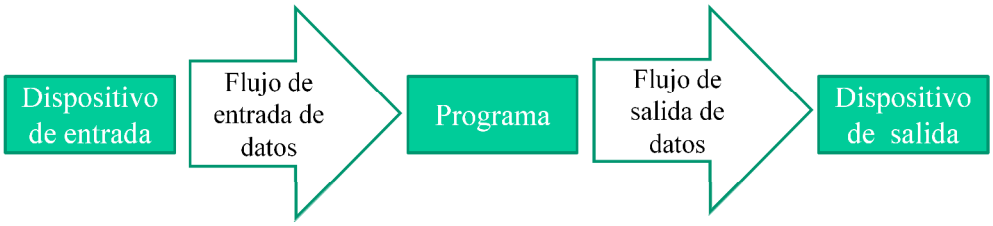
UD15.- Flujos de datos

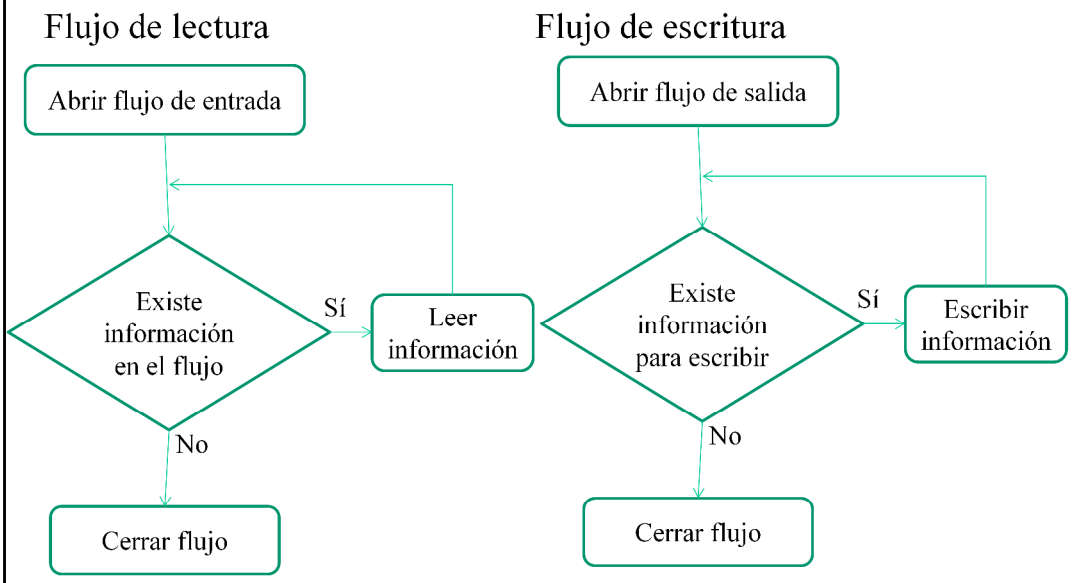
1. Introducción.
2. Jerarquía de flujos.
3. Flujos estándar.
   A. Flujos estándar de salida y salida de error.
   B. Flujos estándar de entrada.
4. Datos con formato.

1. Introducción

- Intercambio de datos entre el programa y el exterior
  - ✓ Diferentes 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



