("Error. Fichero no existe");
    }catch(IOException io){
        System.out.println("Error de escritura");
    }
}
} // fin metodo escrituraObjetos

static void lecturaObjetos(){
    FileInputStream fi = null;
    ObjectInputStream oi = null;

    try{
        fi = new FileInputStream("Alumnos01.dat");
        oi = new ObjectInputStream(fi);

        while(true){
            //creo el objeto donde voy a guardar los datos
            leidos del disco
            Alumnos alumno = (Alumnos) oi.readObject();
            System.out.println(alumno);
        }
    }catch(ClassNotFoundException cn){
        System.out.println("Error en la clase alumnos");
    }catch(FileNotFoundException fn){
        System.out.println("Error. Fichero no existe");
    }catch(EOFException eof){
        System.out.println("Fin de la lectura");
    }catch(IOException io){
        System.out.println("Error de lectura/escritura");
    }
}
} // fin metodo lecturaObjetos
}
```

La **solución** es crear la clase `MiObjectOutputStream` que heredará de la clase `ObjectOutputStream` para sobrescribir el método `writeStreamHeader`

```
public class MiObjectOutputStream extends ObjectOutputStream {
    //Constructor sin parámetros
    protected MiObjectOutputStream() throws IOException,
    SecurityException {
        super();
    }
    //Constructor que recibe como parámetro un objeto OutputStream
    protected MiObjectOutputStream(OutputStream out) throws
    IOException {
        super(out);
    }
}
```

## Unidad didáctica 1 MANEJO DE FICHEROS.

---

```
/*redefinición del método que escribe la cabecera para que no  
haga nada en caso de que el fichero ya tenga datos */
```

```
protected void writeStreamHeader(){  
    }  
}
```

El **programa completo** podría ser:

La clase **MiObjectOutputStream**

```
package ejemplos090objetosAlumnos;
```

```
import java.io.*;
```

```
/* la clase MiObjectOutputStream quedaría con los siguientes métodos*/  
public class MiObjectOutputStream extends ObjectOutputStream {
```

```
    //Constructor sin parámetros  
    protected MiObjectOutputStream() throws IOException,  
    SecurityException {  
        super();  
    }
```

```
    //Constructor que recibe como parámetro un objeto OutputStream  
    protected MiObjectOutputStream(OutputStream out) throws  
    IOException {  
        super(out);  
    }
```

```
    /*redefinición del método que escribe la cabecera para que no  
haga  
nada en caso de que el fichero ya tenga datos  
    */  
    protected void writeStreamHeader(){  
  
    }
```

```
}// fin de la clase MiObjectOutputStream
```

El **Programa principal** con el método main podría ser:

```
package ejemplos090objetosAlumnos;
```

```
import java.io.*;
```

```
public class LecturaEscrituraObjetos03 {  
    // ejemplo que nos permite escribir y leer objetos Alumnos  
    // el fichero no se sobrescribe añade objetos al final  
    // se utiliza la clase MiObjectOutputStream para que no escriba  
la cabecera cada vez que se llama al programa
```

## Unidad didáctica 1 MANEJO DE FICHEROS.

---

```
public static void main(String[] args) throws
NumberFormatException, IOException {
    File f = new File("Alumnos.dat");
    int opcion;
    BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
    do{
        System.out.println("1.- Insertar Registros.");
        System.out.println("2.- Leer Registros.");
        System.out.println("3.- Salir.");
        System.out.println("Elegir opcion: ");
        opcion = Integer.parseInt(br.readLine());
        switch(opcion){
            case 1:
                if(f.exists()){
                    System.out.println("Alumnos.dat existe");
                    escrituraObjetosExisteFichero(f);
                }else{
                    System.out.println("Alumnos.dat no existe");
                    escrituraObjetosNuevoFichero(f);
                }
                break;
            case 2:
                lecturaObjetos(f);
                break;
            case 3:
                System.exit(0);
            default:
                System.out.println("Error en la opcion.");

                }// fin switch
        }while(opcion != 3);
    } // fin main

    static void escrituraObjetosNuevoFichero(File f){
        FileOutputStream fo = null;
        ObjectOutputStream oo = null;

        try{
            fo = new FileOutputStream(f, true);
            oo = new ObjectOutputStream(fo);

            //creo los objetos a escribir
            Alumnos alumno1 = new Alumnos("Luis", "Vigo", 25 );
            Alumnos alumno2 = new Alumnos("Carmen", "Madrid", 32

        );

            Alumnos alumno3 = new Alumnos("Iciar", "Eibar", 22 );

            System.out.println("Escribiendo registros. Espere");
```

## Unidad didáctica 1 MANEJO DE FICHEROS.

---

```
// escribimos en el fichero
oo.writeObject(alumno1);
oo.writeObject(alumno2);
oo.writeObject(alumno3);

}catch(FileNotFoundException fn){
    System.out.println("Error. Fichero no existe");
}catch(IOException io){
    System.out.println("Error de escritura");
}finally{
    try{
        if(fo != null){
            fo.close();
            oo.close();
        }
    }catch(IOException io){
        System.out.println("Error al cerrar el fichero");
    }
}
}
}

// fin metodo escrituraObjetosNuevo

static void escrituraObjetosExisteFichero(File f){
    FileOutputStream fo = null;
    MiObjectOutputStream mo = null;

    try{
        fo = new FileOutputStream(f,true);
        mo = new MiObjectOutputStream(fo);

        //creo los objetos a escribir
        Alumnos alumno1 = new Alumnos("Bea", "Vigo", 25 );
        Alumnos alumno2 = new Alumnos("Pedro", "Madrid", 32 );
        Alumnos alumno3 = new Alumnos("Iñigo", "Eibar", 28 );

        System.out.println("Añadiendo registros. Espere");

        // escribimos en el fichero
        mo.writeObject(alumno1);
        mo.writeObject(alumno2);
        mo.writeObject(alumno3);

        }catch(FileNotFoundException fn){
            System.out.println("Error. Fichero no existe");
        }catch(IOException io){
            System.out.println("Error de escritura");
        }finally{
            try{
```



## Unidad didáctica 1 MANEJO DE FICHEROS.

---

```
        if(fo != null){
            fo.close();
            mo.close();
        }
    }catch(IOException io){
        System.out.println("Error al cerrar el fichero");
    }
}
}

// fin metodo escrituraObjetosExiste

static void lecturaObjetos(File f){
    FileInputStream fi = null;
    ObjectInputStream oi = null;

    try{
        fi = new FileInputStream(f);
        oi = new ObjectInputStream(fi);

        while(true){
            //creo el objeto donde voy a guardar los datos
            leidos del disco
            Alumnos alumno = (Alumnos) oi.readObject();
            System.out.println(alumno);
        }

    }catch(ClassNotFoundException cn){
        System.out.println("Error en la clase alumnos");
    }catch(FileNotFoundException fn){
        System.out.println("Error. Fichero no existe");
    }catch(EOFException eof){
        System.out.println("Fin del la lectura");
    }catch(IOException io){
        System.out.println("Error de lectura");
    }finally{
        try{
            if(fi != null){
                fi.close();
                oi.close();
            }
        }catch(IOException io){
            System.out.println("Error al cerrar el fichero");
        }
    }
}

// fin metodo lecturaObjetos
}
```