MANEJO DE FICHEROS FLUJOS O STREAMS

Javier García-Retamero Redondo

¿QUÉ SON?

- **Flujo o Stream:**
- Canal de comunicación que se establece para cualquier envío o recepción de información entre una fuente y un destino.

- Trabajaremos con dos tipos:
  - Flujos de 16 bits: Operaciones E/S de **caracteres** (Unicode).
    - Clases descendientes de **Reader** y **Writer**.
  - Flujos de 8 bits: Operaciones E/S de **bytes**.
    - Clases descendientes de **InputStream** y **OutputStream.**
    - **No son legibles directamente**
    - **Ocupan menos espacio en disco**

TRABAJANDO CON ARCHIVOS BINARIOS

¿QUÉ PODEMOS HACER?

- La clase **OutputStream** nos permite:
  - **Escribir** bytes en distintas fuentes: array de bytes, un fichero.
  - La subclase más utilizada es **FileOutputStream:** Escribir información en un fichero.
    - Ideal para escribir datos binarios simples, como imágenes, archivos binarios, o texto convertido a bytes.
    - No tiene métodos para escribir tipos de datos primitivos directamente (como int, boolean, double, etc.).

ESCRITURA PASOS

- **Abrimos el fichero:**

```
File ficheroPrueba = new File(".\\archivos_pruebas\\ejemploprueba.dat");
```

- **Creamos el flujo de escritura:**
  - **Opción 1:**

```
FileOutputStream ficheroOut = new FileOutputStream(ficheroPrueba);
```

Si el fichero no existe lo crea.
Si existe lo destruye y lo vuelve a crear.

  - **Opción 2:**

```
FileOutputStream ficheroOut = new FileOutputStream(ficheroPrueba, true);
```

Añadimos datos y no destruimos el fichero ya creado.

Escritura……

- **Cerramos el flujo:**

```
ficheroOut.close();
```

ESCRITURA BYTE A BYTE

- **Escribir un byte:**

```
int i=3;
ficheroOut.write(i);
```

- **Escribir un carácter:**

```
char c='A';
ficheroOut.write(c);
```

ESCRITURA ARRAY DE BYTES

- **Escribir una cadena:**

```
File ficheroPrueba = new File(".\\archivos_pruebas\\ejemploprueba.dat");
FileOutputStream ficheroOut = new FileOutputStream(ficheroPrueba);

String contenido = "Texto que queremos almacenar";
byte[] contenidoEnBytes = contenido.getBytes();  //Convertimos el String a array de bytes

ficheroOut.write(contenidoEnBytes);
ficheroOut.flush();
```

Libera el búffer de memoria

¿QUÉ PODEMOS HACER?

- La clase **InputStream** nos permite:
  - **Leer** bytes de distintas fuentes: array de bytes, un objeto String, un fichero.
  - La subclase más utilizada es **FileInputStream:** Lee información de un fichero.

  Los métodos que proporciona **FileInputStream** son similares a los que suministra la clase **FileReader** para trabajar con archivos de texto.

LECTURA PASOS

PASOS COMUNES

- **Creación de un objeto FileInputStream que apunte a un fichero existente:**

```
File ficheroPrueba = new File (".\\archivos_pruebas\\ejemploprueba.dat");
```

- **Creamos el flujo de lectura:**

```
FileInputStream ficheroIn = new FileInputStream(ficheroPrueba);
```

Lectura……

- **Cerramos el flujo:**

```
ficheroIn.close();
```

LECTURA BYTE A BYTE

- **Leemos el siguiente byte contenido en el fichero y lo devuelve:**

```
int i;
i= ficheroIn.read();
```

Devuelve -1 si es final de fichero

- **Lo mostramos:**

```
System.out.println((char) i); //Si es un carácter hacemos el casteo a char
```

LECTURA ARRAY DE BYTES

Si queremos leer varios bytes de golpe utilizaremos:

int read(byte[] cadena)

- Devuelve -1 si llega al final del fichero
- Devuelve en **cadena** los bytes leídos.

```
int i;
byte[] cadena = new byte(20);
i= ficheroIn.read(cadena);
```

Especificamos el tamaño de la cadena → los bytes a leer

LECTURA CARACTERES

Si queremos leer varios caracteres de golpe utilizaremos:

```
StringBuilder texto = new StringBuilder();
int i;
byte cadena[] = new byte[20];

while ((i= ficheroIn.read(cadena))!=-1) {
        texto.append(new String(cadena));
         cadena = new byte[10];
}
```

Especificamos el tamaño de la cadena → los bytes a leer

TRABAJANDO CON ARCHIVOS BINARIOS

DATOS PRIMITIVOS

ESCRITURA DATOS PRIMITIVOS

- Si queremos escribir datos primitivos, utilizaremos la clase **DataOutputStream**
- Los datos primitivos al escribirlos ocuparan el siguiente espacio:

TAMAÑO EN DISCO DE LOS DISTINTOS TIPOS DE DATOS PRIMITIVOS		
byte	ENTERO (-128 a 127)	1 BYTE
short	ENTERO (-32.768 a 32.767)	2 BYTES
int	ENTERO (-2.147.483.648 a 2.147.483.647)	4 BYTES
long	ENTERO (-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807)	8 BYTES
float	REAL	4 BYTES
double	REAL	8 BYTES
char	CARÁCTER	2 BYTES (writeChars) 1 BYTE (writeBytes)

ESCRITURA DATOS PRIMITIVOS

Métodos para escritura de datos primitivos (**DataOutputStream**):

MÉTODOS PARA ESCRITURA	
void writeBoolean(boolean v);	void writeInt(int v);
void writeByte(int v);	void writeLong(long v);
void writeBytes(String v); (cada carácter 1 byte)	void writeFloat(float v);
void writeShort(int v);	void writeDouble(double v);
void writeChars(String s);	void writeUTF(String str); (cada carácter 1 byte)
void writeChar(int v); (cada carácter **2** bytes)	

ESCRITURA DATOS PRIMITIVOS

- **Escribir una cadena:**

```
String contenido = “Texto que queremos almacenar";
File ficheroPrueba = new File(“.\\archivos_pruebas\\ejemploprueba.dat”);
FileOutputStream ficheroOut = new FileOutputStream(ficheroPrueba);
DataOutputStream datosOut = new DataOutputStream(ficheroOut);

datosOut.writeUTF(contenido);
ficheroOut.flush();

datosOut.close();
```

LECTURA DATOS PRIMITIVOS

Métodos para lectura de datos primitivos (**DataInputStream**):

MÉTODOS PARA LECTURA	
Boolean readBoolean();	int readInt();
Byte readByte(); (lee un byte)	long readLong();
int readUnsignedByte();	float readFloat();
int readUnsignedShort();	double readDouble()
short readShort();	string readUTF(); (lee un String)
char readChar(); (lee un carácter -2 bytes-)	

LECTURA DATOS PRIMITIVOS

- **Lectura mediante DataInputStream:**

```
String cadena;
File ficheroPrueba = new File("C:\\acceso_a_datos\\tema1\\ejemploprueba.dat");
FileInputStream ficheroIn = new FileInputStream(ficheroPrueba);
DataInputStream datosIn = new DataInputStream(ficheroIn);

cadena = datosIn.readUTF();
System.out.println(cadena);

datosIn.close();
```