Tema 7: Flujo de Entrada y Salida

El alumno construirá programas con el principio de flujo de entrada y salida para procesar información a partir de un problema resuelto

7.1 Fundamentos de entrada y salida

- Los archivos tienen como finalidad guardar datos de forma permanente.
- Una vez que acaba la aplicación, los datos siguen disponibles para que otra aplicación pueda recuperarlos, para su consulta o modificación.

Archivos

- Los archivos son persistentes en la memoria
- El sistema operativo es responsable de implementar la abstracción de archivo básico sobre los dispositivos de almacenamiento
- Los sistemas operativos utilizan los archivos como primitiva y modelan otras abstracciones de recursos basándose en ella

7.1 Fundamentos de entrada y salida

- El proceso de manejo de archivos en Java se hace mediante el concepto de flujo (stream) o canal; también denominado secuencia.

7.1 Fundamentos de entrada y salida

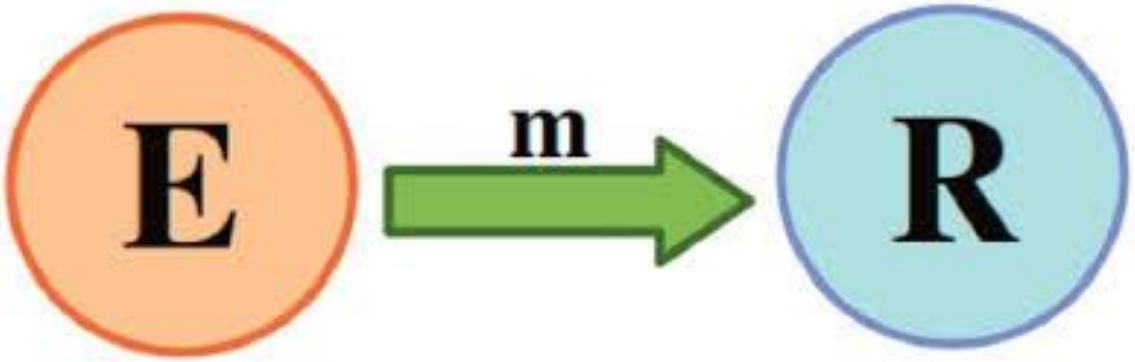
- Los flujos pueden estar abiertos o cerrados, conducen los datos entre el programa y los dispositivos externos.
- Con las clases y sus métodos proporcionados por el paquete java.io se pueden tratar archivos secuenciales, de acceso directo, archivos indexados, etcétera.

7.1 Fundamentos de entrada y salida

- En Java un archivo es una secuencia de bytes, que son la representación de los datos almacenados.
- Java dispone de clases para trabajar las secuencias de bytes como tipos primitivos (int, double, …); o también como objetos.

7.1 Fundamentos de entrada y salida

- Concepto de flujo



7.1 Fundamentos de entrada y salida

- **System.in**; entrada estándar; permite la entrada al programa de flujos de bytes desde el teclado.
- **System.out**; salida estándar; permite al programa la salida de datos por pantalla.
- **System.err**; salida estándar de errores; permite al programa salida de errores por pantalla.

7.1 Fundamentos de entrada y salida

- En Java, un archivo es simplemente un flujo externo, una secuencia de bytes almacenados en un dispositivo externo (normalmente en disco).
- Los programas leen o escriben en el flujo, que puede estar conectado a un dispositivo o a otro.

7.1 Fundamentos de entrada y salida

- **El concepto o la aplicación de flujo de información es una abstracción** de tal forma que las operaciones que realizan los programas son sobre el flujo independientemente del dispositivo al que esté asociado.
    - Flujo a Dispositivo de salida
    - Flujo a dispositivo secundario
    - Flujo a microcontroladores

7.1 Fundamentos de entrada y salida

En Java se establecen diferentes estrategias de entrada y salida. Las 3 más importantes son:

- Texto plano
- Entrada y salida binaria
- Objetos

7.1 Fundamentos de entrada y salida

- **Texto plano**

La E/S de texto es una forma práctica de almacenar tipos de datos primitivos ya que para leer o escribir archivos de texto es posible usar casi cualquier otro tipo de programa de computadora (como los procesadores de texto y las hojas de cálculo).

7.1 Fundamentos de entrada y salida

- **Texto plano**

La computadora transforma datos primitivos de formato nativo en un formato de texto legible para archivos.

El usuario puede crear o ver archivos de texto con casi cualquier editor de texto.

