## Tema 6: Almacenamiento de datos y flujos.

Los programas leen datos y los almacenan en estructuras persistentes como bases de datos o ficheros. Un registro es una agrupación de datos. Cada uno de los elementos de un registro se denomina campo.

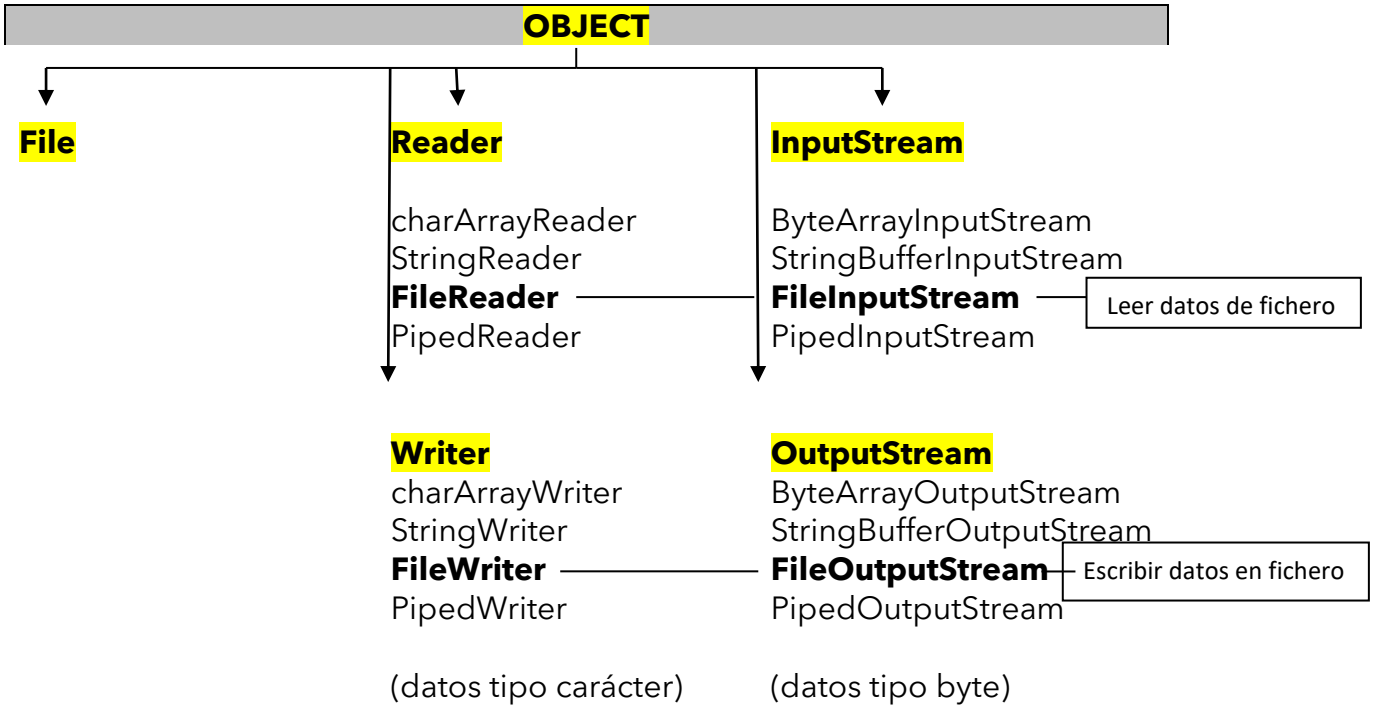
En la POO, lo normal es que un fichero almacene objetos y no registros y que en vez de almacenar campos sean atributos.

### 1. Flujo de datos

Los flujos de datos son flujos de información entre el programa y el origen o destino de la información, tratados en java mediante objetos (stream = flujo).

### 2. Clases relativas a flujos

En java.io encontramos el siguiente esquema:



## 3. Utilización de flujos

Hay clases que alteran el comportamiento de un stream definido y nos permiten manejar el flujo. Entre ellas destacan:

| | |
|---|---|
| BufferedReader c.<br>BufferedWriter c.<br>BufferedInputStream b.<br>BufferedOutpuStream b. | Añaden buffer al funcionamiento del objeto. |
| InputStreamReader c.<br>OutputStreamWriter c. | Transforman flujos de bytes en flujos de caracteres. |
| ObjectInputStream b.<br>ObjectOutputStream b. | Se utilizan en la serialización para la lectura y escritura de objetos. |
| FilterReader c.<br>FilterWriter c.<br>FiletrInputStream b.<br>FilterOutputStream b. | Aplican filtros a los stream de datos. |
| DataInputStream b.<br>DataOutputStream b. | Manejo de datos. |
| PrintWriter c.<br>PrintStream b. | Impresión de la información con métodos como write/print/println/flush() (que por ejemplo limpia el buffer de contenido). |

## 4. Ficheros de datos

Los ficheros contienen información en modo texto y binario. La información en modo texto puede ser leída por cualquier editor de texto, mientras que la información en binario, aunque pueda ser accedida, hay que conocer la escritura interna del fichero para poder interpretar los datos que contiene.