1. Introducción



2. Jerarquía de flujos

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

InputStream/OutputStream

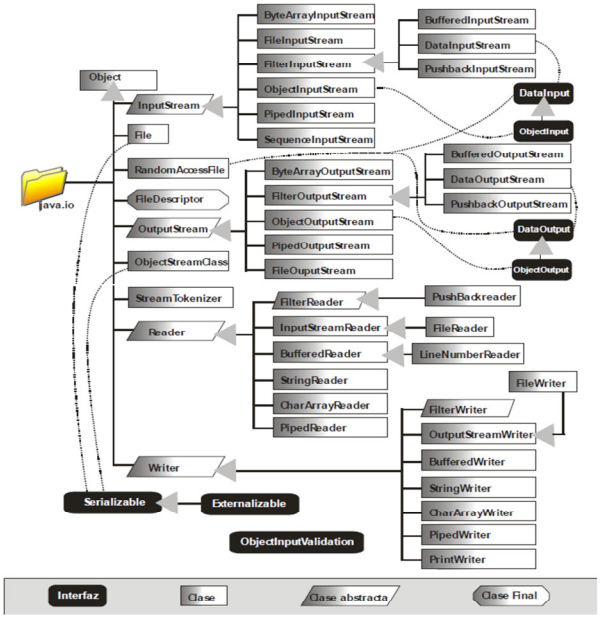
Estas dos clases **abstractas,** definen las funciones básicas de lectura y escritura de una secuencia de **bytes** pura (sin estructurar). Estos flujos de bits, no representan ni textos ni objetos, sino datos binarios puros. Tienen muchas subclases; de hecho casi todas las clases preparadas para la lectura y la escritura, derivan de estas.

Los métodos más importantes son **read (leer) y write (escribir),** que sirven para leer un byte del dispositivo de entrada o escribir un byte respectivamente.

Reader/Writer

Clases abstractas que definen las funciones básicas de escritura y lectura basada en texto Unicode. Se dice que estas clases pertenecen a la jerarquía de lectura/escritura orientada a **caracteres**, mientras que las anteriores pertenecen a la jerarquía orientada a bytes.

2. Jerarquía de flujos



3. Flujos estándar

- Entrada estándar : habitualmente el teclado
- Salida estándar : habitualmente la pantalla
- Salida de error estándar: habitualmente la pantalla

Los flujos estándares los abre y los cierra la JVM. Cuando la JVM va a ejecutar una App, abre los tres flujos estándares antes de empezar a ejecutar el main. Y cuando termina la ejecución del main, se encarga de cerrarlos.

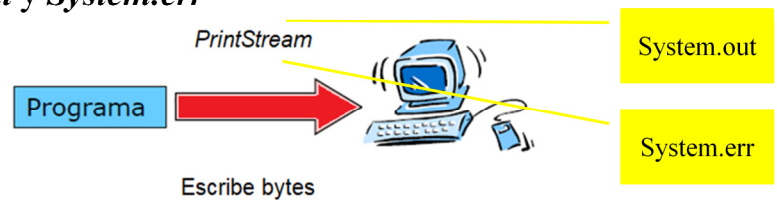
Java tiene acceso a la entrada/salida estándar a través de la clase **java.lang.System**. Los flujos estándar son campos estáticos de System

Fields

Modifier and Type	Field and Description
`static PrintStream`	`err` The "standard" error output stream.
`static InputStream`	`in` The "standard" input stream.
`static PrintStream`	`out` The "standard" output stream.

3.A. Flujos estándar salida y salida error.

❖ ***System.out*** y ***System.err***



Instancia de la clase **PrintStream** (flujo de bytes de salida)

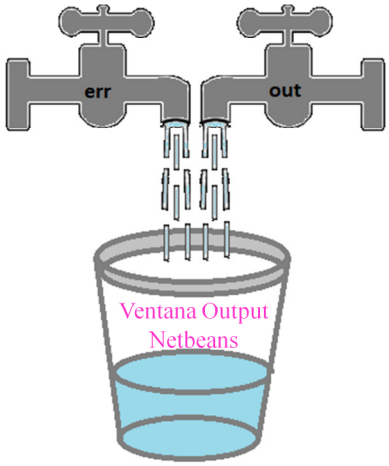
- Algunos métodos de impresión de datos
  - print() → escribe en el buffer
  - println() → escribe en el buffer y flush. Deja el cursor en la siguiente línea.
  - flush() → vacía el buffer de salida escribiendo su contenido en el dispositivo

3.A. Flujos estándar salida y salida error.

```
public class Salida {
    public static void  main (String[] args){
     //Entorno:
          int var;
   //Algoritmo:
          var=5;
          System.out.print(var);
          System.out.println(var);
          System.out.print("hola");
          System.out.flush();
          System.err.println("ERROR 1");
          System.err.println("Otro ERROR");
   }//Fin Programa
}
```