7.1 Fundamentos de entrada y salida

- **Binarios**

Los archivos binarios guardan datos de manera más eficiente que los archivos de texto o los archivos objeto

Los archivos obtienen datos primitivos en formato nativo.

No es posible crear ni ver archivos binarios con un editor de texto, aunque los archivos binarios son bastante compactos. (optimizan memoria)

7.1 Fundamentos de entrada y salida

- **Archivos tipo Objeto**

Cuando los datos a almacenar son complejos, implican objetos o una combinación de objetos y primitivos, los archivos objeto son más fáciles de usar.

7.1 Fundamentos de entrada y salida

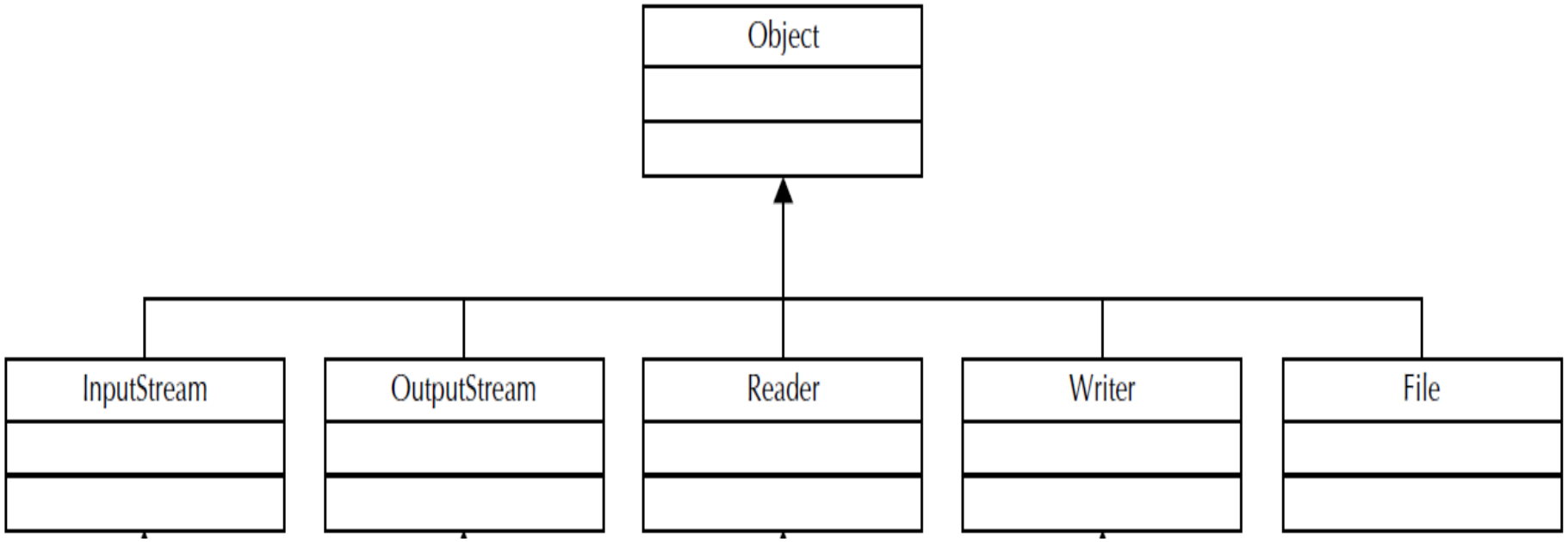
- **Archivos tipo Objeto**

El cpu al realizar el procesamiento, descompone objetos en datos primitivos para archivos, aunque también obtiene encabezados descriptivos.

No es posible crear ni ver archivos de tipo objeto con un editor de texto, aunque la codificación se minimiza.