Hay diferentes formas de acceder a los datos (acceso secuencial, acceso directo, acceso por índice, acceso dinámico,...), y entre ellas destaca el acceso secuencial y el acceso aleatorio. El secuencial se basa en que el acceso al registro n, implica la lectura de los registros del 1 al n-1. El aleatorio en cambio permite acceder a un registro concreto del fichero, pero no está basada en el concepto de flujo o stream.

En el trabajo con flujos y ficheros, siempre hemos de tener en cuenta que el modo de proceder es el siguiente:

- Declarar el flujo sobre el que se va a trabajar.
- Abrir, o crear el fichero.
- Hacer las operaciones necesarias.
- Cerrar el flujo abierto.
- Cerrar el fichero.

### 4.1 Escritura secuencial en un archivo

Flujo de caracteres:

java.lang.Object
→java.io.Writer
→java.io.FileWriter

Flujo de bytes:

java.lang.Object
→java.io.OutputStream
→java.io.FileOutputStream

Existen tres constructores de la clase FileOutputStream:

- FileOutputStream(String "nombre_fichero");
- FileOutputStream(String "nombre_fichero", boolean true);//de este modo se añaden datos
- FileOutputStream(File fichero);

**Observación:**

Si no se indica ninguna ruta, el fichero se creará en el mismo directorio donde se crea el programa.

## 4.2 Lectura secuencial de un fichero

Flujo de caracteres:

java.lang.Object
→java.io.Reader
→java.io.FileReader

Flujo de bytes:

java.lang.Object
→java.io.InputStream
→java.io.FileInputStream

Existen dos constructores de la clase FileInputStream:

- FileInputStream(String "nombre_fichero");
- FileInputStream(File fichero);

## 4.3 La clase File

Un objeto de la clase File representa un archivo del sistema de archivos. Las instancias a la clase File representan nombres de archivo, no los archivos en sí mismos. Puede trabajar tanto con ficheros como con directorios. El uso de la clase File aporta métodos muy prácticos para el tratamiento de información.

java.lang.Object

→java.io.File

Existen dos constructores de la clase File:

- File(String "ruta absoluta"); obs: Atención al separador de directorios de Windows

**File** archivo1=**new File**("c\\datos\\bd.txt");

**File** archivo1=**new File**("/home/carpeta/bd.txt");

- File (String “ruta absoluta”, String “nombre_fichero”);
  **File** archivo2=**new File**(“/home/carpeta”,”bd.txt”);

- File(File ruta, String “nombre_fichero”);
  **File** dir =**new File**(“/home/carpeta”);
  **File** archivo3=new File(dir, “ bd.txt”);

Algunos de los métodos de esta clase:

boolean canRead(): informa si se puede leer la información que contiene.

boolean canWrite(): informa si se puede guardar información.

boolean isFile(): informa si es un archivo.

boolean exists(): informa si el fichero o directorio existe.

delete(): borra el fichero.

long length(): retorna el tamaño del archivo en bytes.

boolean renameTo(file): cambia el nombre al fichero o archivo.

boolean mkdir(): crea el directorio.

String [] list(): devuelve un listado de los archivos que se encuentran en el directorio.

## 5. Almacenamiento de objetos en ficheros. Persistencia. Serialización.

La serialización consiste en transformar un objeto en una secuencia de bytes de tal manera que se represente el estado de dicho objeto. Una vez serializado el objeto se puede enviar a un fichero, por la red,…

Para trabajar de este modo en una clase es necesario implementar la interfaz java.io.Serializable (no define ningún método nuevo, sólo marca la clase que va a convertirse a secuencia de bytes) y Java realizará la serialización automáticamente.

p.e. *public class Amigo implements java.io.Serializable{*

Los esquemas siguientes permiten apreciar la herencia de clases relacionadas con este apartado:

**Clases para el almacenamiento de Objetos**

FileOutputStream ficheroS=new FileOutputStream ("amigos.txt");

ObjectOutputStream flujosalidaobjetos= new ObjectOutputStream (ficheroS);

java.lang.Object

→java.io.OutputStream

→java.io.ObjectOutputStream

**Clases para la recuperación de Objetos**

File fichero =new File("amigos.txt");

ficheroE=new FileInputStream(fichero);
flujoentradaobjetos= new ObjectInputStream(ficheroE);

java.lang.Object

→java.ioInputStream

→java.io.ObjectInputStream

**Observación:**

Para serializar un fichero, no basta con implementar la interfaz Serializable. Debemos asegurarnos también de que todos los objetos incluidos en el fichero implementen la interfaz Serializable.