



TEMA 9

Ficheros

V1.0

Manuel Pereira González

Agenda



- **Introducción**
 - **Conceptos Básicos de Ficheros**
 - Operaciones sobre ficheros
 - Tipos de fichero
- Ficheros en Java
 - Conceptos Básicos de Entrada/Salida
 - Ficheros Binarios
 - Ficheros de Tipos Primitivos
 - Ficheros de Texto
 - Ficheros de Acceso Aleatorio
 - Serialización
- Algoritmos sobre Ficheros
- Resumen



Introducción: Conceptos Básicos de Ficheros



- Estructuras de Datos estudiadas:
 - Tipos: Arrays, Listas, etc.
 - Almacenadas en **memoria principal** (RAM)
 - Rápida
 - Volátil
 - Tamaño Limitado
- Para tratar grandes volúmenes de información es necesario almacenarla en **memoria secundaria** (Disco Duro, CD-ROM, Disquete, Disco USB, etc.)
 - Los datos agrupados en estructuras denominadas **archivos o ficheros** (File en inglés)

Introducción: Conceptos Básicos de Ficheros



- Un **archivo o fichero** es una colección de datos homogéneos almacenados en un soporte físico del computador que puede ser permanente o volátil.
 - Datos **homogéneos**: Almacena colecciones de datos del mismo tipo (igual que arrays/vectores)
 - Cada elemento almacenado en un fichero se denomina **registro**, que se compone de **campos**.
 - Puede ser almacenado en **diversos soportes** (Disco duro, disquete, ...)

Introducción: Conceptos Básicos de Ficheros



- Ejemplos de archivos o ficheros:

Archivo	Registro	Campos
Biblioteca	Libro	Título, Autor, ISBN, ...
Censo	Persona	Nombre, Apellidos, DNI, Dirección, ...
Archivo de coches	Coche	Marca, Color, ...

- El tamaño de un fichero es variable puede crecer según se van insertando registros

Agenda



- Introducción
 - Conceptos Básicos de Ficheros
 - **Operaciones sobre ficheros**
 - Tipos de fichero
- Ficheros en Java
 - Conceptos Básicos de Entrada/Salida
 - Ficheros Binarios
 - Ficheros de Tipos Primitivos
 - Ficheros de Texto
 - Ficheros de Acceso Aleatorio
 - Serialización
- Algoritmos sobre Ficheros
- Resumen



Introducción: Operaciones sobre Ficheros



- Tipos de operaciones
 - Operación de **Creación**
 - Operación de **Apertura**. Varios modos:
 - Sólo lectura
 - Sólo escritura
 - Lectura y Escritura
 - Operaciones de **lectura / escritura**
 - Operaciones de **inserción / borrado**
 - Operaciones de **renombrado / eliminación**
 - Operación de **desplazamiento** dentro de un fichero
 - Operación de **cierre**

Introducción: Operaciones sobre Ficheros



- Operaciones para el manejo habitual de un fichero:
 - 1.- **Crearlo** (sólo si no existía previamente)
 - 2.- **Abrirlo**
 - 3.- **Operar** sobre él (lectura/escritura, inserción, borrado, etc.)
 - 4.- **Cerrarlo**

Agenda



- Introducción
 - Conceptos Básicos de Ficheros
 - Operaciones sobre ficheros
 - **Tipos de fichero**
- Ficheros en Java
 - Conceptos Básicos de Entrada/Salida
 - Ficheros Binarios
 - Ficheros de Tipos Primitivos
 - Ficheros de Texto
 - Ficheros de Acceso Aleatorio
 - Serialización
- Algoritmos sobre Ficheros
- Resumen



Introducción: Tipos de Fichero



- Clasificación de los ficheros según la organización de los registros en memoria:
 - **Organización Secuencial:** Registros almacenados consecutivamente en memoria según el orden lógico en que se han ido insertando.
 - **Organización Directa o Aleatoria:** El orden físico de almacenamiento en memoria puede no coincidir con el orden en que han sido insertados.
 - **Organización Indexada.** Dos ficheros:
 - Fichero de datos: Información
 - Fichero de índice: Contiene la posición de cada uno de los registros en el fichero de datos

