**Tema11.FICHEROS**

**Serialización. Ficheros de Objetos en Java. Interface Serializable.**

**1. Serialización y persistencia de objetos**

La **serialización** es la transformación de un objeto en una secuencia de bytes que pueden ser posteriormente leídos para reconstruir el objeto original.

El objeto serializado pueda guardarse en un fichero o puede enviarse por red para reconstruirlo en otro lugar. Puede crearse en un sistema Windows y enviarlo. por ejemplo, a otro sistema que utilice Linux.

Guardar objetos de forma que existan cuando la aplicación haya terminado se conoce como **persistencia**.

Para poder transformar el objeto en una secuencia de bytes, el objeto debe ser **serializable**.

Un objeto es **serializable** si su clase implementa la interface Serializable.

La interface Serializable se encuentra en el paquete java.io Es una interface vacía. No contiene ningún método.

public interface Serializable{

}

Sirve para indicar que los objetos de la clase que lo implementa se pueden serializar. Solo es necesario que una clase la implemente para que la máquina virtual pueda serializar los objetos.

Si un objeto contiene atributos que son referencias a otros objetos éstos a su vez deben ser serializables.

Todos los tipos básicos Java son serializables, así como los arrays y los String.

**2. Persistencia de Objetos en Ficheros**

Para escribir objetos en un fichero binario en Java se utiliza la **clase ObjectOutputStream** derivada de OutputStream.

Un objeto ObjectOutputStream se crea a partir de un objeto FileOutputStream asociado al fichero.

El constructor de la clase es:

ObjectOutputStream(OutputStream nombre);

Lanza una excepción **IOException**.

**Por ejemplo**, las instrucciones para crear el fichero personas.dat para escritura de objetos serían éstas:

FileOutputStream fos = new FileOutputStream("personas.dat");

ObjectOutputStream salida = new ObjectOutputStream (fos);

La clase proporciona el **método writeObject(Object objeto)** para escribir el objeto en el fichero. Lanza una **IOException**

El método defaultWriteObject() de la clase ObjectOutputStream realiza de forma automática la serialización de los objetos de una clase. Este método se invoca en el método writeObject().

defaultWriteObject() escribe en el stream de salida todo lo necesario para reconstruir los objetos:

- La clase del objeto.

- Los miembros de la clase (atributos).

- Los valores de los atributos que **no** sean **static** o **transient**.

El método defaultReadObject() de la clase ObjectInputStream realiza la *deserialización* de los objetos de una clase. Este método se invoca en el método readObject().

**Ejemplo 1: de persistencia de objetos en fichero en Java**

Programa que escribe 3 objetos de tipo Persona en el fichero personas.dat.

La clase Persona es la siguiente:

//Clase Persona

import java.io.Serializable;

public class Persona implements Serializable{

private String nif;

private String nombre;

private int edad;

public Persona() {

}

public Persona(String nif, String nombre, int edad) {

this.nif = nif;

this.nombre = nombre;

this.edad = edad;

}

public int getEdad() {

return edad;

}

public void setEdad(int edad) {

this.edad = edad;

}

public String getNif() {

return nif;

}

public void setNif(String nif) {

this.nif = nif;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

}

//Clase Crear fichero

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectOutputStream;

public class CrearFpersonas {

public static void main(String[] args) {

FileOutputStream fos = null;

ObjectOutputStream salida = null;

Persona p;

try {

//Se crea el fichero

fos = new FileOutputStream("personas.dat");

salida = new ObjectOutputStream(fos);

//Se crea el primer objeto Persona

p = new Persona("12345678A","Lucas González", 30);

//Se escribe el objeto en el fichero

salida.writeObject(p);

//Se crea el segundo objeto Persona

p = new Persona("98765432B","Anacleto Jiménez", 28);

//Se escribe el objeto en el fichero

salida.writeObject(p);

//Se crea el tercer objeto Persona

p = new Persona("78234212Z","María Zapata", 35);

//Se escribe el objeto en el fichero

salida.writeObject(p);

} catch (FileNotFoundException e) {

System.out.println("1"+e.getMessage());

} catch (IOException e) {

System.out.println("2"+e.getMessage());

} finally {

try {

if(fos!=null) fos.close();

if(salida!=null) salida.close();

} catch (IOException e) {

System.out.println("3"+e.getMessage());

}

}

}

}

**3. Serialización y Herencia**

Para serializar objetos de una jerarquía solamente la clase base tiene que implementar el interface Serializable. No es necesario que las clases derivadas implementen la interfaz.

