**Tema11.FICHEROS**

**Ficheros Binarios en Java**

Un fichero binario o de datos está formado por secuencias de bytes 8 ( bits). Estos archivos pueden contener datos de tipo básico (int, float, char, etc) y objetos.

**Para poder leer el contenido de un fichero binario** debemos conocer la estructura interna del fichero, es decir, **debemos saber cómo se han escrito**: si hay enteros, long, etc. y en qué orden están escritos en el fichero. Si no se conoce su estructura podemos leerlo byte a byte

**ESCRIBIR DATOS EN FICHEROS BINARIOS**

Para escribir datos en un fichero binario utilizaremos las clases Java derivadas de OutputStream.

**FileOutputStream y DataOutputStream**

**FileOutputStream**

La clase FileOutputStream permite tener acceso al fichero para **escribir bytes**.

Para crear objetos FileOutputStream podemos utilizar los constructores:

FileOutputStream (String ruta)

FileOutputStream (File objetoFile);

FileOutputStream (String ruta, boolean append)

FileOutputStream (File objetoFile, boolean append)

Si el parámetro append es true significa que los datos se van a añadir a los existentes.

Si es false los datos existentes se pierden. Si se utiliza uno de los dos primeros constructores los datos existentes se pierden.

Los constructores lanzan una **excepción FileNotFoundException**si no existe y no se ha podido crear el fichero.

La clase FileOutputStream proporciona el **método write()** **para escribir bytes** en el fichero. Este método lanza una **IOException**.

**DataOutputStream**

A partir de un objeto FileOutputStream se puede crear un objeto DataOutputStream, que proporciona métodos para **escribir datos de tipo primitivo** en el fichero.

Para crear un objeto DataOutputStream se utiliza el constructor:

DataOutputStream(OutputStream nombre);

La clase proporciona **métodos writeXxx()** donde Xxx es el nombre del tipo primitivo. Lanzan una**IOException**

Siempre se utilizan conjuntamente.

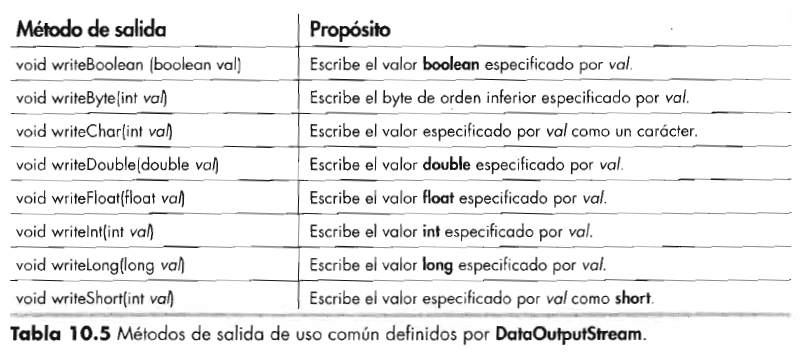
FileOutputStream fos = new FileOutputStream("datos.dat");

DataOutputStream salida = new DataOutputStream(fos);

También se utiliza la clase BufferedOutputStream que almacena el flujo de salida en un buffer, lo mismo con el de entrada, en lugar de DataOutputStream

BufferedOutputStream salida=new BufferedOutputStream(fos)

ejemplo en carpeta binariobuffered



**Ejemplo 1:**

Programa que lee enteros por teclado y los escribe en el fichero datos.dat. La lectura de datos acaba cuando se introduce -1.

import java.io.DataOutputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Scanner;

public class Binarios1 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

FileOutputStream fos = null;

DataOutputStream salida = null;

int n;

try {

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

salida = new DataOutputStream(fos);

System.out.print("Introduce número entero. -1 para acabar: ");

n = sc.nextInt();

while (n != -1) {

salida.writeInt(n);//se escribe el número entero en el fichero

System.out.print("Introduce número entero. -1 para acabar: ");

n = sc.nextInt();

}

} 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());

}

}

}

}

**Ejemplo 2:**

Programa Java que crea una matriz de elementos de tipo double y lee por teclado el valor de sus elementos. A continuación escribe el contenido de la matriz en un fichero. Al principio del fichero se escriben dos enteros con los valores del número de filas y columnas de la matriz.

import java.io.DataOutputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Scanner;

public class Binarios2 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        FileOutputStream fos = null;

        DataOutputStream salida = null;

        double[][] matriz;

        int filas, columnas, i, j;

        do{

            System.out.print("Número de filas: ");

            filas = sc.nextInt();

        }while(filas<=0);

        do{

            System.out.print("Número de columnas: ");

            columnas = sc.nextInt();