Introducción: Tipos de Fichero



- Clasificación de los ficheros según el acceso a la información almacenada:
 - **Acceso secuencial:** Para acceder a un registro es necesario pasar por todos los anteriores. Ej: Cinta de Casete
 - **Acceso directo o aleatorio:** Se puede acceder a un registro sin pasar por todos los anteriores. Ej: Disco Duro.
- Clasificación de los ficheros según el tipo de la información almacenada:
 - Ficheros **Binarios:** Almacenan secuencias de dígitos binarios (ej: ficheros que almacenan enteros, floats,...)
 - Ficheros de **Texto:** Almacenan caracteres alfanuméricos en un formato estándar (ASCII, Unicode, UTF8, UTF16, etc.). Pueden ser leídos y/o modificados por aplicaciones denominadas editores de texto (Ej: Notepad, UltraEdit, Editplus, etc.).

Agenda



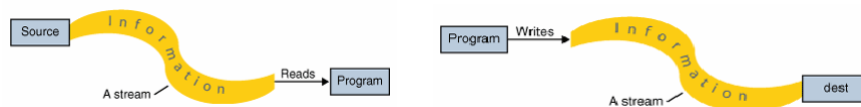
- Introducción
 - Conceptos Básicos de Ficheros
 - Operaciones sobre ficheros
 - Tipos de fichero
- **Ficheros en Java**
 - **Conceptos Básicos de Entrada/Salida**
 - Ficheros Binarios
 - Ficheros de Tipos Primitivos
 - Ficheros de Texto
 - Ficheros de Acceso Aleatorio
 - Serialización
- Algoritmos sobre Ficheros
- Resumen



Conceptos básicos de Entrada/Salida



- **Streams:** Canales, flujos de datos o "tuberías". Entrada (InputStream) o Salida (OutputStream).

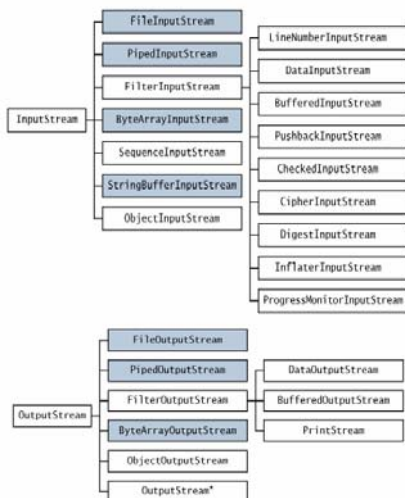


- Agrupados en el paquete **java.io**
- Dos jerarquías de clases independientes, una para lectura/escritura binaria (**bytes**) y otra para lectura/escritura de caracteres de texto (**char**)

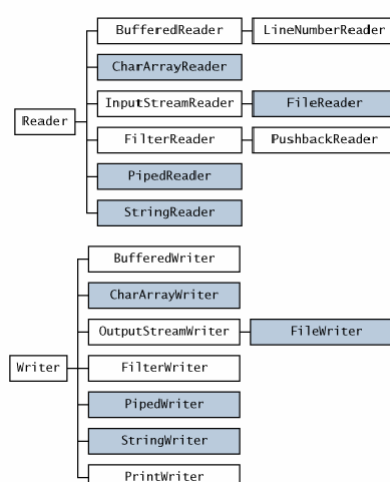
Conceptos básicos de Entrada/Salida



- Streams de bytes



- Streams de caracteres



Conceptos básicos de Entrada/Salida



- Métodos básicos de **Reader**:
 - `int read()`
 - `int read(char cbuf[])`
 - `int read(char cbuf[], int offset, int length)`
- Métodos básicos de **InputStream**:
 - `int read()`
 - `int read(byte cbuf[])`
 - `int read(byte cbuf[], int offset, int length)`

Conceptos básicos de Entrada/Salida



- Métodos básicos de **Writer**:
 - `int write(int c)`
 - `int write(char cbuf[])`
 - `int write(char cbuf[], int offset, int length)`
- Métodos básicos de **OutputStream**:
 - `int write(int c)`
 - `int write(byte cbuf[])`
 - `int write(byte cbuf[], int offset, int length)`
- Los Streams se abren automáticamente al crearlos, pero es necesario cerrarlos explícitamente llamando al método **close()** cuando se dejan de usar