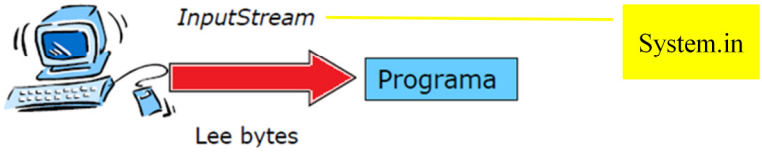
```
run:
55
holaERROR 1
Otro ERROR
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
run:
55
ERROR 1
Otro ERROR
holaBUILD SUCCESSFUL (total time: 0 seconds)
```



3.B. Flujos estándar de entrada. (bytes)

❖ ***System.in***



Instancia de la clase **InputStream** (flujo de bytes de entrada)

- Algunos métodos
  - **read()** → permite leer un byte de la entrada como entero
  - **skip(n)** → ignora n bytes de la entrada
  - **available()** → una **estimación** del número de bytes disponibles para leer en la entrada

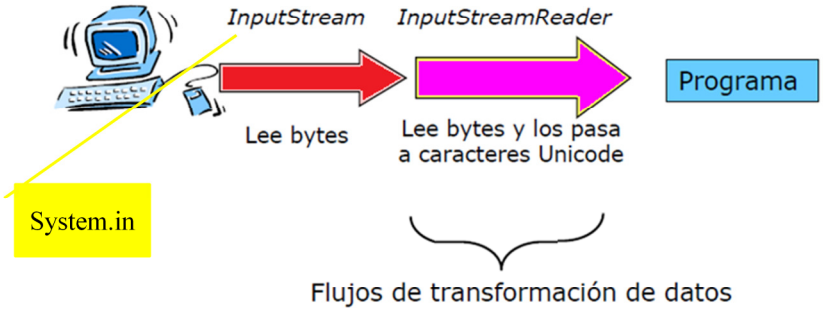
3.B. Flujos estándar de entrada. (bytes)

```
public class LeeBytes {

    public static void main(String[] args) {
        //Entorno:
        int c;
        int contador = 0;
        //Algoritmo:
        try {
            // se lee hasta encontrar el fin de línea
            c = System.in.read();
            while (c != '\n') {
                contador++;
                System.out.print((char) c);
                c = System.in.read();
            }//Fin Mientras
            System.out.println(); // Se escribe el cambio de línea
            System.out.println("Contados " + contador + " bytes en total.");
        } catch (IOException ioe) {
            System.err.println("Error de E/S.");
        }//Fin Try Catch
    }//Fin Programa
}
```

Leer bytes por teclado hasta la introducción del fin de línea \n

3.B. Flujos estándar de entrada. (caracteres)



Constructors

Constructor and Description
`InputStreamReader(InputStream in)` Creates an InputStreamReader that uses the default charset.

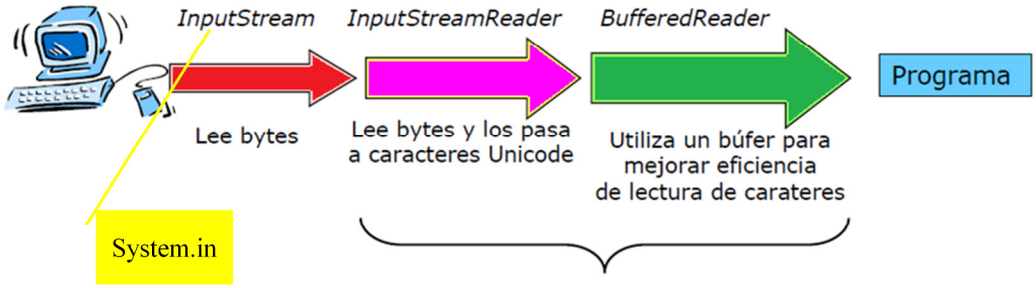
3.B. Flujos estándar de entrada. (caracteres)

```
public class LeeCaracter {
    public static void main(String[] args) {
    //Entorno:
        int c;
        int contador;
        InputStreamReader tecladoCaracter;
    //Algoritmo:
        tecladoCaracter = new InputStreamReader(System.in);
        contador = 0;
        try {
            // se lee hasta encontrar el fin de línea
            c = tecladoCaracter.read();
            while (c != '\n') {
                contador++;
                System.out.print((char) c);
                c = tecladoCaracter.read();
            }//Fin Mientras
            System.out.println(); // Se escribe el cambio de línea
            System.err.println("Contados " + contador + " caracteres en total.");
        } catch (IOException ioe) {
            System.err.println("Error de E/S.");
        }//Fin try catch
    }//Fin Programa
}
```

