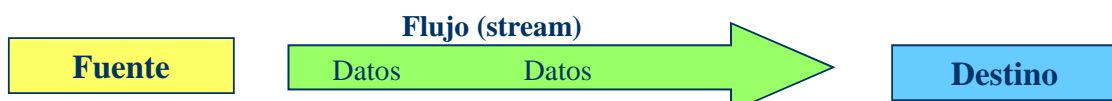


Tema 7. Ficheros

Entrada y salida de datos

- ❑ Intercambio de datos entre el programa y el exterior
 - ❑ Diversidad de dispositivos (fichero, pantalla, red, ...)
 - ❑ Diversidad de formas de comunicación
 - ❑ Modo de acceso: secuencial, aleatorio
 - ❑ Información intercambiada: binaria, caracteres, líneas
- ❑ Flujo (Stream)
 - ❑ Abstracción de cualquier fuente y/o destino de datos



Flujos estándar

- ❑ Los programas se comunican con flujos y estos flujos actúan como interfaz con el dispositivo o clase asociada
 - ❑ Operación independiente del tipo de datos y del dispositivo
 - ❑ Mayor flexibilidad (p.e. redirección, combinación)

Flujos estándar

- ❑ Flujos estándar
 - ❑ Entrada estándar - habitualmente el teclado
 - ❑ Salida estándar - habitualmente la pantalla
 - ❑ Salida de error - habitualmente la pantalla



Operación y flujos estándar en java

❑ Lectura y escritura

Lectura

Abrir un flujo
Mientras existan datos disponibles
 Leer datos
Cerrar el flujo

Escritura

Abrir un flujo
Mientras existan datos disponibles
 Escribir datos
Cerrar el flujo

Operación y flujos estándar en java

- ❑ Java tiene acceso a la entrada/salida estándar a través de la clase *java.lang.System*
 - ❑ Los flujos estándar son campos estáticos de *System*
 - ❑ Flujos
 - ❑ *System.in* implementa la entrada estándar
 - ❑ *System.out* implementa la salida estándar
 - ❑ *System.err* implementa la salida de error

Lectura y escritura en los flujos estándar

❑ Los flujos se implementan en las clases del paquete *java.io*

❑ *System.out*

❑ Instancia de la clase *PrintStream* - flujo de bytes de salida

❑ Metodos de utilidad - impresión de datos

❑ *print()* escribe en el buffer

❑ *println()* escribe en el buffer y flush. Deja el cursor en la siguiente línea.

❑ *flush()* vacía el buffer de salida escribiendo su contenido

Lectura y escritura en los flujos estándar

```
class Salida {  
    public static void main(String[] args){  
        int var=5;  
        System.out.print(var);  
        System.out.println(var);  
        System.out.print("hola" );  
        System.out.flush();  
    }  
}
```



C:\Archivos de programa\Xinox Software\JCreatorV3 LE\GE2001.exe
55
hola
Press any key to continue...

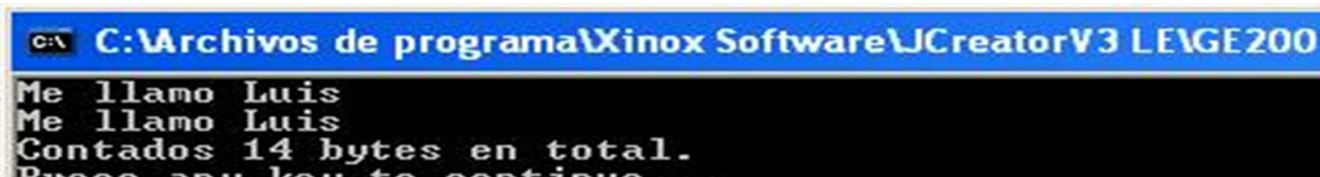
Lectura y escritura en los flujos estándar

System.in

- ❑ Instancia de la clase *InputStream* - flujo de bytes de entrada
- ❑ Metodos
 - ❑ *read()* permite leer un byte de la entrada como entero
 - ❑ *skip(n)* ignora n bytes de la entrada
 - ❑ *available()* número de bytes disponibles para leer en la entrada
- ❑ *System.err* - Funcionamiento similar a *System.out*

Ejemplo - uso flujos estándar

```
rt java.io.*;
s LecturaDeLinea {
lic static void main( String args[] ) throws IOException {
at c;
at contador = 0;
/ se lee hasta encontrar el fin de línea
while( (c = System.in.read() ) != '\n' ) {
    contador++;
    System.out.print( (char) c );
}
ystem.out.println(); // Se escribe el cambio de línea
ystem.err.println( "Contados "+ contador +" bytes en total." );
```



```
C:\Archivos de programa\Xinox Software\JCreatorV3 LEUGE200
Me llamo Luis
Me llamo Luis
Contados 14 bytes en total.
Press any key to continue...
```

Streams

Los Streams son objetos de I/O con operaciones para aceptar y mandar bytes (ASCII).