Conceptos básicos de Entrada/Salida



- **PrintStream / PrintWriter** se utilizan para escribir cadenas de texto.
- **DataInputStream / DataOutputStream** se utilizan para escribir/leer tipos básicos (int, long, float,...).
- Acceso a ficheros: Según el acceso:
 - Acceso **Secuencial**: El más común. Puede ser:
 - Acceso binario: **FileInputStream / FileOutputStream**
 - Acceso a caracteres (texto): **FileReader / FileWriter**
 - Acceso **Aleatorio**: Se utiliza la clase **RandomAccessFile**

Conceptos básicos de Entrada/Salida



- La clase **File** puede usarse para representar el nombre de un **archivo concreto**, o los nombres de los archivos de un **directorio**.
- Para crear/abrir un fichero en Java se invoca a un constructor de la clase **File**.

```
import java.io.File;

public class CreaFile {
    public static void main(String[] args) {
        // Crea/Abre un fichero dado el nombre completo
        File f1 = new File("C:\\practicas\\ejemplo.txt");
        // Crea/Abre un fichero dado el directorio y el nombre
        File f2 = new File("C:\\practicas", "ejemplo.txt");
        // Crea/Abre un fichero dado un File con el directorio y el nombre
        File dir = new File("C:\\practicas");
        File f3 = new File(dir, "ejemplo.txt");
    }
}
```

Conceptos básicos de Entrada/Salida



```
import java.io.File;

/**
 * Dado un fichero indicado por línea de comandos, muestra si se trata de
 * un fichero, un directorio o de ninguno de los dos (no existe)
 */
public class MuestraFichero {

    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Debe indicar el fichero/directorio a analizar");
            System.exit(-1);
        }
        File f = null;
        try {
            f = new File(args[0]);
            if (f.isFile()) {
                System.out.println("El archivo: " + args[0] + " es un fichero");
            } else if (f.isDirectory()) {
                System.out.println("El archivo: " + args[0] + " es un directorio");
            } else {
                System.out.println("El archivo: " + args[0] + " no existe");
            }
        } catch (Exception e) {
            System.out.println("Error abriendo fichero: " + e.getMessage());
        }
    }
}
```

Conceptos básicos de Entrada/Salida



```
C:\Practicas\Programacion>java MuestraFichero
Debe indicar el fichero/directorio a analizar
C:\Practicas\Programacion>java MuestraFichero c:
El archivo: c: es un directorio
C:\Practicas\Programacion>java MuestraFichero c:\practica
El archivo: c:\practica no existe
C:\Practicas\Programacion>java MuestraFichero c:\\practicas
El archivo: c:\\practicas es un directorio
C:\Practicas\Programacion>java MuestraFichero c:\\practicas\\programacion\\MuestraFichero.java
El archivo: c:\\practicas\\programacion\\MuestraFichero.java es un fichero
C:\Practicas\Programacion>
```

Agenda



- Introducción
 - Conceptos Básicos de Ficheros
 - Operaciones sobre ficheros
 - Tipos de fichero
- Ficheros en Java
 - Conceptos Básicos de Entrada/Salida
 - **Ficheros Binarios**
 - Ficheros de Tipos Primitivos
 - Ficheros de Texto
 - Ficheros de Acceso Aleatorio
 - Serialización
- Algoritmos sobre Ficheros
- Resumen



Ficheros Binarios



- Leer/Escribir **bytes** en ficheros
- Streams de Entrada/Salida:
FileInputStream /
FileOutputStream

```
Simbolo del sistema
C:\Practicas\Programacion>java EscribeBytes
Escribiendo en fichero...
Fichero escrito
C:\Practicas\Programacion>
```

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

public class EscribeBytes {
    public static void main(String[] args) throws IOException {
        String ruta = "c:/practicas/programacion";
        // Declara el fichero
        File fichero = new File(ruta, "datos.dat");
        // Crea el flujo de salida hacia el fichero
        FileOutputStream flujoSalida = new FileOutputStream(fichero);
        // Información a almacenar
        byte[] datos1 = new byte[100];
        byte[] datos2 = new byte[100];
        // Escribe datos1 byte a byte
        System.out.println("Escribiendo en fichero...");
        for (int i=0; i<datos1.length; i++) {
            datos1[i] = (byte) i;
            datos2[i] = (byte) i;
            flujoSalida.write(datos1[i]);
        }
        // Escribe datos2 de una sola vez
        flujoSalida.write(datos2);
        // Cierra el Stream, y por tanto el fichero
        flujoSalida.close();
        System.out.println("Fichero escrito");
    }
}
```

Ficheros Binarios



