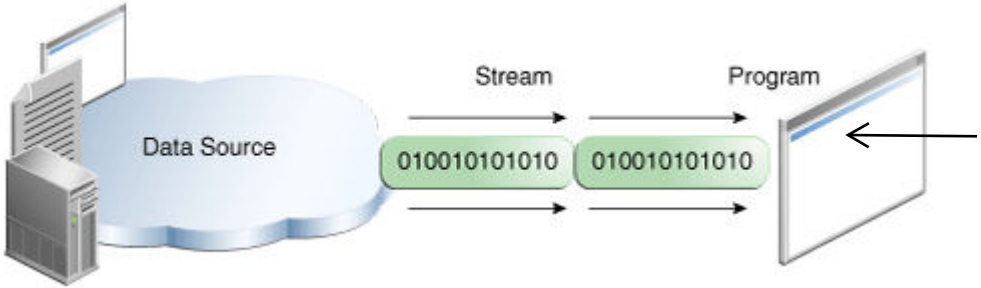
INTRODUCCIÓN
ACCESO A DATOS

FICHEROS

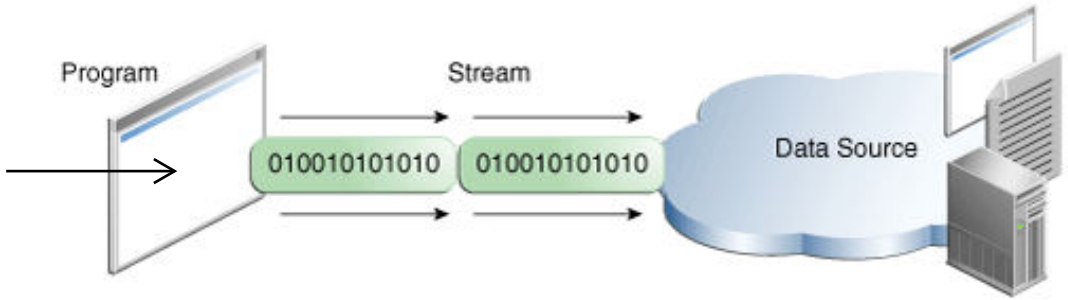
BASES DE DATOS

FLUJO (STREAMS)