Se utilizan para leer y escribir datos independientemente de la plataforma.

Clasificación de flujos

❑ Representación de la información

- ❑ Flujos de bytes (*InputStream*, *OutputStream*)
- ❑ Flujos de caracteres (*Reader*, *Writer*)

❑ Propósito

- ❑ Entrada - (*InputStream*, *Reader*)
- ❑ Salida - (*OutputStream*, *Writer*)
- ❑ Entrada/Salida - (*RandomAccessFile*)

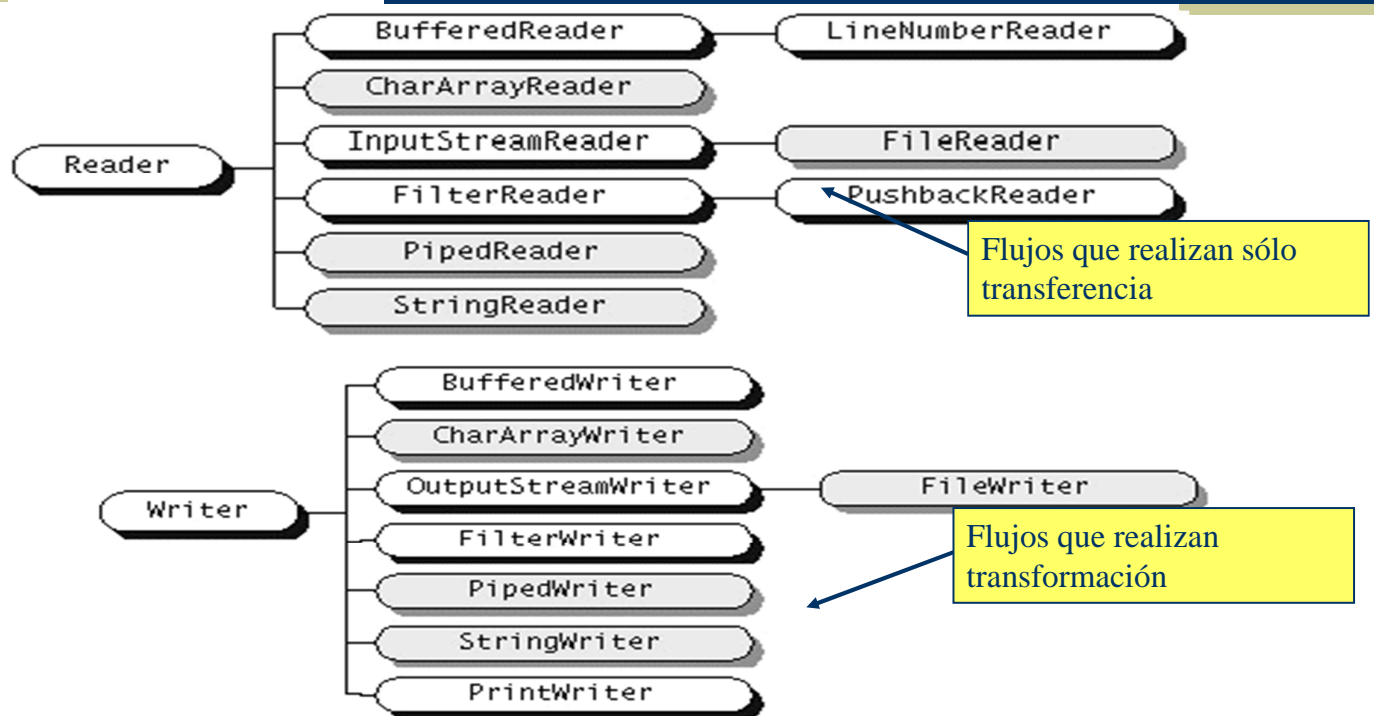
❑ Acceso

- ❑ Secuencial
- ❑ Directo o aleatorio - (*RandomAccessFile*)