```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

public class LeeBytes {
    public static void main (String[] args) throws IOException {
        String ruta = "c:/practicass/programacion/";
        // Declara el fichero
        File fichero = new File(ruta, "datos.dat");
        // Crea el flujo de entrada desde fichero
        FileInputStream flujoEntrada = new FileInputStream(fichero);
        // Calcula el tamaño del fichero
        int tamañoFichero = (int) fichero.length();
        byte[] datosLeídos = new byte[tamañoFichero];
        // Lee los datos
        System.out.println("Leyendo fichero...");
        flujoEntrada.read(datosLeídos);
        // Muestra los datos leídos por pantalla
        for(int i=0; i<datosLeídos.length; i++) {
            System.out.print(datosLeídos[i] + " ");
        }
        System.out.println();
        // Cierra el Stream, y por tanto el fichero
        flujoEntrada.close();
        System.out.println("Fichero leído");
    }
}
```

```
C:\Practicass\Programacion>java LeeBytes
Leyendo fichero...
0 1 2 3 4 5 6 7 8 9 10 11 12
13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40 41 42
43 44 45 46 47 48 49 50 51 52
53 54 55 56 57 58 59 60 61 62
63 64 65 66 67 68 69 70 71 72
73 74 75 76 77 78 79 80 81 82
83 84 85 86 87 88 89 90 91 92
93 94 95 96 97 98 99 0 1 2 3
4 5 6 7 8 9 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44
45 46 47 48 49 50 51 52 53 54
55 56 57 58 59 60 61 62 63 64
65 66 67 68 69 70 71 72 73 74
75 76 77 78 79 80 81 82 83 84
85 86 87 88 89 90 91 92 93 94
95 96 97 98 99
Fichero leído
C:\Practicass\Programacion>
```

Agenda



- Introducción
 - Conceptos Básicos de Ficheros
 - Operaciones sobre ficheros
 - Tipos de fichero
- Ficheros en Java
 - Conceptos Básicos de Entrada/Salida
 - Ficheros Binarios
 - **Ficheros de Tipos Primitivos**
 - Ficheros de Texto
 - Ficheros de Acceso Aleatorio
 - Serialización
- Algoritmos sobre Ficheros
- Resumen



Ficheros de Tipos Primitivos



- Leer/Escribir datos de tipos primitivos: int, long, float, etc.
- Streams de Entrada/Salida: **DataInputStream / DataOutputStream**

```
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

public class EscribeDatosPrimitivos {
    public static void main(String[] args) throws IOException {
        String ruta = "c:/practicas/programacion";
        // Declara el fichero
        File fichero = new File(ruta, "datos2.dat");
        // Crea el flujo de salida hacia el fichero
        FileOutputStream flujoSalida = new FileOutputStream(fichero);
        // Conecta el flujo de bytes al flujo de datos
        DataOutputStream dataOS = new DataOutputStream(flujoSalida);

        // Escribe algunos datos primitivos al fichero
        System.out.println("Escribiendo en fichero...");
        dataOS.writeBoolean(true);
        dataOS.writeChar('M');
        dataOS.writeDouble(222222222D);
        dataOS.writeInt(2006);
        dataOS.writeFloat(9999999F);
        dataOS.writeLong(42398432L);
        dataOS.writeUTF("Hola Mundo");

        // Cierra el Stream, y por tanto el fichero
        dataOS.close();
        System.out.println("Fichero escrito");
    }
}
```

```
ca Símbolo del sistema
C:\Practicas\Programacion>java EscribeDatosPrimitivos
Escribiendo en fichero...
Fichero escrito
C:\Practicas\Programacion>
```

Ficheros de Tipos Primitivos



