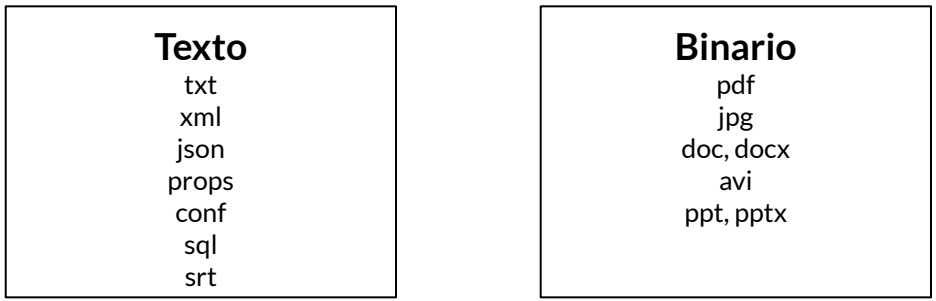
Acceso a Datos

TEMA 1

Fichero o archivo

- Conjunto de bits almacenados en un dispositivo
- No son volátiles
- Se almacenan en un directorio o carpeta
  - Solo puede haber un fichero con el mismo nombre
- Extensiones



Propiedades del sistema

- En Java tenemos el método de System Properties
- Nos permite acceder propiedades de la configuración del sistema
- Principales
  - file.separator
  - user.home
  - user.dir
  - line.separator

```
System.out.println("La propiedad del user.home es " + System.getProperty("user.home"));
```

Clase: File (declaración)

- Utilidades relacionadas con ficheros
- Para crear un objeto de tipo File
  - File(String *directorioyfichero*)
  - File(String *directorio*, String *fichero*)
  - File(File file, String fichero)
- Utiliza rutas absolutas al SO o relativas al directorio de ejecución

Clase: File (métodos)

- getName()
- getPath()
- getAbsolutePath()
- canRead()
- canWrite()
- length()
- list()
- createNewFile()
- delete()
- getParent()
- isDirectory()
- isFile()
- mkdir()
- renameTo(File *nuevoNombre*)

Tipos de ficheros de texto

Texto plano

Configuración

XML

Texto plano

- Texto libre sin formato
- Cualquier editor de texto plano es capaz de mostrarlo
- Tipos:
  - Planos: txt
  - Scripts: java, sql, js
  - Configuración: ini, props, conf
  - Información adicional: html, xml, json

Escribir ficheros de texto plano

```
FileWriter fichero = null;
PrintWriter escritor = null;

try {
  fichero = new FileWriter("archivo.txt");
  escritor = new PrintWriter(fichero) ;
  escritor.println("Esto es una línea del fichero");
} catch (IOException ioe) {
  ioe.printStackTrace() ;
} finally {
if (fichero != null)
  try {
    fichero.close();
  } catch (IOException ioe) { . . . }
}
```

Leer ficheros de texto plano

```
File fichero = null;
FileReader lector = null;
BufferedReader buffer = null;

try {
  buffer = new BufferedReader(new FileReader(new File("archivo.txt")));
  String linea = null;
  while ((linea = buffer.readLine()) != null)
    System.out.println(linea);
} catch (FileNotFoundException fnfe) {
  fnfe.printStackTrace();
} catch (IOException ioe) {
  ioe.printStackTrace();
} finally {
  if (buffer != null)
  try {
    buffer.close();
  } catch (IOException ioe) { . . . }
}
```

Ficheros de configuración

- El API de Java nos sirve de librerías para tratar con ficheros de configuración
- Trabaja a bajo nivel, nos abstraemos como programadores

```
# Fichero de configuración
# Thu Sep 01 10:49:39 CET 2019

user=usuario
password=micontrasena
server=localhost
port=3306
```

Generar fichero de configuración

```
Properties configuracion = new Properties();
configuracion.setProperty("user", miUsuario);
configuracion.setProperty("password", miContrasena);
configuracion.setProperty("server", elServidor);
configuracion.setProperty("port", elPuerto);
try {
  configuracion.store(new FileOutputStream("configuracion.props"),"Fichero de configuración");
} catch (FileNotFoundException fnfe ) {
  fnfe.printStackTrace();
} catch (IOException ioe) {
  ioe.printStackTrace();
}
```

Leer fichero de configuración

```
Properties configuracion = new Properties();
try {
  configuracion.load(new FileInputStream("configuracion.props"));
  usuario = configuracion.getProperty("user");
  password = configuracion.getProperty("password");
  servidor = configuracion.getProperty("server");
  puerto = Integer.valueOf(configuration.getProperty("port"));
} catch (FileNotFoundException fnfe ) {
  fnfe.printStackTrace();
} catch (IOException ioe) {
  ioe.printStackTrace();
}
```