❑ Por operación

- ❑ Transferencia de datos
- ❑ Transformación de los datos
 - ❑ Realizan algún tipo de procesamiento sobre los datos (p.e. *buffering*, conversiones, filtrados)

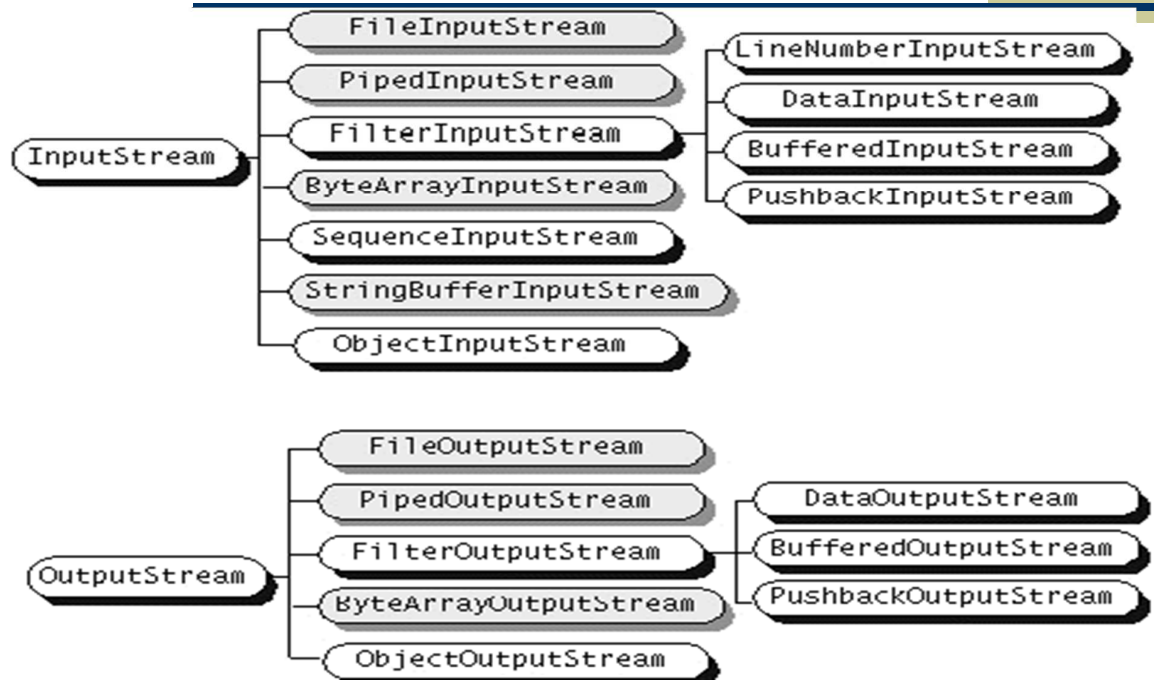
Paquete java.io - Flujos de caracteres



Java

Entrada y Salida¹³

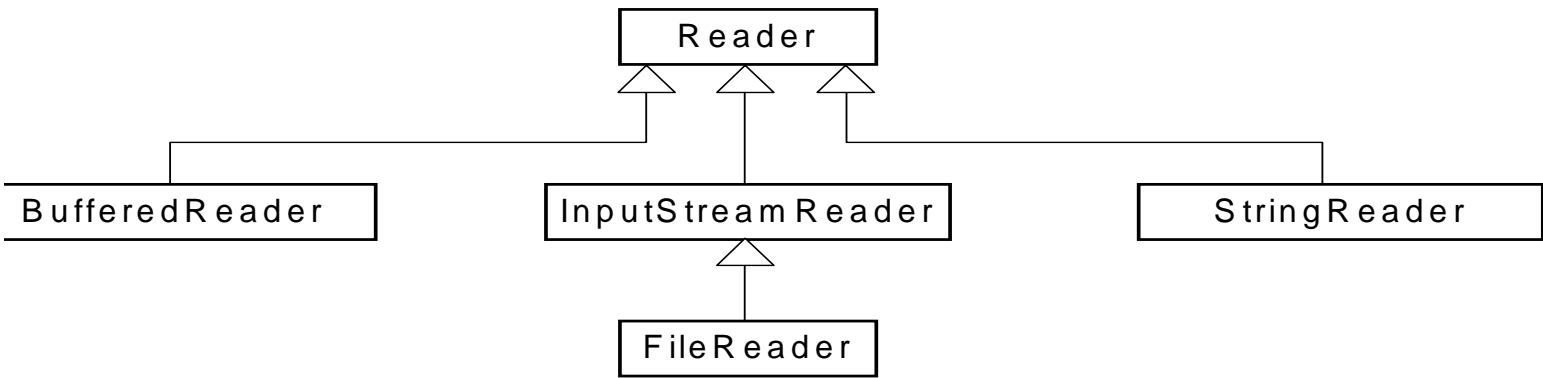
Paquete java.io - Flujos de bytes



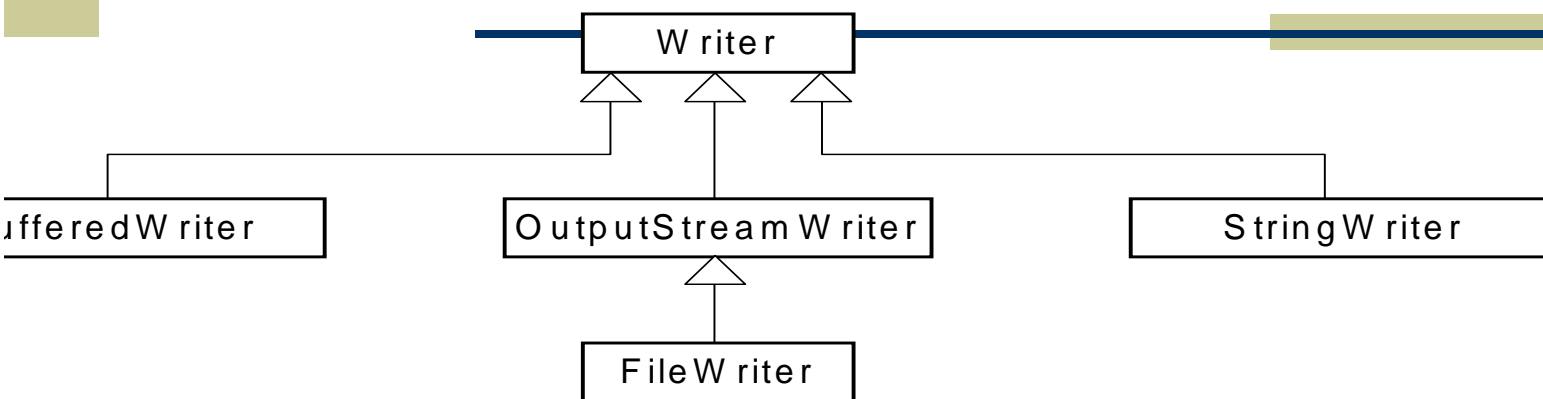
Java

Entrada y Salida¹⁴

Streams



Streams



Paquete java.io

❑ InputStreamReader

- ❑ Lee bytes de un flujo *InputStream* y los convierte en caracteres Unicode
- ❑ Métodos de utilidad
 - ❑ read () lee un único caracter
 - ❑ ready() indica cuando está listo el flujo para lectura

❑ BufferedReader

- ❑ Entrada mediante búfer, mejora el rendimiento
- ❑ Método de utilidad
 - ❑ readLine() lectura de una línea como cadena

Lectura de un String

```
InputStreamReader isr = new InputStreamReader(System.in);  
BufferedReader Teclado = new BufferedReader (isr);  
String cadena = Teclado.readLine();
```

El método debe incluir throws IOException

Lectura de un Caracter

```
public static char damePrimerChar() throws IOException {  
    String s=leeString();  
    return s.charAt(0);  
}
```

El método charAt() de la clase String devuelve el carácter de la posición dada

Lectura de un Entero

```
public int dameEntero() throws IOException {  
    String s=leeString();  
    return Integer.parseInt(s);  
}
```

