Manejo de ficheros

Método	Descripción
getName()	Devuelve el nombre del fichero o directorio
getPath(), getAbsolutePath()	Devuelve camino relativo o absoluto del fichero o directorio
getParent()	Devuelve el nombre del directorio padre o null si no existe
canRead()	Devuelve true si el fichero se puede leer
canWrite()	Devuelve true si el fichero se puede escribir
length()	Devuelve el tamaño del fichero en bytes
createNewFile()	Crea un nuevo fichero vacío, asociado a file si existe un fichero con dicho nombre
delete()	Borra el fichero o directorio
exists()	Devuelve true si el fichero o directorio existe
isDirectory()	Devuelve true si el objeto file corresponde a un directorio
isFile()	Devuelve true si el objeto file corresponde a un fichero
renameTo(File nuevonombre)	Renombra el fichero
mkdir()	Crea un directorio con el nombre indicado en la creación del objeto file

InputStream	ByteArrayInputStream	Permite usar un espacio de almacenamiento intermedio de memoria
	StringBufferInputStream	Convierte un String en un InputStream
	FileInputStream	Para leer información de un fichero
	PipedInputStream	Implementa el concepto de "tubería"
	FilterInputStream	Funcionalidad útil a otras clases InputStream
	SequenceInputStream	Convierte dos o más objetos InputStream en un InputStream único
OutputStream	ByteArrayOutputStream	Crea un espacio de almacenamiento intermedio en memoria. Todos los datos que se envían al flujo se ubican en este espacio.
	FileOutputStream	Para enviar información a un fichero
	PipedOutputStream	Implementa el concepto de "tuberia". Cualquier información que se desee escribir aquí será la entrada del PipedInputStream asociado.
	FilterOutputStream	Funcionalidad útil a otras clases OutputStream

Clases de flujos de bytes	Clase correspondiente de flujo de caracteres	
InputStream	Reader => InputStreamReader	
OutputStream	Writer => OutputStreamreader	
FileInputStream	***FileReader***	**acceso secuencial*
FileOutputStream	***FileWriter***	**acceso secuencial*
StringBufferInputStream	StringReader	
(no tiene equivalente)	StringWriter	
ByteArrayInputStream	CharArrayReader	
ByteArrayOutputStream	CharArrayWriter	
PipedOutputStream	PipedReader	
PipedOutputStream	PipedWriter	

Clases para la gestión de flujos de datos (desde/hacia ficheros).

1. **Ficheros de texto.**

Clases *FileReader y FileWriter*

Siempre hay que hacerlo dentro de un bloque **try-catch** (posibles excepciones: FileNotFoundException, IOException)

Métodos de FileReader:

devuelven el nº de caracteres leídos o -1 si llega al final

- ***int read()*** lee un carácter y lo devuelve
- ***int read(char[] buf)*** lee hasta buf.lenght caracteres de una matriz de caracteres (buf). Los caracteres leídos se almacenan en buf.
- ***int read(char[] buf, int desplazamiento, int n)*** lee hasta n caracteres de datos de la matriz buf comenzando por buf[desplazamiento].

Métodos de FileWriter:

- ***void write(int c)*** escribe un carácter
- ***void write(char[] buf)*** escribe un array de caracteres
- ***void write(char[] buf, int desplazamiento, int n)*** escribe n caracteres en la matriz buf comenzando por buf[desplazamiento]
- ***void write(String str)*** escribe una cadena de caracteres
- ***append (char c)*** añade un carácter a un fichero

Clases para la gestión de flujos de datos (desde/hacia ficheros).

2. **Ficheros binarios.**

Almacenan secuencias de dígitos binarios que no son legibles directamente por el usuario. Tienen la ventaja de que ocupan menos espacio en disco.

Clases *FileInputStream y FileOutputStream*

Métodos de FileInputStream:

devuelven el nº de bytes leídos o -1 si llega al final

- ***int read()*** lee un byte y lo devuelve
- ***int read(byte[] b)*** lee hasta b.lenght bytes de una matriz de bytes.
- ***int read(byte[] b, int desplazamiento, int n)*** lee hasta n bytes de la matriz b comenzando por b[desplazamiento].