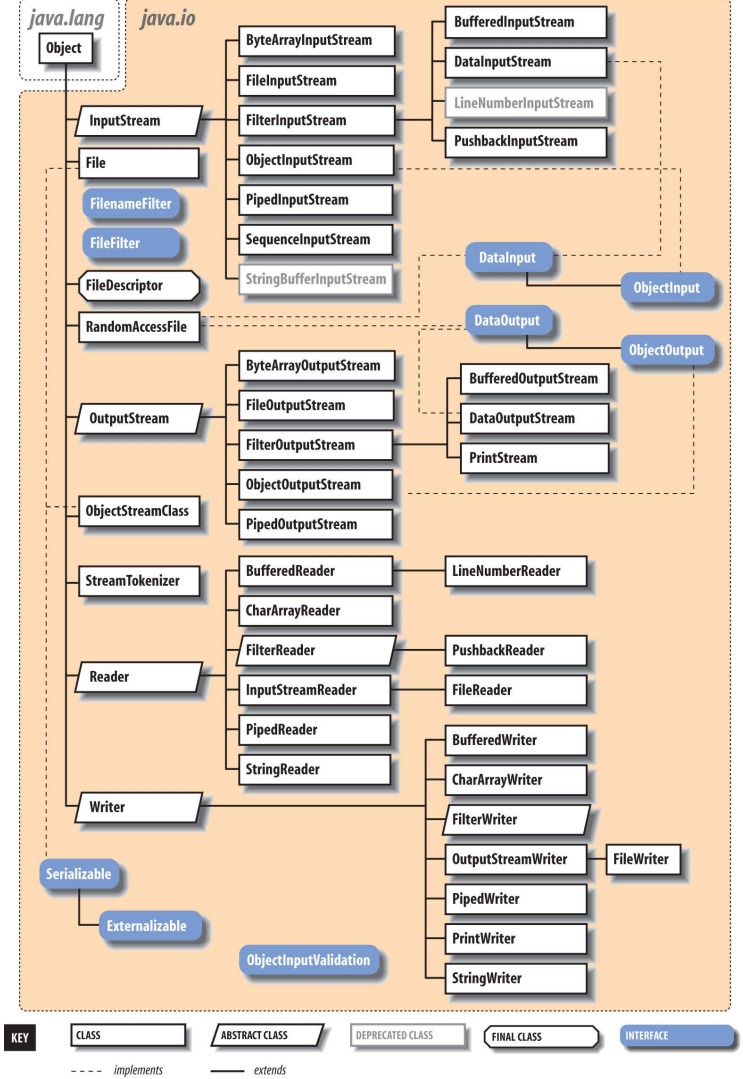
ENTRADA

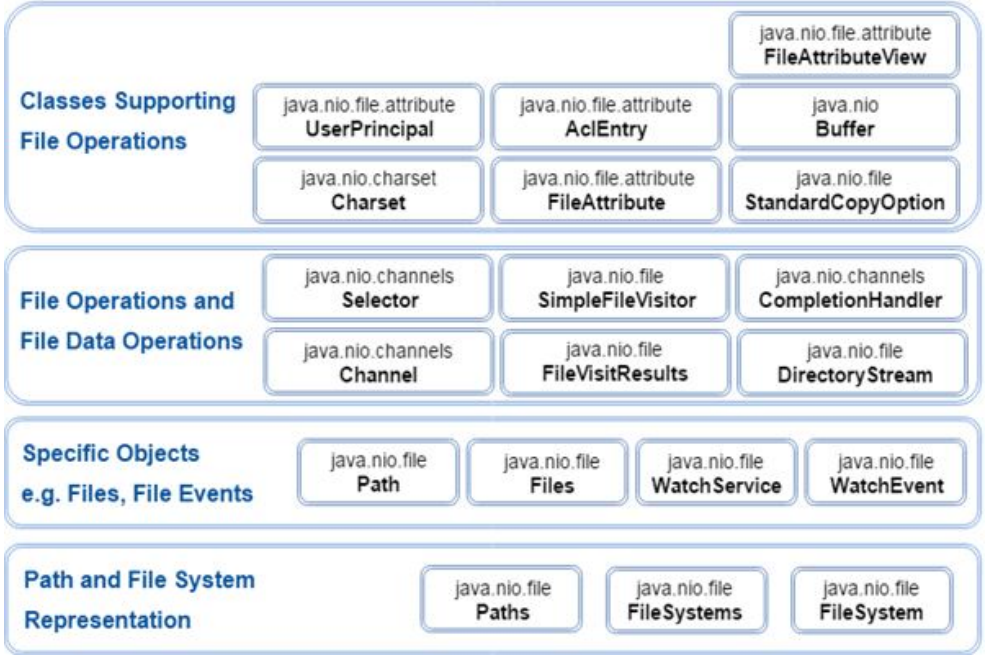
SALIDA



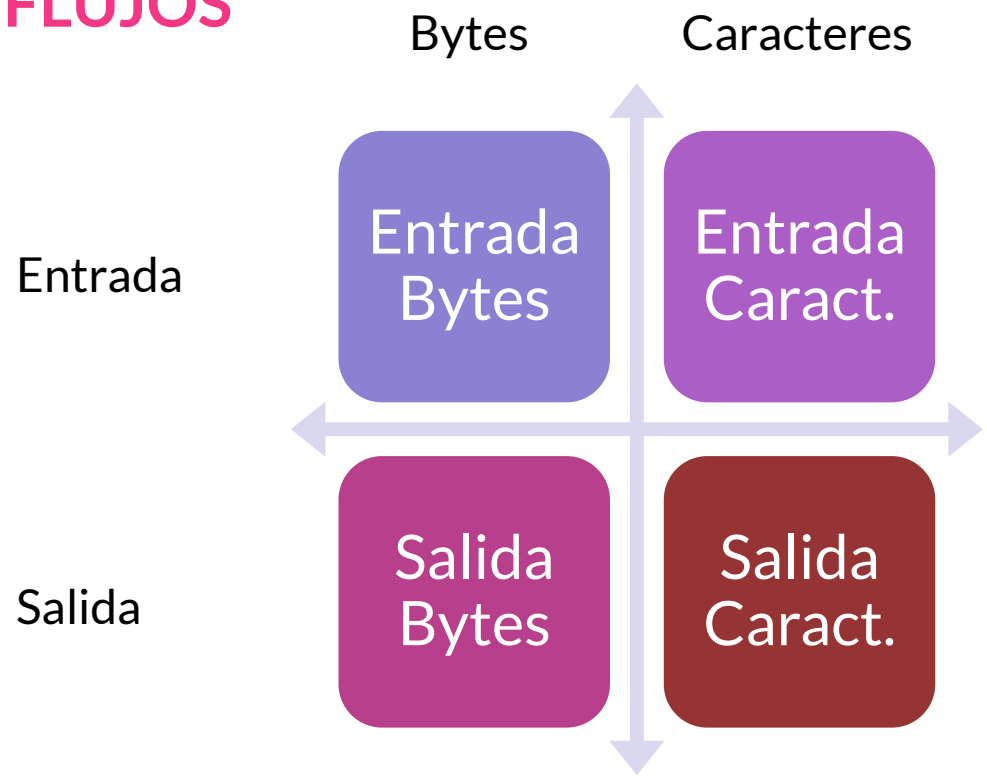
FLUJO (STREAMS)

- Canal de comunicación de nuestro programa con el exterior
- Hay un flujo que recibe los datos desde el dispositivo de entrada, por defecto el teclado
  (input stream)
- Hay un flujo que dirige los datos hacia el dispositivo de salida, por defecto la pantalla
  (output stream)
- Hay todo un ecosistema de clases, interfaces, ... que permiten gestionar estos flujos
  INDEPENDIENTEMENTE de los dispositivos de E/S


java.lang
java.io
Object
InputStream
File
FilenameFilter
FileFilter
FileDescriptor
RandomAccessFile
OutputStream
ObjectStreamClass
StreamTokenizer
Reader
Writer
Serializable
Externalizable
ObjectInputValidation
ByteArrayInputStream
FileInputStream
FilterInputStream
ObjectInputStream
PipedInputStream
SequenceInputStream
StringBufferInputStream
BufferedInputStream
DataInputStream
LineNumberInputStream
PushbackInputStream
DataInput
ObjectInput
DataOutput
ObjectOutput
ByteArrayOutputStream
FileOutputStream
FilterOutputStream
ObjectOutputStream
PipedOutputStream
BufferedOutputStream
