UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica etsinf

Tema 4. Entrada y salida. Flujos y Ficheros

Programación (PRG)

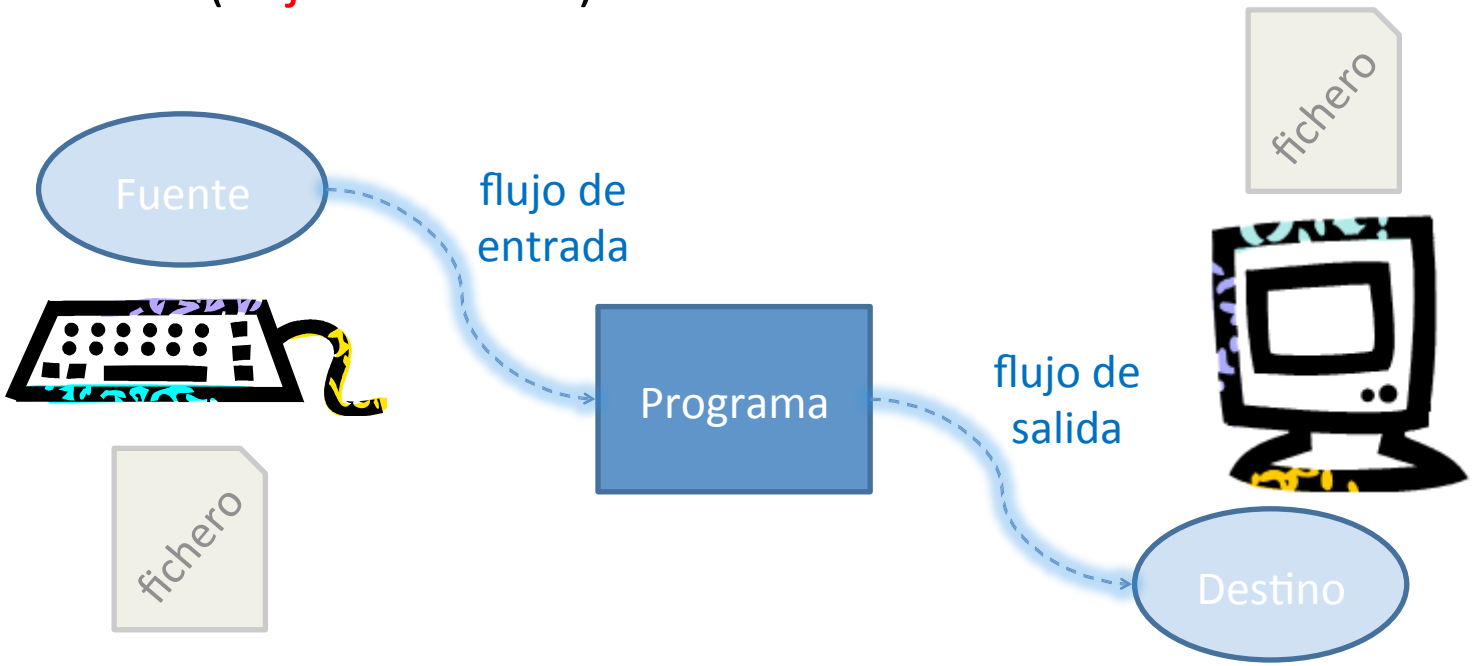
Departamento de Sistemas Informáticos y Computación

Contenidos

1. Introducción: Flujos y Ficheros
2. Ficheros de texto
3. Ficheros binarios

Introducción: Flujos y Ficheros

- En Java la entrada/salida se realiza utilizando flujos (streams), que son secuencias de información que tienen una fuente (flujos de entrada) o un destino (flujos de salida).