Flujos

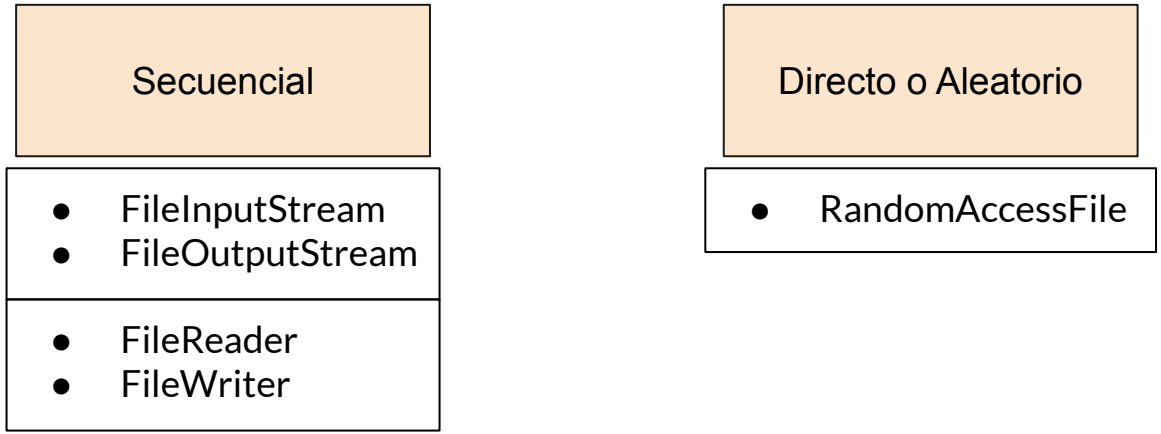
Flujos de Bytes (8 bits)

- InputStream
  - ByteArrayInputStream
  - StringBufferInputStream
  - FileInputStream
  - PipedInputStream
  - FilterInputStream
- OutputStream
  - ByteArrayOutputStream
  - FileOutputStream
  - PipedOutputStream
  - FilterOutputStream

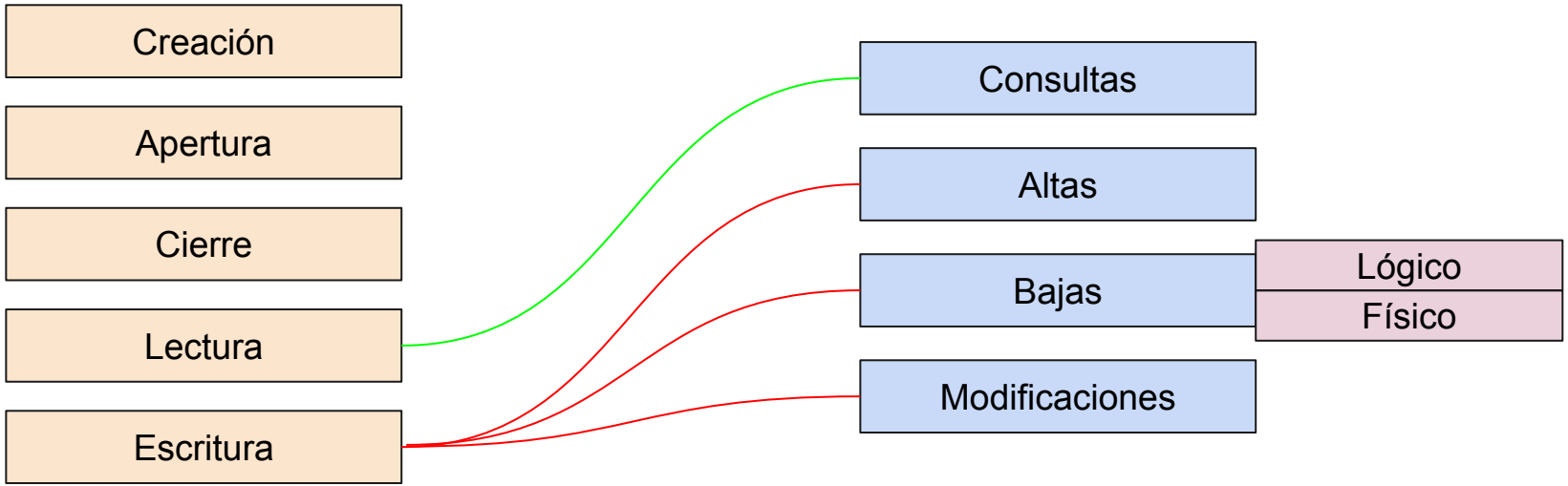
Flujos de Caracteres (16 bits)

