1️⃣

Manejo de ficheros

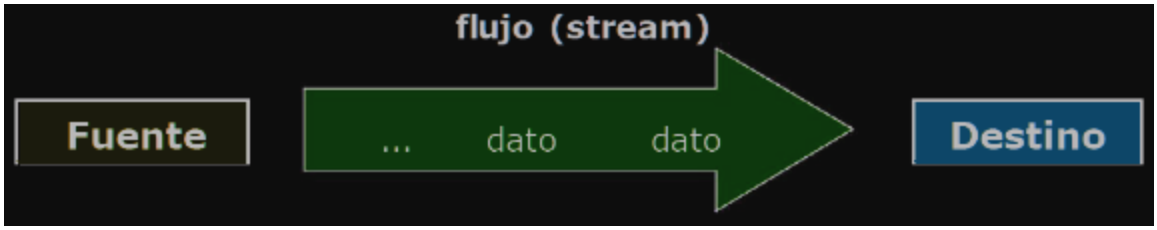
▼ Entrada y salida con flujos (streams)

En java se define la abstracción de stream (flujo) para tratar la comunicación de información entre el programa y el exterior.

- Entre una fuente y un destino fluye una secuencia de datos.

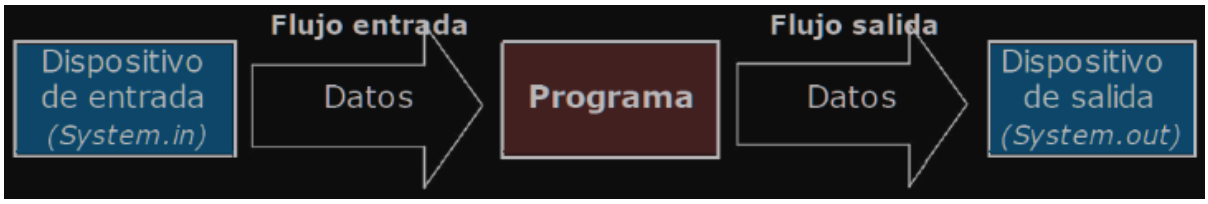
Los flujos actúan como interfaz con el dispositivo o clase asociada.

- Operación independiente del tipo de datos y del dispositivo
- Mayor flexibilidad (p.e. redirección, combinación)
- Diversidad de dispositivos (fichero, pantalla, teclado, red, …)
- Diversidad de formas de comunicación
  - Modo de acceso: secuencial, aleatorio
  - Información intercambiada: binaria, caracteres, líneas



En java se accede a la entrada o salida estándar a través de campos estáticos de la clase `java.lang.System`.

- `System.in` → Implementa la entrada estándar
- `System.out` → Implementa la salida estándar
- `System.err` → Implementa la salida de error




```
public class Prueba {

  public static void main(String args[]) throws IOException {
    int c;
    int contador=0;
    // Se lee hasta encontrar el fin de línea
    while((c=System.in.read())!='\n') {
      contador++;
      System.out.print((char)c);
    }
    // Se escribe el fin de línea
    System.out.println();
    System.err.println("Contados "+contador+"bytes en total.");
  }
}
```

Utilización de los flujos

Lectura

1. Abrir un flujo a una fuente de datos (creación del objeto stream)
   - Teclado
   - Fichero
   - Socket remoto
2. Mientras existan datos disponibles
   - Leer datos
3. Cerrar el flujo (método close)

Escritura

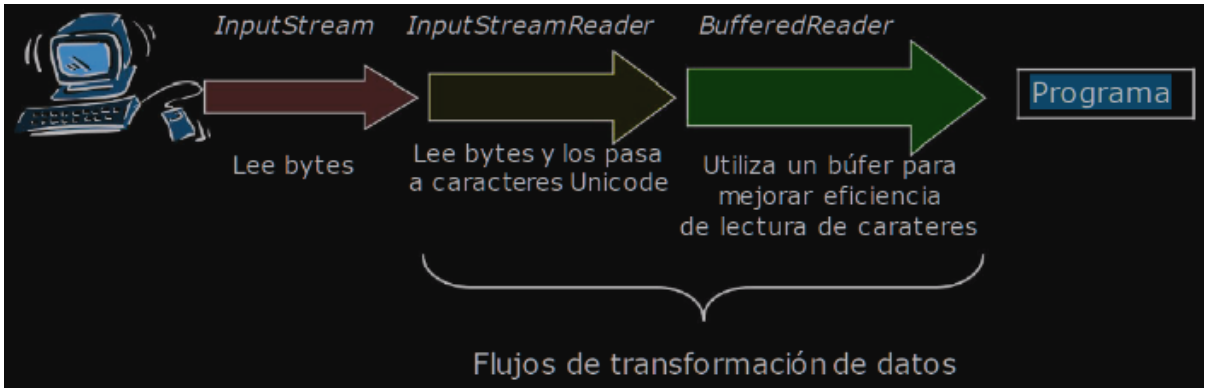
1. Abrir un flujo a una fuente de datos (creación del objeto stream)
   - Pantalla
   - Fichero

    - Socket local
2. Mientras existan datos disponibles
    - Escribir datos
3. Cerrar el flujo (método close)

- Para los flujos estándar ya se encarga el sistema de abrirlos y cerrarlos.
- Un fallo en cualquier punto produce la excepción `IOException`.

▼ Combinación de flujos

Los flujos se pueden combinar para obtener la funcionalidad deseada.




```
public static void main(String[] args) {
  InputStreamReader input=new InputStreamReader(System.in);
  BufferedReader entradaEstandar=new BufferedReader(input);
  String mensaje="";
  System.out.println("Introducir una línea de texto:");
  try {
    mensaje=entradaEstandar.readLine();
  } catch(IOException e) {
    // Excepción generada
  }
  System.out.println("Introducido: \""+mensaje+"\"");
}
```

▼ Ficheros

▼ Clase File

▼ Constructores

- `File(String ruta)`
- `File(String ruta,String nombre)`
- `File(File directorio,String nombre)`

▼ 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)`

```
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();
}
```

▼ Ficheros de texto

▼ FileReader

Para leer de ficheros de texto.

Hereda de `InputStreamReader`, que hereda de Reader.

Constructor → `FileReader(String nombreFichero)`

▼ FileWriter

Para escribir en ficheros de texto.

Hereda de `OutputStreamReader`, que hereda de Writer.

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()`

```
// Entrada de texto desde un fichero
public static void main(String args[]) {
  try {
    FileReader fr=new FileReader("nombrefichero");
    BufferedReader reader=new BufferedReader(fr);
    String linea=reader.readLine();
    while(linea!=null) {
      // Procesar el texto de la línea
      linea=reader.readLine();
      System.out.println(linea);
    }
    reader.close();
  } catch(FileNotFoundException e) {
    // No se encontró el fichero
  } catch(IOException e) {
    // Algo fue mal al leer o cerrar el fichero
  }
}
```

```
public static void main( String args[] ) {
  try {
    FileReader fr = new FileReader("nombrefichero");
    BufferedReader reader = new BufferedReader(fr);
    String linea = "";
    while(reader.ready()) {
```

```
      // procesar el texto de la línea
      linea = reader.readLine();
      System.out.println(linea);
    }
    reader.close();
  } catch(FileNotFoundException e) {
    // no se encontró el fichero
  } catch(IOException e) {
    // algo fue mal al leer o cerrar el fichero
  }
}
```