Ficheros binarios

Introducción

- Los ficheros binarios almacenan secuencias de dígitos binarios que no son legibles directamente por el usuario como ocurría con los ficheros de texto. Un ejemplo serían las imágenes.
- Como ya hemos visto previamente, en Java hay dos clases que permiten trabajar con ficheros: **FileInputStream** y **FileOutputStream.** Estas clases trabajan con flujos de bytes y crean un enlace entre el flujo de bytes y el fichero.
- Los métodos que proporciona la clase **FileInputStream** para lectura son similares a los vistos para la clase **FileReader.** Estos métodos devuelven número de bytes leídos o -1 si se ha llegado al final del fichero.

- Métodos de **FileInputStream:**
  - int read () → Lee un byte del flujo y lo devuelve
  - int read (byte[] b) → Lee hasta b.length bytes del flujo y los alberga en el array b
  - int read (byte [] b, int desplazamiento, int n) → Lee hasta n bytes comenzando en b*[desplazamiento]*
- Métodos **FileOutputStream:**
  - void write (int b) → escribe un byte
  - void write (byte [] b) → escribe b.length bytes
  - void write (byte [] b, int desplazamiento, int n) → escribe n bytes desde la matriz b comenzando por *b[desplazamiento]*

A continuación veremos un ejemplo en el que primero se escriben los bytes en un fichero y después se visualizan.

```
import java.io.* ;
public class EscribirFicheroBytes
{
	public static void main (String [] args) throws IOException
	{
		File fichero = new File (“FicheroDatos.dat”);
		FileOutputStream fileout = new FileOutputStream (fichero);
		FileInputStream filein = new FileInputStream(fichero);
		int i;
		for (i=1; i<100; i++)
			fileout.write(i);
		fileout.close();
		while ((i=filein.read()) != -1)
			System.out.println(i);
		filein.close();
	}
}
```

Bucle de escritura de bytes

Bucle de lectura

DataInputStream / DataOutputStream

- **DataInputStream y DataOutputStream** se utilizan para leer y escribir datos de tipos primitivos: int, float, long, etc.
- Estas clases definen diversos métodos readXXX y writeXXX que son variaciones de los métodos read() y write() de la clase base. Veamos algunos:

• boolean readBoolean(); • byte readByte(); • int readUnsignedByte(); • int readUnsignedShort(); • short readShort(); • char readChar(); • int readInt(); • float readFloat(); • String readUTF();	• void writeBoolean (boolean v); • void writeByte (int v); • void writeBytes (String s); • void writeShort (int v); • void writeChars (String s); • void writeChar (int v); • void writeInt (int v); • void writeFloat (float v); • void writeUTF (String str);

- Para abrir un objeto **DataInputStream** se utilizan los mismos métodos que para **FileInputStream**

```
File fichero = new File (“FicheroDatos.dat”);
FileInputStream filein= new FileInputStream (fichero);
DataInputStream dataIS = new DataInputStream (filein);
```

- Lógicamente, lo mismo sucede para **DataOutputStream:**

```
File fichero = new File (“FicheroDatos.dat”);
DataOutputStream dataOS = new DataOutputStream ( new FileOutputStream (fichero));
```

Problemas FicherosBinarios 1

1. Averigua jerárquicamente, ¿cuál es la relación entre **DataInputStream y FileInputStream**?
2. Escribe un programa que inserte datos en “FicherosDatos.dat”. Los datos los tomará de dos arrays definidos en el propio programa. Uno contendrá los nombres de una serie de personas y el otro sus edades. Se irá recorriendo los arrays e iremos escribiendo en el fichero el nombre (mediante el método **writeUTF(String str)** y la edad (**writeInt (int v))**. NOTA: si queremos añadir bytes al final del fichero (FicheroDatos.dat) se puede usar el siguiente constructor: *FileOutputStream fileout = new FileOutputStream (fichero, true)*

3. Ahora escribe un programa que permita visualizar los datos grabados anteriormente en el fichero FicheroDatos.dat. Se deben obtener en el mismo orden en el que se escribieron, es decir, primero obtenemos el nombre y luego la edad.
4. Vuelve a ejecutar los problemas anteriores pero en los pasos intermedios trata de leer el fichero “FicheroDatos.dat” con algún editor de texto del sistema. ¿Qué pasa?