```
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

public class LeeDatosPrimitivos {
    public static void main(String[] args) throws IOException {
        String ruta = "c:/practicas/programacion";
        // Declara el fichero
        File fichero = new File(ruta, "datos2.dat");
        // Crea el flujo de entrada hacia el fichero
        FileInputStream flujoSalida = new FileInputStream(fichero);
        // Conecta el flujo de bytes al flujo de datos
        DataInputStream dataIS = new DataInputStream(flujoSalida);

        // Escribe algunos datos primitivos al fichero
        System.out.println("Leyendo en fichero...");
        System.out.println(dataIS.readBoolean());
        System.out.println(dataIS.readChar());
        System.out.println(dataIS.readDouble());
        System.out.println(dataIS.readInt());
        System.out.println(dataIS.readFloat());
        System.out.println(dataIS.readLong());
        System.out.println(dataIS.readUTF());

        // Cierra el Stream, y por tanto el fichero
        dataIS.close();
        System.out.println("Fichero leído");
    }
}
```

```
ca Símbolo del sistema
C:\Practicas\Programacion>java LeeDatosPrimitivos
Leyendo en fichero...
true
M
2.22222222E8
2006
9999999.0
42398432
Hola Mundo
Fichero leído
C:\Practicas\Programacion>
```

Agenda



- Introducción
 - Conceptos Básicos de Ficheros
 - Operaciones sobre ficheros
 - Tipos de fichero
- Ficheros en Java
 - Conceptos Básicos de Entrada/Salida
 - Ficheros Binarios
 - Ficheros de Tipos Primitivos
 - **Ficheros de Texto**
 - Ficheros de Acceso Aleatorio
 - Serialización
- Algoritmos sobre Ficheros
- Resumen



Ficheros de Texto



- Escribir / Leer **cadena de texto**
- Se utilizan las clases Reader / Writer
- **PrintWriter**: Hereda de Writer, permite escribir texto en un OutputStream
 - En el constructor recibe el OutputStream a utilizar
 - Tiene métodos para escribir en forma de texto todos los tipos básicos y los Strings.
 - Métodos duplicados para insertar retorno de carro al final del dato escrito (print / println)
- **PrintStream** es similar a PrintWriter, pero sus métodos están deprecados (la clase no, porque todavía se usa en System.out) -> Por defecto utilizar PrintWriter que es más moderna.

Ficheros de Texto



```
import java.io.*;

public class EscribeTexto {
    public static void main(String[] args) throws IOException {
        String ruta = "c:/practicas/programacion";
        // Declara el fichero
        File fichero = new File(ruta, "datos3.dat");
        // Crea el flujo de salida hacia el fichero
        FileOutputStream flujoSalida = new FileOutputStream(fichero);
        // Conecta el flujo de bytes al flujo de datos
        PrintWriter pw = new PrintWriter(flujoSalida);

        // Escribe algunos textos al fichero
        System.out.println("Escribiendo en fichero...");
        pw.write("Hola Mundo");
        pw.write("Adios Mundo");
        pw.print("Hola otra vez");
        pw.println("Adios otra vez");
        // Cierra el Stream, y por tanto el fichero
        pw.close();
        System.out.println("Fichero escrito");
    }
}
```

Simbolo del sistema

```
C:\Practicas\Programacion>java EscribeTexto
Escribiendo en fichero...
Fichero escrito
C:\Practicas\Programacion>
```

```
1 Hola MundoAdios MundoAdios otra vez
2 Hola otra vez
```

Ficheros de Texto



```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class LeeTexto {
    public static void main(String[] args) throws IOException {
        String ruta = "c:/practicas/programacion";
        // Declara el fichero
        File fichero = new File(ruta, "datos3.dat");
        // Crea el flujo de entrada hacia el fichero
        FileReader fr = new FileReader(fichero);
        // Crea un bufferedReader para leer linea a linea
        BufferedReader br = new BufferedReader(fr);

        // Lee cadenas del fichero
        System.out.println("Leyendo fichero...");
        String s = br.readLine();
        while(s != null) {
            System.out.println("Linea leida: " + s);
            s = br.readLine();
        }
        // Cierra el Stream, y por tanto el fichero
        br.close();
        System.out.println("Fichero leido");
    }
}
```

Simbolo del sistema