El método parseInt() de la clase Integer convierte de String a tipo int

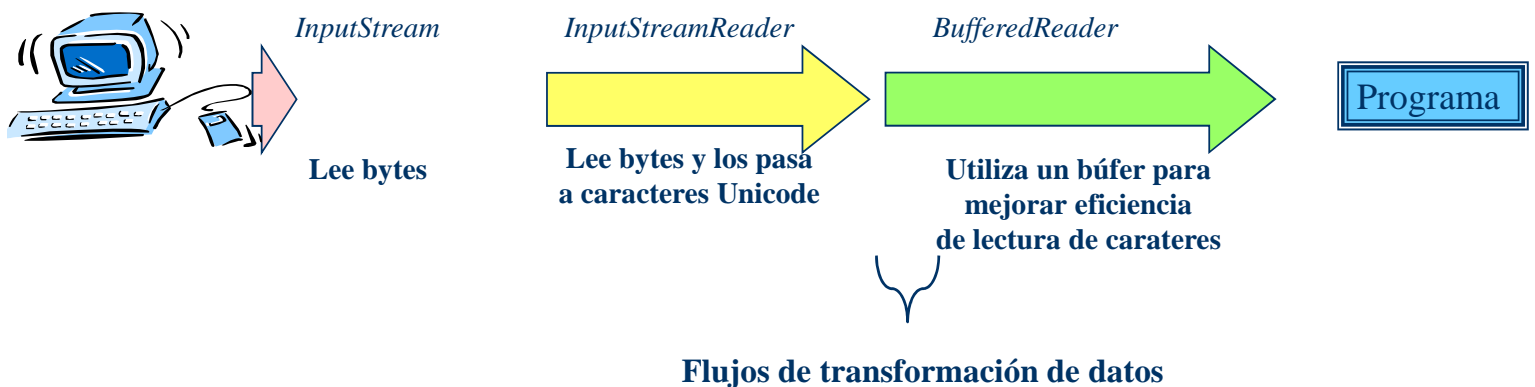
Lectura de números en coma flotante

```
public int dameDouble() throws IOException {  
    String s=leeString();  
    Double a = Double.valueOf(s);  
    return a.doubleValue();  
}
```

El envoltorio Float tiene los correspondientes métodos valueOf() y floatValue()

Combinación de flujos

- ❑ Los flujos se pueden combinar para obtener la funcionalidad deseada



Ejemplo - combinación de flujos

```
import java.io.*;

public class Eco {
    public static void main (String[] args)
        throws IOException {

        BufferedReader entradaEstandar = new BufferedReader
            (new InputStreamReader(System.in));

        String mensaje;

        System.out.println("Introducir una linea de texto:");
        mensaje = entradaEstandar.readLine();
        System.out.println("Introducido:\""+mensaje + "\"");
    }
}
```

Java

Entrada y Salida²³

La clase Teclado -- Javadoc

```
package es.ucm.esi;
import java.io.*;

/**
 * <p> La clase <em>Teclado</em> permite hacer transparente la lectura sencilla
 * los tipos mas comunes de datos desde la entrada estandar */
public class Teclado {
    /** variable de clase asignada a la entrada estandar del sistema */
    public static BufferedReader entrada =
        new BufferedReader(new InputStreamReader(System.in));

    /** lee una cadena desde la entrada estandar
     * @return cadena de tipo String
     * @exception excepciones No lanza ningun tipo de excepcion de entrada/salida
     */
    public static String leerString() {
        String cadena="";
        try {
            cadena = new String(entrada.readLine());
        } catch (IOException e) {
            System.out.println("Error de E/S");
        }
        return cadena;
    } // la clase Teclado continua
}
```