- InputStream > **Reader**
- OutputStream > **Writer**
- FileInputStream > **FileReader**
- FileOutputStream > **FileWriter**
- StringBufferInputStream > **StringReader**
- ByteArrayInputStream > **CharArrayReader**
- ByteArrayOutputStream > **CharArrayWriter**
- PipedInputStream > **PipedReader**
- PipedOutputStream > **PipedWriter**
- **BufferedReader**
- **BufferedWriter**

Formas de acceso a un fichero



Operaciones sobre ficheros



Pros y contras de ficheros de acceso aleatorio

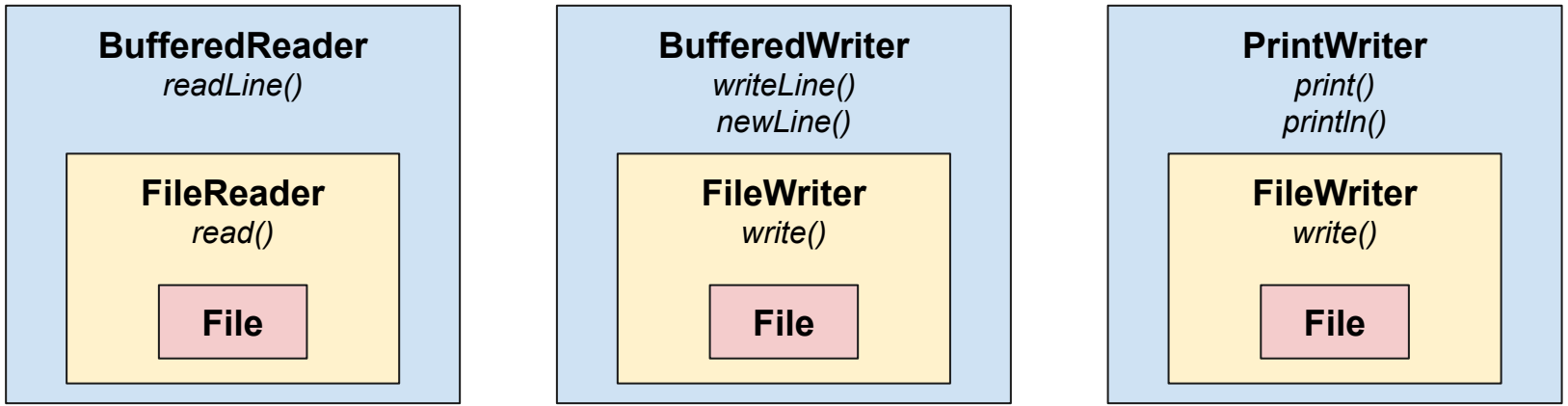
- Ventajas:
  - Rápido acceso a una posición determinada para leer o escribir
- Inconvenientes:
  - Establecer relaciones entre el contenido y la clave
  - Posiciones ocupadas
    - Recurrir a la zona de excedentes
  - Desaprovechamiento del espacio
    - Posiciones no ocupadas entre un registro y otro

Ficheros de texto y binarios

Ficheros de texto

- **File** como objeto principal
- **FileReader** y **FileWriter** para leer y escribir caracteres
  - Reciben como argumento en el constructor el objeto **File**
  - **read()** y **write()** para leer y escribir caracteres.
- **BufferedReader** y **BufferedWriter** para leer y escribir líneas completas
  - Reciben como argumento en el constructor el objeto **FileReader** y **FileWriter**
  - **readline()** lee una línea
  - **write()** escribe en el fichero pero para hacer salto de línea hay que invocar al método **newLine()**
- **PrintWriter** para escribir líneas directamente
  - Reciben como argumento en el constructor el objeto **FileWriter**
  - **print()** para escribir y **println()** para escribir una línea