```
C:\Practicas\Programacion>java LeeTexto
Leyendo fichero...
Linea leida: Hola MundoAdios MundoAdios otra vez
Linea leida: Hola otra vez
Fichero leido
C:\Practicas\Programacion>
```

Agenda



- Introducción
 - Conceptos Básicos de Ficheros
 - Operaciones sobre ficheros
 - Tipos de fichero
- Ficheros en Java
 - Conceptos Básicos de Entrada/Salida
 - Ficheros Binarios
 - Ficheros de Tipos Primitivos
 - Ficheros de Texto
 - **Ficheros de Acceso Aleatorio**
 - Serialización
- Algoritmos sobre Ficheros
- Resumen



Ficheros de Acceso Aleatorio



- Se utiliza la clase **RandomAccessFile**
 - **No** está basada en el concepto de **flujos o Streams**.
 - **No** deriva de **InputStream/OutputStream** ni **Reader/Writer**
 - Permite **leer y escribir** sobre el fichero, no es necesario dos clases diferentes
 - Necesario especificar el **modo de acceso** al construir un objeto de esta clase: **sólo lectura** o **lectura/escritura**
 - Dispone de **métodos específicos de desplazamiento** como **seek(long posicion)** o **skipBytes(int desplazamiento)** para poder moverse de un registro a otro del fichero, o posicionarse directamente en una posición concreta del fichero.

Ficheros de Acceso Aleatorio



- Constructores:
 - `RandomAccessFile(File f, String modoAcceso)`
 - `RandomAccessFile(String nombreArchivo, String modoAcceso)`
- modoAcceso** puede ser: "r" (sólo lectura) o "rw" (lectura y escritura).
- Métodos:
 - void seek(long posicion)**: Sitúa el puntero de lectura/escritura en la posición indicada, desde el principio del fichero.
 - long getFilePointer()**: Devuelve la posición actual del puntero de lectura/escritura.
 - int skipBytes(int desplazamiento)**: Desplaza el puntero desde la posición actual, el número de bytes indicado por desplazamiento
 - long length()**: Devuelve la longitud o tamaño del fichero en bytes

Ficheros de Acceso Aleatorio



```
import java.io.IOException;
import java.io.RandomAccessFile;

public class FicherosAccesoAleatorio {
    public static void main(String[] args) throws IOException {
        String nombre = "c:/practicass/programacion/datos4.dat";
        // Declara el fichero de acceso aleatorio
        RandomAccessFile f = new RandomAccessFile(nombre, "rw");
        // Escribe 100 bytes al fichero
        for(int i=0; i<100; i++) {
            f.write(("byte" + i));
        }
        // Vuelve al principio del fichero
        f.seek(0);
        // Lee uno de cada dos bytes
        for(int i=0; i<50; i++) {
            int b = f.read();
            System.out.print(b + " ");
            f.skipBytes(1);
        }
        // Cierra el fichero
        f.close();
    }
}
```

```
C:\Practicass\Programacion>java FicherosAccesoAleatorio
0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98,
C:\Practicass\Programacion>
```

Agenda



- Introducción
 - Conceptos Básicos de Ficheros
 - Operaciones sobre ficheros
 - Tipos de fichero
- Ficheros en Java
 - Conceptos Básicos de Entrada/Salida
 - Ficheros Binarios
 - Ficheros de Tipos Primitivos
 - Ficheros de Texto
 - Ficheros de Acceso Aleatorio
 - **Serialización**
- Algoritmos sobre Ficheros
- Resumen



Serialización



- **Serialización:** Posibilidad de escribir/leer Objetos java en Streams.
- Para poder serializar un objeto en java deben cumplirse los siguientes requisitos:
 - Debe implementar la interfaz **Serializable** (se estudiarán las interfaces con más detenimiento en el siguiente tema).
 - Todos los objetos incluidos en él tienen que implementar la interfaz **Serializable**
- Para escribir/leer objetos serializables a un Stream se utilizan las clases java **ObjectInputStream** / **ObjectOutputStream**.

Serialización



