Programación

Lectura y Escritura de la Información. Flujos

Esta presentación ha sido elaborada a partir del material del módulo de Programación del IES El Majuelo (Gines)

ÍNDICE

1. Flujos (*Streams*)

- Cualquier programa que vaya a realizar alguna entrada de datos o salida de información necesita usar un `Stream`.
- Representa la conexión o vía de comunicación entre el programa y el dispositivo de entrada o de salida con el que se establece el flujo de datos.
- De dicha conexión se encarga JAVA gracias al uso de diferentes clases.
  - Esto es lo que hace que la entrada o salida se realicen de la misma forma independientemente del dispositivo físico al que estén conectados.

1. Flujos (*Streams*)
1.1. Tipos de flujos

- El lenguaje JAVA distingue básicamente dos tipos de flujo:
  - Flujo de caracteres (de texto).
  - Flujo de *bytes* (binario).
- Todas las clases que ofrece JAVA para gestionar la E/S de datos se pueden clasificar en dos grupos:
  - Las que operan con bytes. Implementan las clases abstractas **`InputStream`** y **`OutputStream`**.
  - Las que operan con caracteres. Implementan las clases abstractas **`Reader`** y **`Writer`**.
- Todas las operaciones de E/S se realizan en *bytes*. Las clases **`InputStreamReader`** y **`OutputStreamWriter`** hacen posible la conversión entre *byte* y carácter:

```
InputStreamReader via = new InputStreamReader(System.in);
```

1. Flujos (*Streams*)
Flujos predefinidos

- En JAVA, la entrada desde el teclado y la salida por la pantalla están gestionadas por la clase **`System`**. Esta clase pertenece al paquete **`java.lang`**.

- Dicha clase tiene tres atributos `public` y `static` que son los llamados *flujos predefinidos*:
  - **`System.in`**: Entrada estándar de datos.
  - **`System.out`**: Salida estándar de datos.
  - **`System.err`**: Salida estándar de información de errores (salen en rojo).

1. Flujos (*Streams*)
1.2. Clases relativas a flujos

- Jerarquía de clases relativas a flujos:
  - Clases que manejan caracteres:
    - Clase abstracta **`Reader`**.
    - Clase abstracta **`Writer`**.
  - Clases que manejan *bytes*:
    - Clase abstracta **`InputStream`**.
    - Clase abstracta **`OutputStream`**.

1. Flujos (*Streams*)
1.2. Clases relativas a flujos

Clase abstracta **`Reader`**:

- `BufferedReader` (`LineNumberReader`)
- `CharArrayReader`
- `FilterReader` (`PushbackReader`)
- `InputStreamReader` (`FileReader`)
- `PipedReader`
- `StringReader`

Clase abstracta **`Writer`**:

- `BufferedWriter` (`LineNumberWriter`)
- `CharArrayWriter`
- `FilterWriter`
- `OutputStreamWriter` (`FileWriter`)
- `PipedWriter`
- `PrintWriter`
- `StringWriter`

1. Flujos (*Streams*)
1.2. Clases relativas a flujos

Clase abstracta `InputStream`:

- `ByteArrayInputStream`
- `FileInputStream`
- `FilterInputStream` (`BufferedInputStream`)
- `ObjectInputStream`
- `PipedInputStream`

Clase abstracta `OutputStream`:

- `ByteArrayOutputStream`
- `FileOutputStream`
- `FilterOutputStream` (`BufferedOutputStream`)
- `ObjectOutputStream`
- `PipedOutputStream`

2. Entrada / Salida estándar
2.1. Entrada desde el teclado

- La entrada estándar de datos desde el teclado se hace con la clase **`System`**, usando la propiedad **`in`**.
- Dicha propiedad tiene una serie de métodos, donde el más importante es:

```
int read();
```

- **`System.in.read()`:**
  - Lee un solo *byte* y lo devuelve en código ASCII.
  - Devuelve -1 cuando no hay más *bytes* que leer.

2. Entrada / Salida estándar
2.2. Salida por pantalla

- La entrada estándar de datos desde el teclado se hace con la clase **`System`**, usando la propiedad **`out`**.
- Dicha propiedad tiene una serie de métodos, donde los más importantes son:
  - `System.out.print (…)`
  - `System.out.println (…)`
  - `System.out.printf(…)`