Introducción: Ficheros

- Un fichero es un conjunto de bits guardado en un dispositivo de almacenamiento secundario (disco duro, USB stick, etc.).
  - Permite almacenar los datos existentes en memoria de un programa para ser utilizados posteriormente (por ese u otro programa)
- Características principales:
  - Nombre (i.e. fichero.txt).
  - Ruta en el dispositivo de almacenamiento (i.e. /home/lucas/docs/file.txt).
  - Tamaño, tipicamente expresado en bytes (Kb, Mbytes, Gbytes, etc.).
- Características adicionales:
  - Permisos de acceso (dependientes del sistema de archivos).
  - Fecha de última modificación.
  - …
- Acciones principales sobre ficheros:
  - Abrir, Leer, Cerrar.
  - Abrir, Escribir, Cerrar.

Tipos de Ficheros

- En general, distinguimos entre dos tipos de ficheros:

Ficheros de Texto	Ficheros Binarios

• Secuencia de caracteres.
• Interpretable por un ser humano.
• Generalmente portable (legible en diferentes equipos).
• Escritura/lectura menos eficiente que los ficheros binarios.
• Requiere más tamaño que un fichero binario para representar la misma información.
  • Ej. Un entero de 10 dígitos en un fichero de texto ocuparía 10 bytes (asumiendo codificación ASCII de 1 byte/carácter).
	• Secuencia de bytes interpretables como valores (primitivos u objetos).
	• No interpretable por un ser humano.
	• Generalmente no portable (debe ser leído en el mismo tipo de ordenador y con el mismo lenguaje de programación que fue escrito).
	• Escritura/lectura eficiente.
	• Almacenamiento eficiente de la información.
	  • Ej. Un entero de 10 dígitos en un fichero binario ocupa 4 bytes.

- En Java, los ficheros binarios SÍ son independientes de la plataforma (pueden ser leídos en diferentes plataformas), eso sí, desde Java.

Funcionalidad de Ficheros en Java

- Java permite:
  1. Interactuar con el sistema de archivos independientemente de su tipo (FAT32, NTFS, EXT3, etc.) y del S.O. (Windows, Linux, OS X, etc.).
  2. Crear y consumir ficheros de texto (compuestos tanto por las representaciones en caracteres de los tipos primitivos como por Strings).
  3. Producir y leer ficheros binarios compuestos por datos de tipos primitivos y Strings.
  4. Trabajar con ficheros binarios compuestos por objetos (serialización de objetos).
  5. Manipular ficheros binarios a bajo nivel (E/S a nivel de bytes).
- En este tema nos centraremos en las funcionalidades de los Puntos 1 al 4.

Acceso al Sistema de Ficheros

- Utilizando la clase java.io.File:
  - Permite representar tanto ficheros como directorios en Java.

Método / Constructor	Descripción
File(String pathname)	Crea un nuevo objeto de tipo File a partir de su ruta
boolean **delete**()	Borra el fichero/directorio
boolean **exists**()	Indica si el fichero/directorio existe
String **getName**()	Devuelve el nombre del fichero/directorio (sin la ruta)
String **getParent**()	Devuelve la ruta al fichero/directorio (sin el nombre)
long **length**()	Devuelve el tamaño del fichero. No válido para directorios
File[] **listFiles**()	Obtiene un listado de ficheros en el directorio
boolean **isDirectory**()	Indica si se trata de un directorio
…	

Crear un objeto File no afecta al sistema de archivos hasta que no se invocan métodos de la clase.

Uso de la Clase File

- Ejemplo de uso de la clase File para tratar con el sistema de archivos:

```
import java.io.*;
class TestFile {
 public static void main(String[] args){
  File f = new File("/home/plopez/file.txt");
  if (f.exists()) System.out.println("El fichero existe!");
  else System.err.println("El fichero NO existe!");

  System.out.println("getName(): " + f.getName());
  System.out.println("getParent(): " + f.getParent());
  System.out.println("length(): " + f.length());
  }
}
```

- getName() obtendría “file.txt” y getParent() obtiene “/home/plopez”
- Las rutas en Windows son del estilo:
  - String path = "C:\\Documents and Settings\\Manoj\\Desktop";