```
import java.io.Serializable;

public class Persona implements Serializable {
    private Mano manoDerecha;
    private Mano manoIzquierda;
    private String nombre;

    public Persona(String nombre, int numDedosDerecha,
        int numDedosIzquierda) {
        this.manoDerecha = new Mano(numDedosDerecha);
        this.manoIzquierda = new Mano(numDedosIzquierda);
        this.nombre = nombre;
    }

    public String toString() {
        return "Soy una persona, me llamo " + nombre + " y tengo la" +
            " mano derecha con " + manoDerecha.numeroDedos + " dedos y la" +
            " mano izquierda con " + manoIzquierda.numeroDedos;
    }
}

import java.io.Serializable;

public class Mano implements Serializable {
    public int numeroDedos;
    public Mano(int numeroDedos) {
        this.numeroDedos = numeroDedos;
    }
}
```

Serialización



```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;

public class EscribePersona {

    public static void main(String[] args) throws IOException {
        // Crea la persona
        Persona per = new Persona("Manuel", 5, 5);
        System.out.println(per);

        String ruta = "c:/practicas/programacion";
        // Declara el fichero
        File fichero = new File(ruta, "datos5.dat");
        // Crea el flujo de salida hacia el fichero
        FileOutputStream flujoSalida = new FileOutputStream(fichero);
        // Conecta el flujo de bytes al flujo de datos
        ObjectOutputStream dataOS = new ObjectOutputStream(flujoSalida);
        System.out.println("Escribiendo la persona a fichero...");
        // Escribe la persona al fichero
        dataOS.writeObject(per);
        // Cierra el fichero
        dataOS.close();
        System.out.println("Persona escrita");
    }
}
```

```
C:\Practicas\Programacion>java EscribePe
rsona
Soy una persona, me llamo Manuel y tengo
la mano derecha con 5 dedos y la mano i
zquierda con 5
Escribiendo la persona a fichero...
Persona escrita
C:\Practicas\Programacion>
```

Serialización



```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;

public class LeePersona {

    public static void main(String[] args) throws IOException, ClassNotFoundException {
        String ruta = "C:/practicas/programacion";
        // Declara el fichero
        File fichero = new File(ruta, "datos5.dat");
        // Crea el flujo de entrada desde el fichero
        FileInputStream flujoEntrada = new FileInputStream(fichero);
        // Conecta el flujo de bytes al flujo de datos
        ObjectInputStream dataIN = new ObjectInputStream(flujoEntrada);
        System.out.println("Leyendo la persona desde el fichero...");
        // Lee la persona del fichero
        Persona per = (Persona) dataIN.readObject();
        // Cierra el fichero
        flujoEntrada.close();
        System.out.println("Persona leída:");
        System.out.println(per);
    }
}
```