DataOutputStream
PrintStream
BufferedReader
CharArrayReader
FilterReader
InputStreamReader
PipedReader
StringReader
LineNumberReader
PushbackReader
FileReader
BufferedWriter
CharArrayWriter
FilterWriter
OutputStreamWriter
PipedWriter
PrintWriter
StringWriter
FileWriter
KEY
CLASS
ABSTRACT CLASS
DEPRECATED CLASS
FINAL CLASS
INTERFACE
implements
extends


Classes Supporting File Operations
java.nio.file.attribute FileAttributeView
java.nio.file.attribute UserPrincipal
java.nio.file.attribute AclEntry
java.nio Buffer
java.nio.charset Charset
java.nio.file.attribute FileAttribute
java.nio.file StandardCopyOption
File Operations and File Data Operations
java.nio.channels Selector
java.nio.file SimpleFileVisitor
java.nio.channels CompletionHandler
java.nio.channels Channel
java.nio.file FileVisitResults
java.nio.file DirectoryStream
Specific Objects e.g. Files, File Events
java.nio.file Path
java.nio.file Files
java.nio.file WatchService
java.nio.file WatchEvent
Path and File System Representation
java.nio.file Paths
java.nio.file FileSystems
java.nio.file FileSystem

FLUJOS


Bytes
Caracteres
Entrada
Salida
Entrada Bytes
Entrada Caract.
Salida Bytes
Salida Caract.