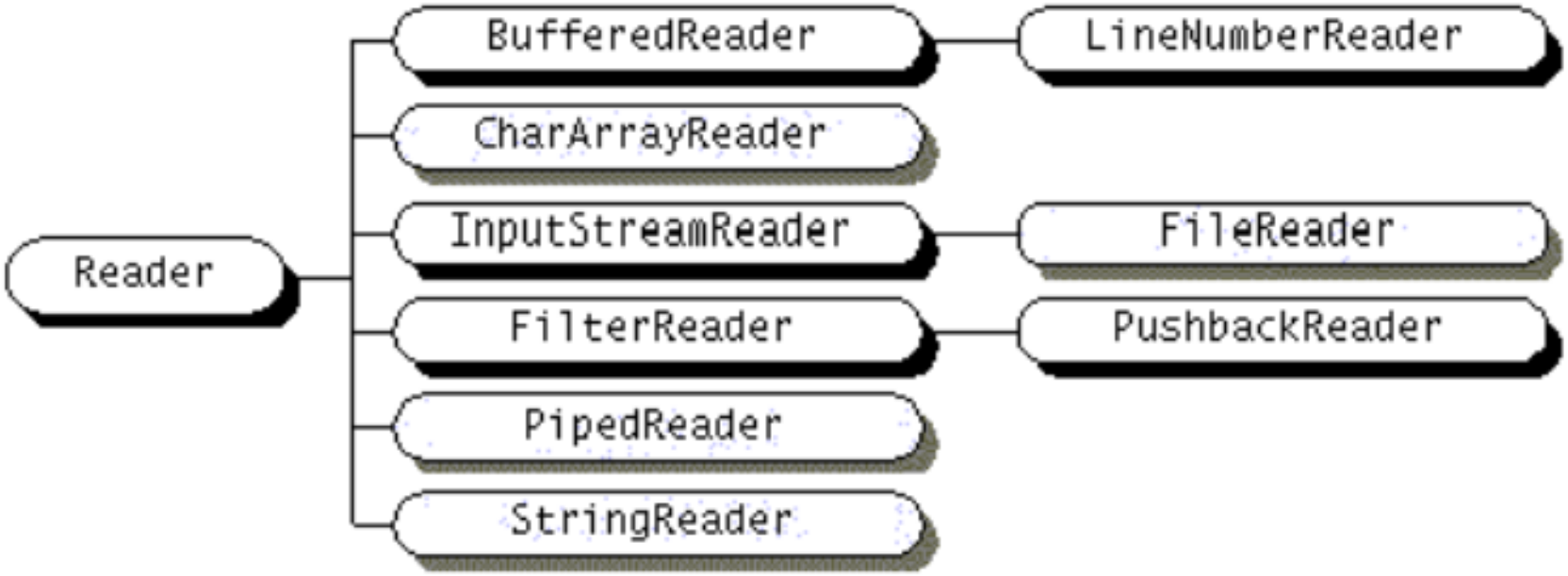
6.2 Clases de los flujos de datos

Jerarquía I/O



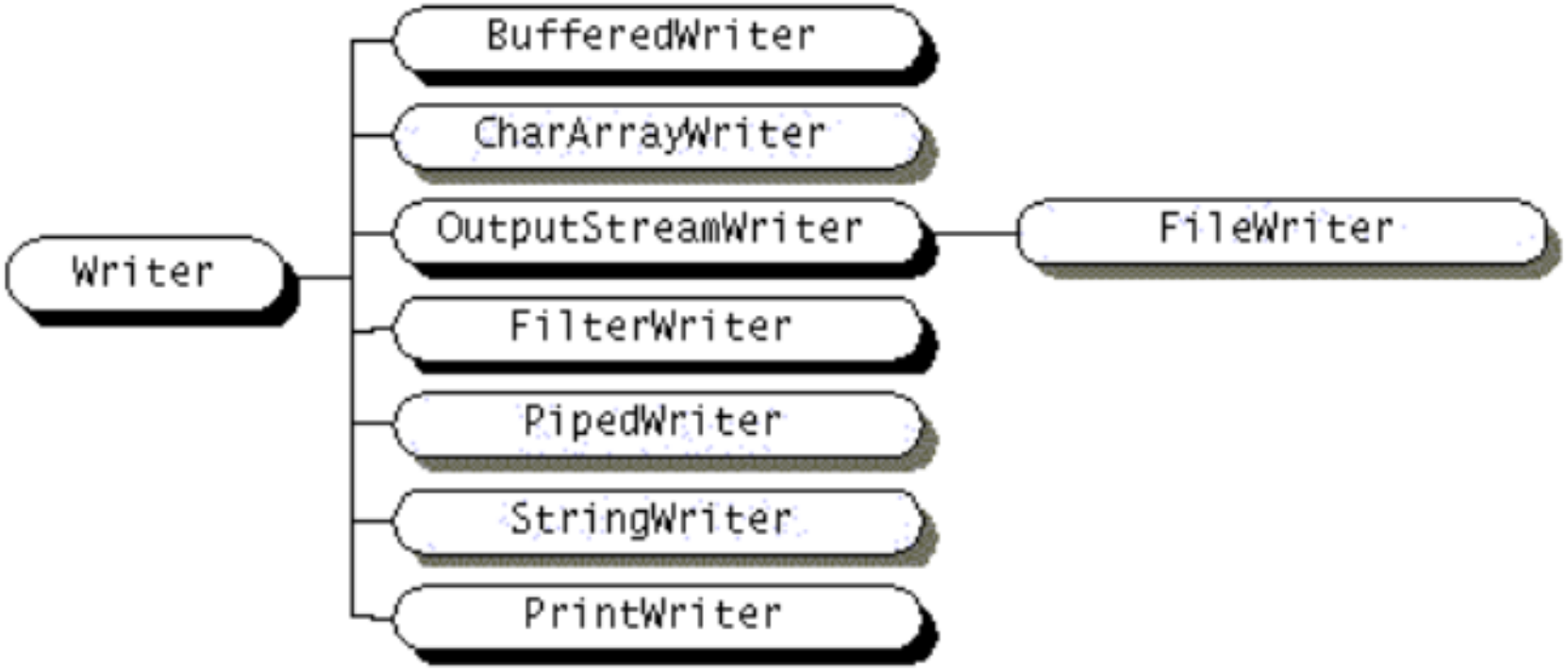
7.2 Clases de los flujos de datos

Archivos de texto



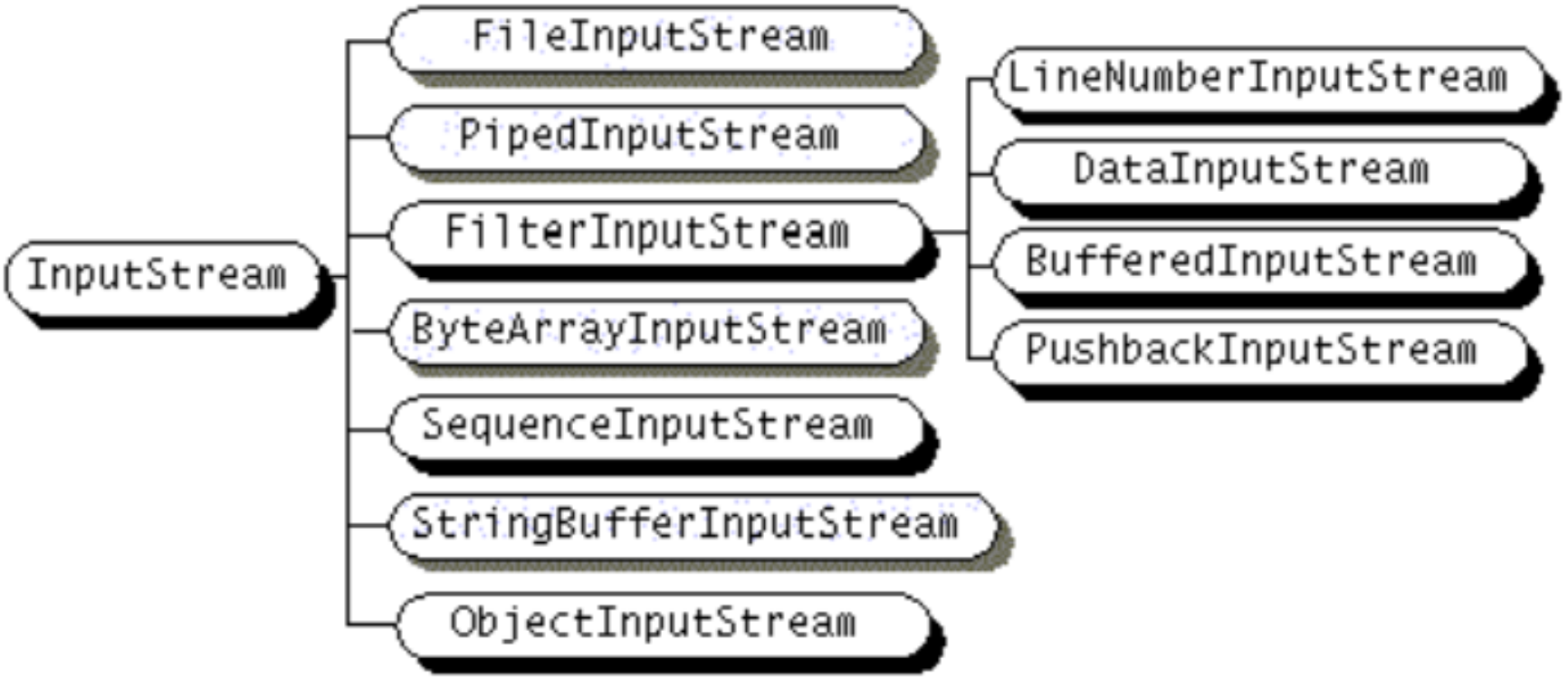
7.2 Clases de los flujos de datos

Archivos de texto



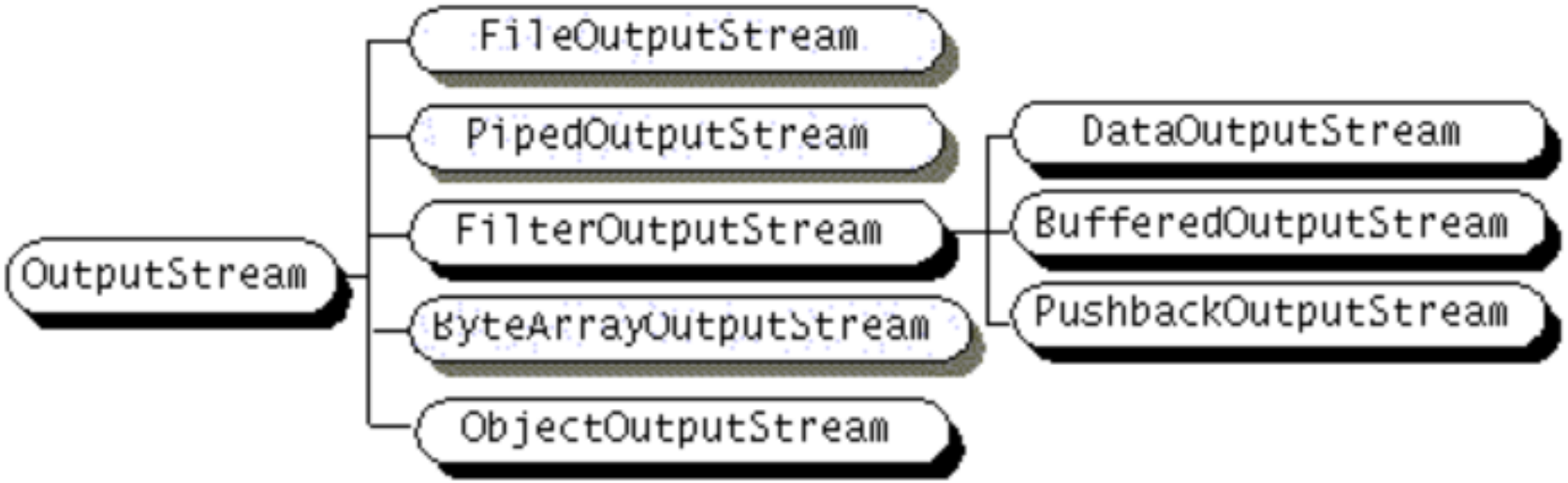
7.2 Clases de los flujos de datos

Archivos Binarios/Objeto



7.2 Clases de los flujos de datos

Archivos Binarios/Objeto



7.3 Flujos de entrada y salida

Escritura en archivos de texto

Por lo general en el manejo y tratamiento de archivos en programas orientados a objetos se siguen 3 pasos

- Abrir el archivo para instanciar las clases idóneas
- Escribir o leer desde el archivo
- Cerrarlo

7.3 Flujos de entrada y salida

- Para crear un archivo de texto, la clase PrintWriter debe instanciarse así:

```
PrintWriter <identificador>;
...
<identificador> = new PrintWriter(<filename>);
```

- No hay ningún comando “open” explícito.

7.3 Flujos de entrada y salida

¿Qué sucede?

- Siempre que se instancia un objeto PrintWriter, también se obtiene un objeto de la clase FileWriter.
- El objeto FileWriter usa uno de los métodos *write* que hereda de Output-StreamWriter, para transformar un *flujo* de caracteres Unicode en bytes.
- FileWriter proporciona un *buffer* para ese flujo final de bytes.