Objetos en ficheros binarios

- Hemos visto cómo se guardan los tipos de datos primitivos en un fichero, pero, ¿qué pasa con los objetos definidos en el programa? Una opción sería ir guardando cada atributo por separado pero sería muy engorroso
- Solución: Java permite guardar objetos que implementen la interfaz **Serializable**.
- La interfaz **Serializable** impone una serie de métodos para guardar y leer objetos. Los más importantes son:
  - **Object readObject ()** → para leer objetos desde archivo.
    - Requiere un objeto del **ObjectInputStream**
    - Puede lanzar **IOException** y **ClassNotFoundException**
  - **void writeObject (Object obj)** → para escribir el objeto **obj** en archivo
    - Requiere un flujo **ObjectOutputStream**
    - Puede lanzar **IOException**

Ejemplo completo

- Primero definiremos la clase Persona. Contendrá dos atributos: nombre y persona. También de los métodos get y set correspondientes.

```
import java.io.Serializable;
public class Persona implements Serializable {
	private String nombre;
	private int edad;
	public Persona (String nombre, int edad) {
		this.nombre = nombre;
		this.edad = edad;
	}
	public Persona () {
		this.nombre=null;
	}
	public void setNombre (String nombre) {this.nombre = nombre;}
	public void setEdad (int edad) {this.edad = edad;}

	public String getNombre ( ) {return this.nombre;}
	public int getEdad ( ) {return this.edad;}
}
```

- A continuación, el siguiente ejemplo “escribe” un objeto Persona en un fichero

```
import.java.io.*;
public class EscribirFicheroObject {
        public static void main (String[] args) throws IOException {
                File fichero = new File (“FicheroPersona.dat”);
                FileOutputStream fileout = new FileOutputStream (fichero);
                ObjectOutputStream dataOS = new ObjectOutputStream (fileout);
                Persona persona = new Persona (“pepito”, 15);
                dataOS.writeObject (persona) ;
                dataOS.close();
        }
}
```

- Ahora veremos la lectura. El proceso de lectura se realiza en un bucle *while(true)* y éste se encierra en un bloque try-catch, ya que la lectura finalizará cuando llegue a final de fichero y se lance la excepción EOFException.

```
import java.io.*;
public class LeerFichObject {
	public static void main(String [] args) throws IOException, ClassNotFoundException
	{
		Persona persona;
		File fichero = new File (“FicheroPersona.dat”);
		FileInputStream filein = new FileInputStream (fichero);
		ObjectInputStream dataIS = new ObjectInputStream (filein) ;
		try {
			while (true) {
				persona = (Persona) dataIS.readObject ();
				System.out.printf(“Nombre: %s, edad : %d %n”,
					persona.getNombre(), persona.getEdad());
			}
		} catch (EOFException eo) {
			System.out.println (“Fin de Lectura.”);
		}
		dataIS.close();
	}
}
```

Problema FicherosBinarios 2

Voy a descomponer el problema en varias fases:

1. Diseña una clase que llamarás EstadoPartida para gestionar el estado de una partida de 3 en raya. Debe incluir: posición de las piezas (será una matriz 3x3) y a quién le toca tirar (jugador1 o jugador2).
2. Diseña una clase que llamarás GuardarYRecuperarEstado que disponga de dos métodos: guardarEstado que permitirá guardar en un archivo el estado de la partida y otro, recuperarEstado, que permitirá recuperar el estado de una partida desde archivo.
3. Crea un programa en java que genere un objeto EstadoPartida, modifique la posición de algunas fichas del tablero, guarde en un archivo el estado de la partida, recupere el estado desde el archivo y lo muestre por pantalla.