```
C:\Practicas\Programacion>java LeePersona
Leyendo la persona desde el fichero...
Persona leída:
Soy una persona, me llamo Manuel y tengo
la mano derecha con 5 dedos y la mano i
zquierda con 5
C:\Practicas\Programacion>
```

Agenda



- Introducción
 - Conceptos Básicos de Ficheros
 - Operaciones sobre ficheros
 - Tipos de fichero
- Ficheros en Java
 - Conceptos Básicos de Entrada/Salida
 - Ficheros Binarios
 - Ficheros de Tipos Primitivos
 - Ficheros de Texto
 - Ficheros de Acceso Aleatorio
 - Serialización
- **Algoritmos sobre Ficheros**
- Resumen



Algoritmos sobre ficheros



- Algoritmos de ordenación estudiados hasta ahora lo hacen en memoria.
- Si existe gran cantidad de datos, no se pueden tener todos a la vez en memoria.
- Entonces es necesario almacenar y ordenar los datos utilizando ficheros.
- Dos algoritmos para ordenar utilizando ficheros:
 - Algoritmo de **mezcla directa**
 - Algoritmo de **mezcla natural**

Algoritmos sobre ficheros: Mezcla Directa



- Algoritmo de **Mezcla Directa**:
 - Dividir una secuencia inicial de datos en dos subcadenas y mezclar elemento a elemento de forma ordenada.
 - El proceso se repite hasta que la secuencia inicial queda totalmente ordenada.
- Pasos:
 - 1) Se divide la secuencia inicial de datos del fichero **a** en dos mitades **b** y **c**
 - 2) Se mezclan **b** y **c** combinando elementos aislados para formar **pares ordenados**
 - 3) La secuencia resultante se almacena en el fichero **a** y se repiten los pasos 1) y 2) para formar **cuádruplos ordenados**.
 - 4) Se repiten los pasos anteriores para formar **octetos ordenados**, y así sucesivamente.

Algoritmos sobre ficheros: Mezcla Directa



- Ejemplo de **Mezcla Directa**:
- Las comillas simples (') indican fin de tupla.

Situación inicial:

fichero a: [44, 55, 12, 42, 94, 18, 6, 67]
fichero b: []
fichero c: []

1ª división:

fichero b: [44, 55, 12, 42]
fichero c: [94, 18, 6, 67]

1ª mezcla:

fichero a: [44, 94*, 18, 55*, 6, 12*, 42, 67*]

2ª división:

fichero b: [44, 94*, 18, 55*]
fichero c: [6, 12*, 42, 67*]

2ª mezcla:

fichero a: [6, 12, 44, 94*, 18, 42, 55, 67*]

3ª división:

fichero b: [6, 12, 44, 94*]
fichero c: [18, 42, 55, 67*]

3ª mezcla:

fichero a: [6, 12, 18, 42, 44, 55, 67, 97]

Algoritmos sobre ficheros: Mezcla Natural



- Algoritmo de **Mezcla Natural**:
 - Se quiere ordenar el fichero **c**
 - Se realizan **sucesivas distribuciones y mezclas** del fichero **c** en los ficheros **a** y **b** hasta lograr ordenar los datos.
 - La **distribución** consiste en repartir la secuencia original en dos secuencias, de forma que se pasan **alternativamente secuencias ordenadas (o tramos) de longitud máxima** del fichero **c** a los ficheros **a** y **b**.
 - La **mezcla** consiste en tomar una subsecuencia de **a** y otra de **b** y ordenarlas internamente para formar una sola subsecuencia ordenada, y así con todos los pares de subsecuencias (una de **a** y otra de **b**). Si **a** y **b** tienen distinto número de subsecuencias, las restantes se añaden tal cual al fichero **c** sin mezclar.

Algoritmos sobre ficheros: Mezcla Natural



- Ejemplo de **Mezcla Natural**:
- Las comillas simples (') indican fin de tupla.

Secuencia inicial:

- fichero c: [17 31 5 59 13 41 43 67 11 23 29 47 3 7 71 2 19 57 37 61]
- Distribución:
 - fichero a: [17 31' 13 41 43 67' 3 7 71' 37 61']
 - fichero b: [5 59' 11 23 29 47' 2 19 57']
- Mezcla:
 - fichero c: [5 17 31 59' 11 13 23 29 41 43 47 67' 2 3 7 19 57 71' 37 61']
- Distribución:
 - fichero a: [5 17 31 59' 2 3 7 19 57 71']
 - fichero b: [11 13 23 29 41 43 47 67' 37 61']
- Mezcla:
 - fichero c: [5 11 13 17 23 29 31 41 43 47 59 67' 2 3 7 19 37 57 61 71']
- Distribución:
 - fichero a: [5 11 13 17 23 29 31 41 43 47 59 67']
 - fichero b: [2 3 7 19 37 57 61 71']
- Mezcla:
 - fichero c: [2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 57 59 61 67 71]

Agenda



- Introducción
 - Conceptos Básicos de Ficheros
 - Operaciones sobre ficheros
 - Tipos de fichero
- Ficheros en Java
 - Conceptos Básicos de Entrada/Salida
 - Ficheros Binarios
 - Ficheros de Tipos Primitivos
 - Ficheros de Texto
 - Ficheros de Acceso Aleatorio
 - Serialización
- Algoritmos sobre Ficheros
- **Resumen**



Resumen: Para más información



- <http://java.sun.com/docs/books/tutorial/essential/io/index.html>
- <http://elvex.ugr.es/decsai/java/pdf/C1-files.pdf>
- <http://home.cogeco.ca/~ve3ll/jatutor8.htm>
- <http://www.inf-cr.uclm.es/www/cmunoiz/mytp/ficheros.pdf>
- <http://x500.cica.es/formacion/JavaTut/Cap8/fichero.html>