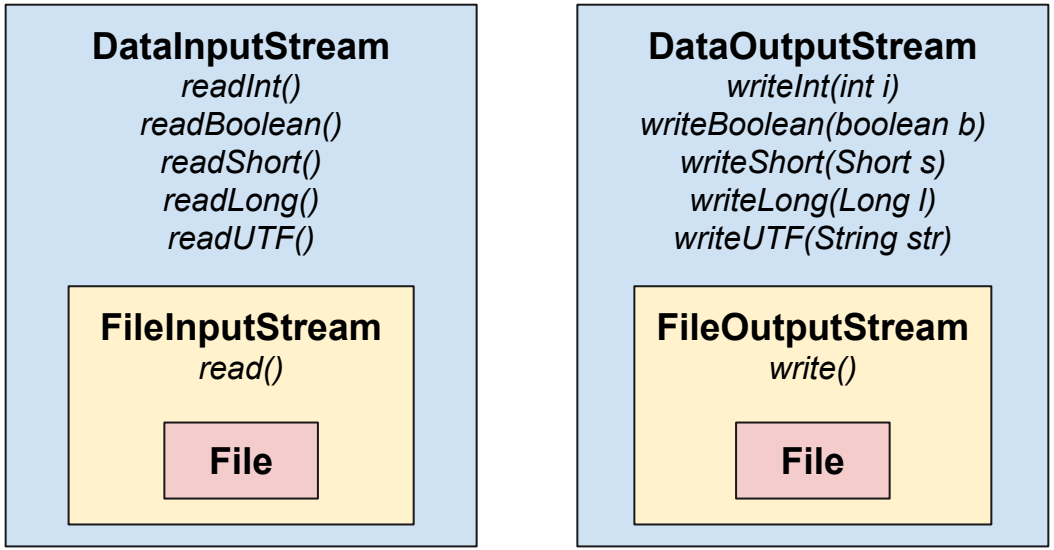
Esquemas de ficheros de texto



Ficheros binarios

- **File** como objeto principal
- **FileInputStream** y **FileOutputStream** para leer y escribir caracteres
  - Reciben como argumento en el constructor el objeto **File**
  - **read()** y **write()** para leer y escribir bytes.
- **DataInputStream** y **DataOutputStream** para leer y escribir tipos primitivos
  - Reciben como argumento en el constructor el objeto **FileInputStream** y **FileOutputStream**
  - **writeByte(), writeInt(), writeFloat(), writeUTF(), readByte(), readInt(), readFloat(), readUTF(),** etc.

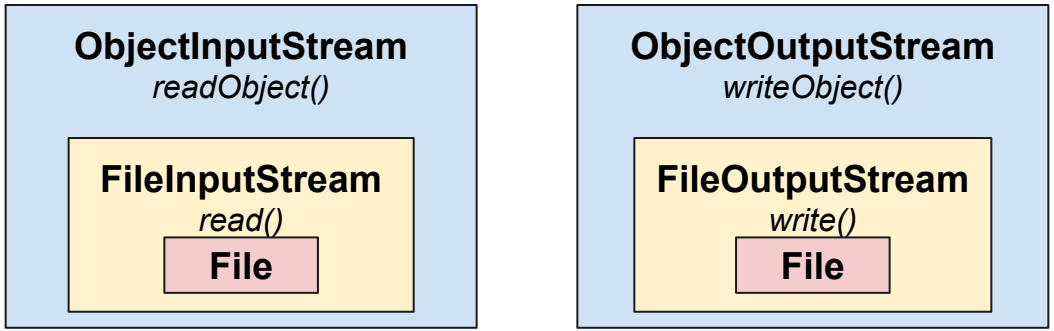
Esquemas de ficheros binarios



Objetos serializables

- **File** como objeto principal
- **FileInputStream** y **FileOutputStream** para leer y escribir caracteres
  - Reciben como argumento en el constructor el objeto **File**
  - **read()** y **write()** para leer y escribir bytes.
- **ObjectInputStream** y **ObjectOutputStream** para leer y escribir objetos
  - Reciben como argumento en el constructor el objeto **FileInputStream** y **FileOutputStream**
  - **writeObject** y **readObject** se sirven para escribir y leer objetos y guardarse en el fichero .dat

Esquema de ficheros serializables



Ficheros de acceso aleatorio

- La clase utilizada es RandomAccessFile
- Se puede declarar de dos formas:

RandomAccessFile raf = new RandomAccessFile(String nombre, String modo de acceso);

RandomAccessFile raf = new RandomAccessFile(File f, String modo de acceso);

- Modos de acceso:
  - “r” para solo lectura
  - “rw” para lectura y escritura

Ficheros de acceso aleatorio

- Maneja un fichero de posición sobre el fichero
- Cuando se hacen llamadas a *read()* o *write()* ajusta el puntero según la cantidad de bytes leídos
- Métodos más importantes:
  - *long getFilePointer()*
  - *void seek(long posicion)*
  - *long length()*
  - *int skipBytes(int desplazamiento)*

Serialización de objetos en XML: Escritura

- Librería XStream
  - ***XStream xs = new XStream();***
- Cambiar el nombre a las etiquetas XML
  - ***xs.alias("mietiqueta", claseutilizada.class);***
- Quitar etiquetas que vienen implícitas
  - ***xs.addImplicitCollection(claseutilizada.class, "mietiqueta");***
- Guardar objeto
  - ***xs.toXML(objetoquequieroguardar, new FileOutputStream("fich.xml"));***

Serialización de objetos en XML: Lectura

- Librería XStream
  - ***XStream xs = new XStream();***
- Cambiar el nombre a las etiquetas XML
  - ***xs.alias("mietiqueta", claseutilizada.class);***
- Quitar etiquetas que vienen implícitas
  - ***xs.addImplicitCollection(claseutilizada.class, "mietiqueta");***
- Guardar objeto
  - ***MiObjeto mo = (MiObjeto) xs.fromXML(new FileOutputStream("fich.xml"));***