        }while(columnas<=0);

        matriz = new double[filas][columnas]; //se crea la matriz

        for (i = 0; i < filas; i++) {   //lectura de datos por teclado

            for (j = 0; j < columnas; j++) {

                System.out.print("matriz[" + i + "][" + j + "]: ");

                matriz[i][j] = sc.nextDouble();

            }

        }

        try {

            //crear el fichero de salida

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

            salida = new DataOutputStream(fos);

            //escribir el número de filas y columnas en el fichero

            salida.writeInt(filas);

            salida.writeInt(columnas);

           //escribir la matriz en el fichero

            for (i = 0; i < filas; i++) {

                for (j = 0; j < columnas; j++) {

                    salida.writeDouble(matriz[i][j]);

                }

            }

        } 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());

            }

        }

    }

}

**LECTURA DE FICHEROS BINARIOS**

Para leer de un fichero binario utilizaremos las clases Java **FileInputStream y DataInputStream** derivadas de InputStream.

**FileInputStream**

La clase FileInputStream permite **leer bytes** de un fichero.

Para crear objetos FileInputStream podemos utilizar los constructores:

FileInputStream (String ruta)

FileInputStream (File objetoFile);

Ambos lanzan una **excepción FileNotFoundException** si el fichero no existe.

La clase proporciona el **método read()** **para leer bytes** del fichero.

El método read lanza una excepción **IOException**.

**DataInputStream**

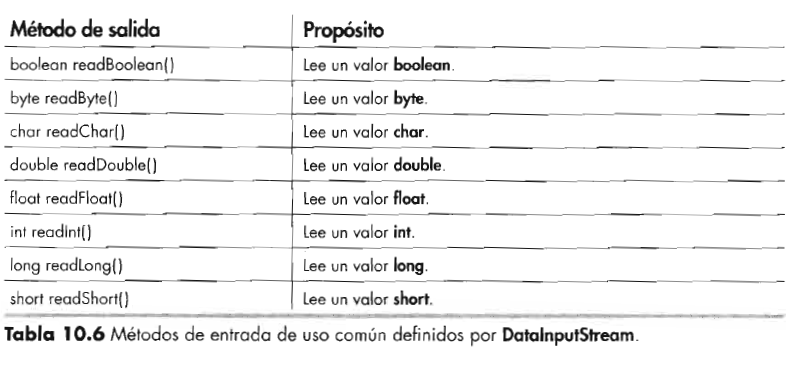
A partir de un objeto FileInputStream podemos crear un objeto DataInputStream para **leer datos de tipo primitivo**.

Para crear un objeto DataInputStream se utiliza el constructor:

DataInputStream (InputStream nombre);

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

DataInputStream entrada = new DataInputStream(fis);



La clase proporciona **métodos readXxx()** donde Xxx es el nombre del tipo primitivo. Lanzan una excepción**IOException**.

Cuando un método readXxx() alcanza el final del fichero lanza una **excepción EOFException**.

**método int available() Devuelve el número de bytes disponibles para leer en la entrada. Entonces mientras el valor sea >0 no se ha terminado el fichero.**

**Ejemplo 3:**

Programa que lee el contenido del fichero creado en el Ejemplo 1. Utilizaremos un bucle infinito para leer los datos. Cuando se llega al final del fichero se lanza la excepción EOFException que se utiliza para salir del bucle while.

import java.io.DataInputStream;

import java.io.EOFException;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

public class Binarios3 {

    public static void main(String[] args) {

        FileInputStream fis = null;

        DataInputStream entrada = null;

        int n;

        try {

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

            entrada = new DataInputStream(fis);

// while(entrada.available()>0){

            while (true) {

                n = entrada.readInt();  //se lee  un entero del fichero

                System.out.println(n);  //se muestra en pantalla

            }

        } catch (FileNotFoundException e) {

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

        } catch (EOFException e) {

            System.out.println("Fin de fichero");

        } 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 4:**

Programa Java que lee el contenido del fichero creado en el Ejemplo 2 y lo muestra por pantalla.

import java.io.DataInputStream;

import java.io.EOFException;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

public class Binarios4 {

    public static void main(String[] args) {

        FileInputStream fis = null;

        DataInputStream entrada = null;

        int filas, columnas, i, j;

        try {

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

            entrada = new DataInputStream(fis);

            filas = entrada.readInt();  //se lee el primer entero del fichero

            columnas = entrada.readInt();//se lee el segundo entero del fichero

            for (i = 0; i < filas; i++) {

                for (j = 0; j < columnas; j++) {  // se leen los double y se muestran por pantalla

                    System.out.printf("%8.2f", entrada.readDouble());

                }

                System.out.println();

            }

        } catch (FileNotFoundException e) {

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

        } catch (EOFException e) {

            System.out.println("Fin de fichero");

        } 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());

            }

        }

    }

}