FLUJOS DE ENTRADA

- Patrón básico de uso de flujos de entrada:

```
Abrir el flujo
Mientras hay datos que leer
    Leer datos del flujo
    Procesarlos
Cerrar el flujo
```

FLUJOS DE SALIDA

- Patrón básico de uso de flujos de salida:

```
Abrir el flujo
Mientras hay datos que escribir
    Escribir datos en el flujo
Cerrar el flujo
```

FLUJOS DE ENTRADA DE CARACTERES

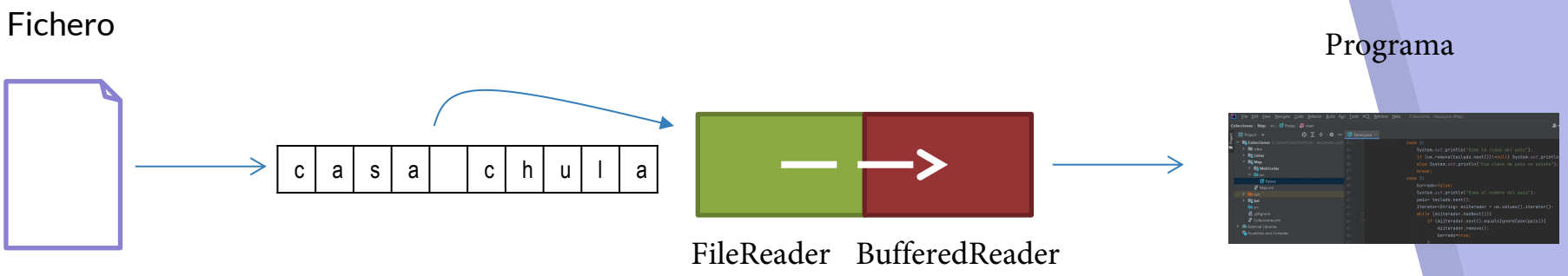
- **Reader**: clase abstracta, padre de la mayoría de los flujos de caracteres.
- **FileReader**: flujo que permite leer de un fichero, caracter a caracter.
- **BufferedReader**: flujo que permite leer líneas de texto.
- ***Scanner***: clase que permite leer texto con formatos (tipo de datos,...), iteraciones...

MÉTODOS ENTRADA DE CARACTERES

- **Reader**:
  - close()
  - read(): lee el siguiente carácter devolviendo el entero asociado o -1 si ya no hay más datos que leer
- **FileReader**:
  - FileReader(File fichero)
  - FileReader(String nombre_fichero)
  - FileReader(File fichero, Charset codificación)
  - getEncoding(): devuelve la codificación del fichero (ascii,utf-16,...)
- **BufferedReader**:
  - BufferedReader(Reader entrada)
  - readLine(): lee la siguiente línea completa del fichero o null si ya no hay más datos que leer

FLUJOS HACIA OTROS FLUJOS

- Podemos construir flujos que leen (o escriben) de otros flujos (encadenados).



MÉTODOS ENTRADA DE CARACTERES

- **Scanner**:
  - Scanner (File entrada) --> sobre un fichero
  - Scanner (Readable entrada) --> sobre un flujo de caracteres
  - Scanner (InputStream entrada) --> sobre un flujo de bytes, como por ejemplo Scanner(System.in)
  - close(): cerrar el scanner
  - useDelimiter(Pattern patrón): indica el patrón que separa los tokens
  - hasNext(): para las iteraciones de lectura
  - hasNextInt(): para iteraciones con un tipo de dato
  - next(): lee el siguiente token como un String
  - nextLine(): lee la siguiente línea completa como un String
  - nextInt(): lee el siguiente token como un int

FLUJOS DE SALIDA DE CARACTERES

- **Writer**: clase abstracta, padre de la mayoría de los flujos de caracteres.
- **FileWriter**: flujo que permite escribir en un fichero, caracter a caracter (o un string)
- **BufferedWriter**: flujo que permite escribir líneas de texto.
- **PrintWriter**: flujo que permite escribir usando formatos