Métodos de FileOutputStream:

- ***void write(int b)*** escribe un byte
- ***void write(byte[] b***) escribe b.lenght bytes
- ***void write(byte[] b, int desplazamiento, int n***) escribe n bytes a partir de la matriz de bytes de entrada y comenzando por b[desplazamiento]

- Al igual que con los ficheros de caracteres, es posible añadir bytes al final de un fichero, colocando el segundo parámetro de **FileOutputStream** a *true*.

Las **clase DataInputStream y DataOutputStream** permiten leer y escribir datos primitivos (int, float, long…), de modo independiente en cada máquina.

Los métodos para abrir un objeto DataInputStream o DataOutputStream son los mismos que para FileInputSTream y FileOutputStream. Y además de read() y write() proporcionan:

Métodos lectura	Métodos escritura
boolean readBoolean()	void writeBoolean(boolean v)
byte readByte()	void writeByte(int v)
int readUnisgnedByte()	void writeBytes(String s)
int readUnsignedShort()	void writeShort(int v)
short readShort()	void writeChars(String s)
char readChar()	void writeChar(int v)
int readInt()	void writeInt(int v)
long readLong()	void writeLong(long v)
float readFloat()	void writeFloat(float v)
double readDouble()	void writeDouble(double v)
String readUTF()	void writeUTF(String str)

fichero (al crearse es 0, y cada read()-write() modifica el valor). *Es importante saber cuánto ocupa cada tipo de datos, para un correcto manejo del puntero.*

Métodos de RandomAccessFile:

- ***long getFilePointer():*** posición actual del puntero del fichero
- ***void seek(long posicion):*** coloca el puntero del fichero en una posición determinada desde el comienzo.
- **long length():** devuelve el tamaño del fichero en bytes. La posición length() marcará el final del fichero.
- **int skipBytes(int desplazamiento):** desplaza ealpuntero desde la posición actual el nº de bytes indicados en desplazamiento.

Tipo	Tamaño
Byte	8 bits
Short	16 bits
Int	32 bits
Long	64 bits
Float	32 bits
Double	64 bits
char	16 bits
boolean	1 byte

Manejo de Ficheros

Ejemplo: Inserta datos de alumnos en un fichero de acceso aleatorio.

```
import java.io.*;

public class ficheroaleatorio {
    public static void main(String[] args) throws IOException {
                File f= new File("C:/Users/Trabajo/Documents/borrar6.dat");
                RandomAccessFile alea=new RandomAccessFile(f,"rw");
                String nombre[]={"David","Eduardo","Joseph","Miguel","Luis"};
                int edad[]={19,20,18,21,21};

                StringBuffer buf=null;

                        /*Un objeto String representa una cadena alfanumérica de un valor constante
                         * que no puede ser cambiada después de haber sido creada.
                         * Un objeto StringBuffer representa una cadena cuyo tamaño puede variar
                         * */

                for (int i=0; i<nombre.length; i++){
                alea.writeInt(i+1);// se pone un identificador
                buf=new StringBuffer(nombre[i]);
                buf.setLength(10);  // establece un tamaño fijo para el nombre
                alea.writeChars(buf.toString());
                alea.write(edad[i]);
                }
                alea.close();
    }
}
```

Para recorrer el fichero, calculamos tamaño del registro: 4 + 20 + 4 (28 bytes)