**Si una clase es serializable lo son también todas sus clases derivadas.**

**Ejemplo 2** de herencia y serialización:

Programa Java que escribe en un fichero tres objetos de tipo Empleado. Empleado es una clase derivada de la clase Persona del ejemplo anterior. Como la clase Persona es serializable, no es necesario indicar que la clase Empleado también lo es. Empleado es serializable por el hecho de heredar de Persona.

//Clase Empleado

public class Empleado extends Persona{

private double sueldo;

public Empleado(String nif, String nombre, int edad, double sueldo) {

super(nif, nombre, edad);

this.sueldo = sueldo;

}

public Empleado() {

}

public double getSueldo() {

return sueldo;

}

public void setSueldo(double sueldo) {

this.sueldo = sueldo;

}

}

//Crear fichero de empleados

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectOutputStream;

public class CrearFempleados {

public static void main(String[] args) {

FileOutputStream fos = null;

ObjectOutputStream salida = null;

Empleado emp;

try {

fos = new FileOutputStream("empleados.dat");

salida = new ObjectOutputStream(fos);

emp = new Empleado("12345678A","Lucas González", 30, 1200.40);

salida.writeObject(emp);

emp = new Empleado("98765432B","Anacleto Jiménez", 28, 1000);

salida.writeObject(emp);

emp = new Empleado("78234212Z","María Zapata", 35, 1100.25);

salida.writeObject(emp);

} catch (FileNotFoundException e) {

System.out.println(e.getMessage());

} catch (IOException e) {

System.out.println(e.getMessage());

} finally {

try {

if(fos!=null) fos.close();

if(salida!=null) salida.close();

} catch (IOException e) {

System.out.println(e.getMessage());

}

}

}

}

**4. Serialización y clases compuestas**

Cuando una clase contiene un atributo que es una referencia a otro objeto, la clase a la que pertenece dicho atributo también debe ser serializable.

**Ejemplo 3** de serialización de una clase compuesta:

Programa java que escribe en un fichero tres objetos de tipo Alumno. Alumno es una clase derivada de Persona y contiene un atributo Fecha:

//Clase Fecha

import java.io.Serializable;

public class Fecha implements Serializable{

private int dia;

private int mes;

private int año;

public Fecha(int dia, int mes, int año) {

this.dia = dia;

this.mes = mes;

this.año = año;

}

public Fecha() {

}

public int getAño() {

return año;

}

public void setAño(int año) {

this.año = año;

}

public int getDia() {

return dia;

}

public void setDia(int dia) {

this.dia = dia;

}

public int getMes() {

return mes;

}

public void setMes(int mes) {

this.mes = mes;

}

@Override

public String toString(){

return " dia:"+dia+" mes:"+mes+" anio:"+año;

}

}

//Clase Alumno

public class Alumno extends Persona{

private Fecha fechaMatricula;

public Alumno(String nif, String nombre, int edad, Fecha fechaMatricula) {

super(nif, nombre, edad);

this.fechaMatricula = new Fecha();

setFechaMatricula(fechaMatricula);

}

public Alumno() {

}

public Fecha getFechaMatricula() {

return fechaMatricula;

}

public void setFechaMatricula(Fecha fechaMatricula) {

this.fechaMatricula.setDia(fechaMatricula.getDia());

this.fechaMatricula.setMes(fechaMatricula.getMes());

this.fechaMatricula.setAño(fechaMatricula.getAño());

}

@Override

public String toString(){

return super.toString()+"fecha matricula:"+fechaMatricula.toString();

}

}

//Clase Crear fichero alumnos

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectOutputStream;

public class CrearFalumnos {

public static void main(String[] args) {

FileOutputStream fos = null;

ObjectOutputStream salida = null;

Alumno a;

Fecha f;

try {

fos = new FileOutputStream("alumnos.dat");

salida = new ObjectOutputStream(fos);

f = new Fecha(5,9,2011);

a = new Alumno("12345678A","Lucas González", 20, f);

salida.writeObject(a);

f = new Fecha(7,9,2011);

a = new Alumno("98765432B","Anacleto Jiménez", 19, f);

salida.writeObject(a);

f = new Fecha(8,9,2011);

a = new Alumno("78234212Z","María Zapata", 21, f);

salida.writeObject(a);

} catch (FileNotFoundException e) {

System.out.println(e.getMessage());

} catch (IOException e) {

System.out.println(e.getMessage());

} finally {

try {

if(fos!=null) fos.close();

if(salida!=null) salida.close();

} catch (IOException e) {

System.out.println(e.getMessage());

}

}

}

}

**5. Leer objetos de ficheros**

Para leer los objetos contenidos en un fichero binario que han sido almacenados mediante ObjectOutputStream se utiliza la **clase ObjectInputStream** derivada de InputStream.