InputStream

InputStreamReader

3.B. Flujos estándar de entrada. (líneas)

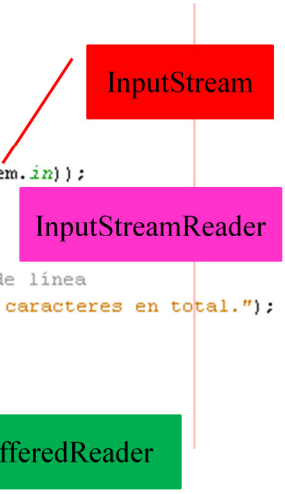


Constructors

Constructor and Description
`BufferedReader(Reader in)` Creates a buffering character-input stream that uses a default-sized input buffer.

3.B. Flujos estándar de entrada. (líneas)

```
public class LeeLinea {
    public static void main(String[] args) {
    //Entorno:
        String linea;
        BufferedReader teclado;
    //Algoritmo:
        teclado = new BufferedReader(new InputStreamReader(System.in));
        try {
            // se lee hasta encontrar el fin de línea
            linea = teclado.readLine();
            System.out.println(linea); // Se escribe el cambio de línea
            System.err.println("Contados " + linea.length() + " caracteres en total.");
        } catch (IOException ioe) {
            System.err.println("Error de E/S.");
        }
    }//Fin Programa
}
```



4. Datos con formato

Para construir una cadena con los datos con un determinado formato se usa

java.lang

Class String

`static String`	**`format(Locale l, String format, Object... args)`** Returns a formatted string using the specified locale, format string, and arguments.
`static String`	**`format(String format, Object... args)`** Returns a formatted string using the specified format string and arguments.

Si queremos mostrar en pantalla directamente los datos con formato

`PrintStream`	**`printf(Locale l, String format, Object... args)`** A convenience method to write a formatted string to this output stream using the specified format string and arguments.
`PrintStream`	**`printf(String format, Object... args)`** A convenience method to write a formatted string to this output stream using the specified format string and arguments.

4. Datos con formato

```
nombre = "Nuria  ";
apellido = "Fuentes";
resultado = String.format("%s %s", nombre, apellido);
System.out.println(resultado);

//Enteros: %d
resultado = String.format("%d * %d - %d", 10, 20, 30);
System.out.println(resultado);

//Reales: %f 6 decimales %.nf n=número de decimales
pi = 3.14159265359F;

resultado = String.format("%f - Decimales: %.2f", pi,pi);
System.out.println(resultado);
//%posición$tipo
resultado=String.format("%1$f - Decimales: %1$.4f", pi);
System.out.println(resultado);

//Booleano: %b
cantidad = 10;
esMayorQue5 = cantidad > 5;
resultado = String.format("La cantidad es mayor a %d : %b",
                          cantidad, esMayorQue5);
System.out.println(resultado);
resultado = String.format("%1f - Decimales: %1$.2f", pi);
System.out.println(resultado);
//Fecha: %t
System.out.println(String.format("%1$TA",new GregorianCalendar()));
System.out.println(String.format("%1$td/%1$tm/%1$tY", new GregorianCalendar())
```