Java

Entrada y Salida

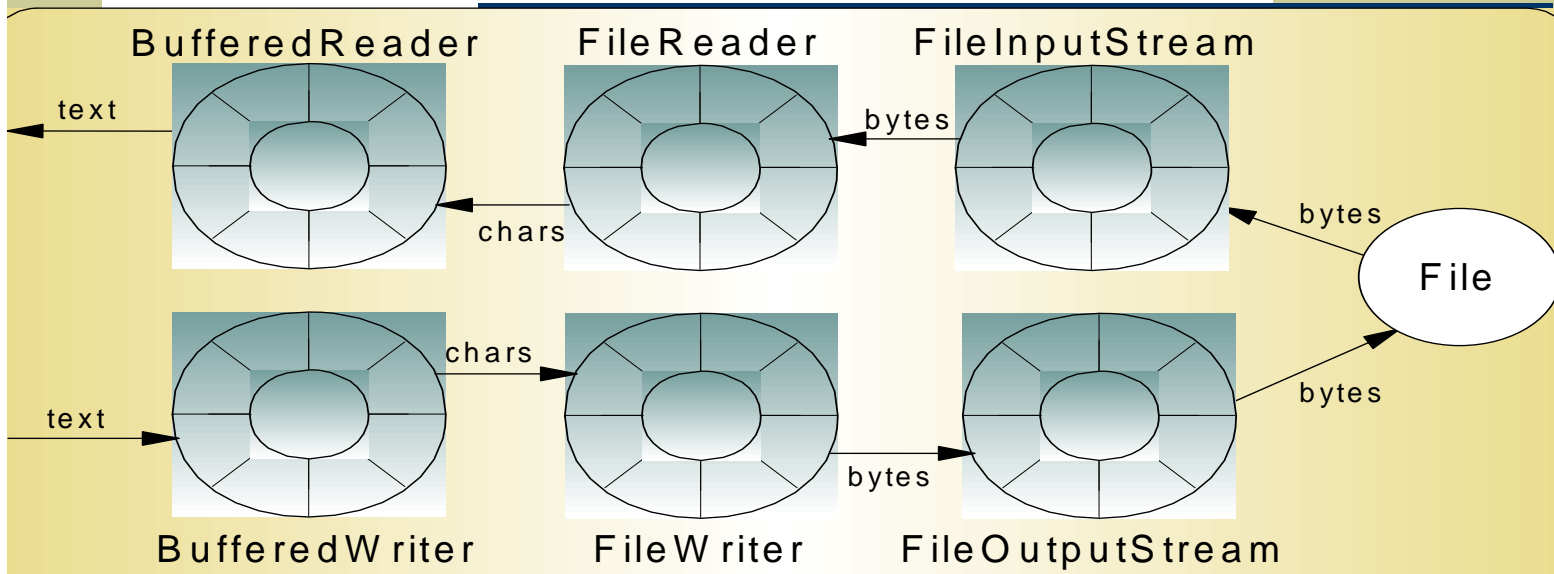
La clase Teclado

```
// continuación de la clase teclado
/** lee un numero entero desde la entrada estandar
 *
 * @return numero entero de tipo int
 *
 * @exception excepciones No lanza ningun tipo de excepcion de estrada/salida */
public static int leerInt() {
    int entero = 0;
    boolean error = false;
    do {    try {
            error = false;
            entero = Integer.valueOf(entrada.readLine()).intValue();
        } catch (NumberFormatException e1) {
            error = true;
            System.out.println("Error en el formato del numero, intentelo de nuevo."
        } catch (IOException e) {
            System.out.println("Error de E/S");
        }
    } while (error);
    return entero;
} } // final de la clase Teclado
```

Java

Entrada y Salida ²⁵

Ficheros de texto



Java

Entrada y Salida ²⁶

Ficheros de texto

❑ FileReader

❑ Util para leer ficheros de texto

- ❑ Constructor: `FileReader(String nombreFichero)`

❑ FileWriter

❑ Util para escribir ficheros de texto

❑ Constructores

- ❑ `FileWriter(String nombreFichero)` -- reescribe
- ❑ `FileWriter(String nombreFichero, boolean añadirFinal)` -- añade

❑ PrintWriter

❑ Implementa un flujo de salida de caracteres

❑ Métodos de utilidad

- ❑ `print()`, `println()`, `close()`

Java

Entrada y Salida²⁷

Ejemplo

```
import java.io.*;

public class BufferedConsole {
    public static void main(java.lang.String[] args)
        throws java.io.IOException {
        BufferedReader br = new BufferedReader(
            new InputStreamReader(System.in));
        BufferedWriter bw = new BufferedWriter(
            new OutputStreamWriter(System.out));
        System.out.println("Dame unos caracteres o EXIT");
        String s = br.readLine();
        while (!s.equals("EXIT")) {
            bw.write("s: " + s);
            bw.newLine();
            s = br.readLine(); }
        bw.close();
        br.close(); } }
```