Y se utilizará una variable para recorrerlo en bytes.

```
import java.io.*;
public class LeeFicheroAleatorio {
    public static void main(String[] args) throws IOException {
                File f= new File("C:/Users/Trabajo/Documents/borrar6.dat");
                RandomAccessFile alea=new RandomAccessFile(f,"r");
                int id, vedad, pos=0;
                char cnombre[]= new char[10], aux;

                try{
                        while (true){
                                alea.seek(pos);
                                id=alea.readInt();
                                for (int i=0; i<cnombre.length; i++) {
                                        aux=alea.readChar();
                                        cnombre[i]=aux;
                                }
                                String vnombre=new String(cnombre);
                                vedad=alea.readInt();
                                System.out.println("Id: "+id+" Nombre: "+vnombre+" Edad: "+vedad);
                                pos=pos+28;                     //se avanza según tamaño del registro
                        }
                } catch (EOFException e) {}
                alea.close();
}}
```

```
FileInputStream fi = new FileInputStream(fl);
DataInputStream dis = new DataInputStream(fi);

ArrayList<ObjEquipos> equipos = new ArrayList<>();
Gson gson = new Gson();
String json;

while (dis.available() > 0) {
    int numero200 = dis.readInt();
    String cPresident = dis.readUTF();
    String cNomClub = dis.readUTF();
    String cTelClub = dis.readUTF();
    String cLocal = dis.readUTF();

    ObjEquipos equipo = new ObjEquipos(numero200, cPresident, cNomClub, cTelClub, cLocal);
    equipos.add(equipo);
    json = gson.toJson(equipos);
    try {
        File f = new File( pathname: "./src/Equipos.json");
        FileWriter fw = new FileWriter(f);
        fw.write(json);
        fw.flush();

    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    // Filtrar los números de club entre 200 y 300
    if (numero200 > 200 && numero200 < 300) {
        System.out.println(numero200 + "," + cPresident + "," + cNomClub + "," + cTelClub + "," + cLocal);
    }
}
```

Remplazar espacios

```
nile((linea = bf.readLine() )!=null) {
   String lineaSinEspacios = linea.replaceAll("\\s","");
```

Tipo de fichero		Clases asociadas	Métodos	Ejemplo sintaxis
Todos (independiente de modo acceso)		File	getName() getPath()	File fichero=new File("disco\\nomfich");
Ficheros secuenciales	**Ficheros de texto**	FileReader	read()	FileReader fichin=new FileReader(fichero); while ((int c=fichin.read())!= -1){ System.out.print((char) c);}
		FileWriter	write()	FileWriter fichout = new FileWriter(fichero); for(int i=texto.length()-1;i>=0; i--) { fichout.write(texto.charAt(i));
		BufferedReader	readLine()	BufferedReader bufin =new BufferedReader(new FileReader(fichero)); while(bufin.readLine() != null) {

		BufferedWriter	write()
			newline()
				BufferedWriter bufout =new BufferedWriter(new FileWriter(fich_salida));
				bufout.write(texto);
				bufout.newLine();

		PrintWriter	print() println()	PrintWriter printWriter = new PrintWriter(fichero); printWriter.println ("ejemplo");
	Ficheros binarios	FileInputStream	read()	FileInputStream fi= new FileInputStream(fichero); while ((int val=fi.read())!= -1){ System.out.print((char)val);}

		FileOutputStream	write()	FileOutputStream fo= new FileOutputStream(fichero);
				byte codigos[]=texto.getBytes();
				fo.write(codigos);

		DataInputStream	readInt() readChar() readUTF()	DataInputStream dis=new DataInputStream(new FileInputStream(fichero); System.out.println(dis.readInt());
		DataOutputStream	writeInt() writeChar() writeUTF()	DataOutputStream dos=new DataOutputStream(new FileOutputStream(fichero); dos.writeInt(23);
		ObjectInputStream	readObject()	ObjectInputStream ent =new ObjectInputStream(new FileInputStream(fichero); System.out.println((String) ent.readObject());
		ObjectOutputStream	writeObject()	ObjectOutputStream sal=new ObjectOutputStream(new FileOutputStream(fichero)); sal.writeObject(persona);

Ficheros de Acceso Aleatorio		RandomAccessFile	seek()
			length()
			getFilePointer()
				Utiliza métodos read() y write() de DataInputStream y DataOutputStream
				RandomAccessFile raf = new RandomAccessFile(fichero, "r");
				raf.writeInt(10);
				raf.seek(pos);
				raf.readInt;