Un objeto ObjectInputStream se crea a partir de un objeto FileInputStream asociado al fichero.

El constructor de la clase es:

ObjectInputStream(InputStream nombre);

Lanza una excepción **IOException**.

**Por Ejemplo,**  las instrucciones para crear el objeto ObjectInputStream para lectura de objetos del fichero personas.dat serían éstas:

FileInputStream fis = new FileInputStream ("personas.dat");

ObjectInputStream entrada = new ObjectInputStream (fis);

La clase proporciona el **método readObject()** que devuelve el objeto del fichero (tipo Object).

Es necesario hacer un **casting** para guardarlo en una variable del tipo adecuado.

Lanza una **IOException**

**Ejemplo 4**  de lectura de objetos contenidos en un fichero en Java

Programa que lee los objetos del fichero creado en el Ejemplo 1:

//Clase lectura y visualización del fichero personas

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.ObjectInputStream;

public class LeerFpersonas {

public static void main(String[] args) {

FileInputStream fis = null;

ObjectInputStream entrada = null;

Persona p;

try {

fis = new FileInputStream("personas.dat");

entrada = new ObjectInputStream(fis);

p = (Persona) entrada.readObject(); //es necesario el casting

System.out.println(p.getNif() + " " + p.getNombre() + " " + p.getEdad());

p = (Persona) entrada.readObject();

System.out.println(p.getNif() + " " + p.getNombre() + " " + p.getEdad());

p = (Persona) entrada.readObject();

System.out.println(p.getNif() + " " + p.getNombre() + " " + p.getEdad());

} catch (FileNotFoundException e) {

System.out.println(e.getMessage());

} catch (ClassNotFoundException e) {

System.out.println(e.getMessage());

} catch (IOException e) {

System.out.println(e.getMessage());

} finally {

try {

if (fis != null) {

fis.close();

}

if (entrada != null) {

entrada.close();

}

} catch (IOException e) {

System.out.println(e.getMessage());

}

}

}

}

**Ejemplo 5:**

Programa Java que lee los objetos del fichero creado en el Ejemplo 2:

//Clase lectura y visualización del fichero empleados

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.ObjectInputStream;

public class LeerFempleados {

public static void main(String[] args) {

FileInputStream fis = null;

ObjectInputStream entrada = null;

Empleado emp;

try {

fis = new FileInputStream("empleados.dat");

entrada = new ObjectInputStream(fis);

emp = (Empleado) entrada.readObject();

System.out.println(emp.getNif() + " "

+ emp.getNombre() + " " + emp.getEdad() + " " + emp.getSueldo());

emp = (Empleado) entrada.readObject();

System.out.println(emp.getNif() + " "

+ emp.getNombre() + " " + emp.getEdad() + " " + emp.getSueldo());

emp = (Empleado) entrada.readObject();

System.out.println(emp.getNif() + " "

+ emp.getNombre() + " " + emp.getEdad() + " " + emp.getSueldo());

} catch (FileNotFoundException e) {

System.out.println(e.getMessage());

} catch (ClassNotFoundException e) {

System.out.println(e.getMessage());

} catch (IOException e) {

System.out.println(e.getMessage());

} finally {

try {

if (fis != null) {

fis.close();

}

if (entrada != null) {

entrada.close();

}

} catch (IOException e) {

System.out.println(e.getMessage());

}

}

}

}

**Ejemplo 6:**

Programa Java que lee los objetos del fichero creado en el Ejemplo 3:

//Clase lectura y visualización del fichero alumnos

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.ObjectInputStream;

import persona.Alumno;

public class LeerFalumnos {

public static void main(String[] args) {

FileInputStream fis = null;

ObjectInputStream entrada = null;

Alumno a;

try {

fis = new FileInputStream("alumnos.dat");

entrada = new ObjectInputStream(fis);

while(true){

a = (Alumno) entrada.readObject();//hay que hacer casting porque son Object

System.out.println(a.toString());

}

} catch (FileNotFoundException e) {

System.out.println(e.getMessage());

} catch (ClassNotFoundException e) {

System.out.println(e.getMessage());

} catch (IOException e) {

System.out.println(e.getMessage());

} finally {

try {

if (fis != null) {

fis.close();

}

if (entrada != null) {

entrada.close();

}

System.out.println("correcto");

} catch (IOException e) {

System.out.println(e.getMessage());

}

}

}

}