7.3 Flujos de entrada y salida

Para añadir texto nuevo a un archivo, es necesario utilizar de manera diferente el constructor de la clase PrintWriter

```
PrintWriter <identificador>;
   ...
   <identificador> =  new PrintWriter(
new FileWriter(<filename>, true));
```

7.3 Flujos de entrada y salida

Lectura de archivos de texto

- Es posible abrir un archivo para entrada usando una instancia de la clase FileReader con el argumento del nombre del archivo.
- Eso permite leer un carácter a la vez, una línea completa, una palabra completa o algún tipo de número.
- La clase Scanner cuenta con métodos que ejecutan estas y otras operaciones.

7.3 Flujos de entrada y salida

- Es recomendable instanciar las clases FileReader y Scanner juntas en una sola instrucción

```
Scanner <identificador>;
...
<identificador> =
new Scanner(new FileReader(<filename>));
```

7.3 Flujos de entrada y salida

Archivos binarios

- Las clases FileInputStream y FileOutputStream se utilizan para leer o escribir bytes en un archivo; objetos de estas dos clases son los flujos de entrada y salida, respectivamente, a nivel de bytes.
- Los constructores de ambas clases permiten crear flujos (objetos) asociados a un archivo que se encuentra en cualquier dispositivo.
- El archivo queda abierto

7.3 Flujos de entrada y salida

Archivos binarios

- La clase FileInputStream dispone de métodos para leer un byte o una secuencia de bytes, todos con visibilidad public.
- Es importante tener en cuenta la excepción que pueden *lanzar* para que cuando se invoquen se haga un tratamiento de la excepción.

7.3 Flujos de entrada y salida

```
FileInputStream(String nombre) throws
FileNotFoundException;
```

Crea un objeto para un archivo binario con el nombre de archivo como parámetro|.

```
FileInputStream(File nombre) throws
FileNotFoundException;
```

Crea un objeto inicializado con el objeto archivo pasado como argumento.

```
int read( ) throws IOException;
```

Lee un byte del flujo asociado. Devuelve −1 si alcanza el fin del archivo.

7.3 Flujos de entrada y salida

- `int read(byte[ ] s) throws IOException;`

Lee una secuencia de bytes del flujo y se almacena en el arreglo s. Devuelve –1 si alcanza el fin del fichero, o bien el número de bytes leídos.

- `void close( )throws IOException;`

Cierra el flujo, el archivo queda disponible para posterior uso.

7.3 Flujos de entrada y salida

Archivos de objetos

- Para que un objeto persista una vez que termina la ejecución de una aplicación se debe de guardar en un archivo de objetos.
- Por cada objeto que se almacena en un archivo se graban características de la clase y los atributos del objeto.

7.3 Flujos de entrada y salida

Archivos de objetos

- Las clases **ObjectInputStream** y **ObjectOutputStream** están diseñadas para crear flujos de entrada y salida de objetos persistentes.

7.4 Manipulación de archivos y carpetas

La clase File

- Difiere de las demás clases de la jerarquía en que no tiene que ver con el contenido de un archivo.
- Trata con el archivo de forma general, y describe el entorno del archivo: la ruta donde se encuentra y sus características

7.4 Manipulación de archivos y carpetas

- Para instanciar un objeto File para un archivo denominado calificaciones.xlsx, debe hacerse lo siguiente:

```
File calif = new File(“calificaciones.xlsx");
```

- Un objeto File no es un archivo en sí. Es simplemente un contenedor de información concerniente al mismo.

7.4 Manipulación de archivos y carpetas

- Los constructores de File permiten inicializar el objeto con el nombre de un archivo y la ruta donde se encuentra. También, inicializar el objeto con otro objeto File como ruta y el nombre del archivo.

```
public File(String nombreCompleto)
public File(String ruta, String nombre)
public File(File ruta, String nombre)
```

7.4 Manipulación de archivos y carpetas

- Con los métodos de la clase File se obtiene información relativa al archivo o ruta con que se ha inicializado el objeto.
- Por ejemplo, antes de crear un flujo para leer de un archivo es conveniente determinar si el archivo existe; en caso contrario no se puede crear el flujo.

7.4 Manipulación de archivos y carpetas

Métodos de propiedades de un archivo

- public boolean exists( )
- public boolean canWrite( )
- public boolean canRead( )
- public boolean isFile( )
- public boolean isDirectory( )
- public boolean isAbsolute( )
- public long length( )

7.4 Manipulación de archivos y carpetas

Métodos para modificar propiedades de un archivo

- public String getPath( )
- public boolean setReadOnly( )
- public boolean delete( )
- public boolean renameTo(File nuevo)
- public boolean mkdir( )