Java

Entrada y Salida²⁸

Ejemplo Ficheros de texto

```
t java.io.*;
c class FicheroTexto {
b public static void main(String args[]) {
try {          // escritura de datos
    PrintWriter salida = new PrintWriter( new BufferedWriter(
        new FileWriter("prueba.txt")));
    salida.println("en un lugar de la mancha de cuyo");
    salida.println("nombre no quiero acordarme");
    salida.close();
    // lectura de datos
    BufferedReader entrada = new BufferedReader(new FileReader("prueba.txt"));
    String s, s2 = new String();
    while((s = entrada.readLine()) != null)
        s2 = s2 + s + "\n";
    System.out.println("Texto leído:" + "\n" + s2);
    entrada.close();
} catch (java.io.IOException e) {}
}
```

Ficheros de texto con datos numéricos

```
import java.io.*;
public class FicheroTexto {
    public static void main(String args[]) {
        int i=5;
        float f=5.5f;
        try {          // escritura de datos
            PrintWriter salida = new PrintWriter( new BufferedWriter(
                new FileWriter("hola.txt")));
            salida.println("en un lugar de la mancha de cuyo");
            salida.println(i);
            salida.println("nombre no quiero acordarme");
            salida.println(f);
            salida.close();
        }
    }
}
```

Ficheros de texto con datos numéricos

```
// lectura de datos
    BufferedReader entrada = new BufferedReader(
FileReader("hola.txt"));
    String s, s2 = new String();
    while((s = entrada.readLine()) != null)
        s2 = s2 + s + "\n";
    System.out.println("Texto leído:" + "\n" + s2);
    entrada.close();
} catch (java.io.IOException
{System.out.println("excepcion");}
}}
```

Ficheros

❑ Clase *File*

- ❑ Representa un nombre de ruta a un fichero o a un subdirectorio del disco
- ❑ Constructores
 - ❑ File(String ruta)
 - ❑ File(String ruta, String nombre)
 - ❑ File(File directorio, String nombre)

Ficheros

❑ Métodos

- ❑ `canRead()` comprueba si el fichero se puede leer
- ❑ `canWrite()` comprueba si el fichero se puede escribir
- ❑ `delete()` borra dicho fichero
- ❑ `getPath()` devuelve la ruta del fichero
- ❑ `mkdir()` crea un directorio con la ruta del objeto que lo recibe
- ❑ `isDirectory()` comprueba si dicho fichero es un directorio

❑ Constructores de otras clases

- ❑ `FileReader(File fichero)`, `FileWriter(File fichero)`

Ejemplo - copia de ficheros

```
rt java.io.*;

ic class CopiaFicheros {
public static void main(String[] args) throws IOException {
    File ficheroEntrada = new File("original.txt");
    File ficheroSalida = new File("copia.txt");
    FileReader entrada = new FileReader(ficheroEntrada);
    FileWriter salida = new FileWriter(ficheroSalida);
    int dato;
    while ((dato = entrada.read()) != -1)
        salida.write(dato);
    entrada.close();
    salida.close();
}
```

Serialización

Object Output Stream Instance

writeObject()

Object Input Stream Instance

readObject()

Serializable

implements

Sample Object

Java

Entrada y Salida³⁵

Escritura de objeto a fichero

```
rt java.io.*;
s Main{
lic static void main(String[] args){
y {SampleObject originalObj1 = new SampleObject();
    SampleObject originalObj2 = new SampleObject();
    originalObj1.setName("Mary Popins");
    originalObj1.setAge(32);
    originalObj2.setName("Popeye");
    originalObj2.setAge(42);
    FileOutputStream fos = null;
    ObjectOutputStream oos = null;
    fos = new FileOutputStream("SerializedObj.obj");
    oos = new ObjectOutputStream(fos);
    oos.writeObject(originalObj1);
    oos.writeObject(originalObj2);
    oos.flush();
    oos.close();
tch(Exception e){System.out.println("Main: main(): " + e);
    } }}
```

Lectura de fichero a objeto

```
rt java.io.*;

s Main{
lic static void main(String[] args){
y {
ileInputStream fis = null;
objectInputStream ois = null;
is = new FileInputStream("SerializedObj.obj");
is = new ObjectInputStream(fis);
ampleObject newObj1 = (SampleObject) ois.readObject();
ampleObject newObj2 = (SampleObject) ois.readObject();
is.close();
ystem.out.println("SampleObject1 name: " + newObj1.getName());
ystem.out.println("SampleObject2 name: " + newObj2.getName());
tch(Exception e){
ystem.out.println("Main: main(): " + e);
```