MÉTODOS SALIDA DE CARACTERES

- **Writer**:
  - close()
  - write(String texto): escribe un texto completo
  - flush(): imprime el texto en el fichero!!!!
- **FileWriter**:
  - FileWriter(File fichero)
  - FileWriter(String nombre_fichero)
  - FileWriter(File fichero, Charset codificación)
  - FileWriter(File fichero, boolean append)
- **BufferedWriter**:
  - BufferedWriter(Writer salida)
  - newLine(): escribe un salto de línea
- **PrintWriter:**
  - PrintWriter(Writer salida)
  - Mismos métodos que "System.out": print, println, printf

FLUJOS DE ENTRADA DE BYTES

- ***InputStream***: clase abstracta, padre de la mayoría de los flujos de bytes.
- ***FileInputStream***: flujo que permite leer de un fichero, byte a byte.
- ***BufferedInputStream***: flujo que permite leer grupos (buffers) de bytes.

MÉTODOS ENTRADA DE BYTES

- **InputStream**:
  - close()
  - read(): lee el siguiente byte devolviendo el entero asociado o -1 si ya no hay más datos que leer
- **FileInputStream**:
  - FileInputStream(File fichero)
  - FileInputStream(String nombre_fichero)
- **BufferedInputStream**:
  - BufferedInputStream(InputStream entrada)
  - protected byte[] buf: propiedad de la clase donde se almacena la información leída.

FLUJOS DE SALIDA DE BYTES

- **OutputStream**: clase abstracta, padre de la mayoría de los flujos de bytes.
- **FileOutputStream**: flujo que permite escribir en un fichero, byte a byte.
- **BufferedOutputStream**: flujo que permite escribir grupos (buffers) de bytes.
- **PrintStream**: nos permitirá redirigir nuestra salida estándar desde la pantalla a ficheros.

MÉTODOS SALIDA DE BYTES

- **OutputStream**:
  - close()
  - write(): escribe el siguiente byte en el buffer
  - flush: escribe el contenido en el fichero!!!!
- **FileOutputStream**:
  - FileOutputStream(File fichero)
  - FileOutputStream(String nombre_fichero)
  - FileOutputStream(File fichero, boolean append)
- **BufferedOutputStream**:
  - BufferedOutputStream(OutputStream salida)
  - protected byte[] buf: propiedad de la clase donde se almacena la información que se va a escribir.
- **PrintStream**:
  - PrintStream(OutputStream salida)
  - Útil para redirigir la salida estándar a un fichero (log)

CLASE FILE

- Permite manejar ficheros y directorios, **independientemente** del sistema de archivos del ordenador
- Tiene una propiedad *separator* que almacena el carácter separador propio del sistema de archivos (/ en Linux o \ en Windows)
- File(String ruta): crear una **instancia** Fichero a partir de una ruta, que puede ser absoluta o relativa
  Ejemplo --> c:\java\fichero.txt, ..\fichero.txt c:\java\prueba
- File(URI identificador): crear una instancia Fichero a partir de una URI (identificador uniforme de recursos)
  Ejemplo --> http://example.com/languages/java/index.html
  ftp://192.168.20.1/java/fichero.txt

CLASE FILE

Nombre	Uso
isDirectory	Devuelve true si el File es un directorio
isFile	Devuelve true si el File es un fichero
createNewFile	Crea un nuevo fichero, si aun no existe.
createTempFile	Crea un nuevo fichero temporal
delete	Elimina el fichero o directorio
getName	Devuelve el nombre del fichero o directorio
getPath	Devuelve la ruta relativa del File
getCanonicalPath	Devuelve la ruta canónica del File
list	Devuelve el contenido de un directorio
mkdir	Crea un nuevo directorio

INTERFAZ PATH Y CLASE FILES

- Pertenecen al paquete java.nio
- Permite manejar de forma más eficiente diferentes sistemas de ficheros (Windows, Linux, Mac, …)
- Comparar rutas, unir rutas...
- Comparar ficheros, gestionar sus permisos de acceso...
- Crear, modificar, listar, borrar, mover directorios y ficheros...
- Leer y escribir "directamente"