Tema 2

FICHEROS

Tema 2
Ficheros

1. Persistencia de datos en ficheros
2. Tipos de ficheros según contenido
3. Codificaciones para texto

Clase file

Objetivos

- Gestión de ficheros de texto y ficheros binarios
- Acceso secuencial y aleatorio a ficheros
- Clases de Java para el manejo de ficheros
- Mecanismos de aceleración para operaciones Lectura/Escritura
- Acceso rápido a ficheros
- Organizaciones de ficheros
- Persistencia de datos en ficheros

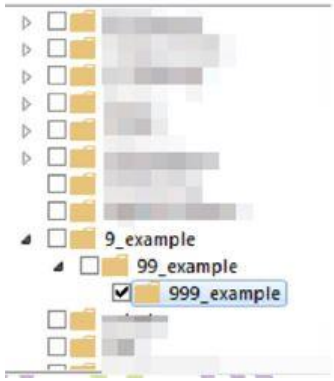
1. Persistencia de datos en ficheros

Fichero :

- Medio almacenamiento de información más elemental
- Palabra fichero => fichas libro biblioteca
  - Almacenamiento digital de la información en papel
- Ficheros de longitud fija
- Ficheros auxiliares de índice para acelerar las búsquedas
- Acceso secuencial : método de acceso de COBOL
- Actualidad:
  - Uso de ficheros XML
  - El sistema de correo como almacenamiento de datos
  - Copias de seguridad
  - Sistema de exportación/importación de datos CSV

2.- Tipos de ficheros según contenido

- Fichero
  - Secuencia de bytes que almacena cualquier tipo de información
  - Tiene nombre
  - Tiene ubicación dentro de la jerarquía de directorios



2.- Tipos de ficheros según contenido

- Tipos
  - Ficheros Texto
    - Formados únicamente por secuencia de caracteres
      - Visibles
      - Espacios en blanco
      - Separadores
  - Ficheros binarios
    - Resto de los ficheros
    - Necesitan programas espaciales para abrirlos
    - Almacenan ceros y unos

3.- Codificaciones para texto

- Texto
  - Secuencia de caracteres almacenados como secuencia de bytes
    - En memoria
    - En dispositivo de almacenamiento
  - Codificación:
    - Método de transformación de texto a secuencia de bytes valida
    - UTF-8 sistema de codificación ASCII de la información
      - Formato de codificación UNICODE
        - Estándar de codificación universal de caracteres
        - Proporciona una base de procesamiento, almacenamiento e intercambio de información
        - Cubre todos los caracteres especiales de todos los sistemas de escritura del mundo
      - Divide los caracteres en grupos en función del numero de bytes que use (1 a 4 bytes)
        - 1 byte : código ASCII
        - 2 bytes: incluye caracteres alfabeto griego hebrero árabe, signos diacríticos
        - 3 bytes : texto plano básico multilingüe, chino japonés el coreano
        - 4 bytes : símbolos matemáticos y alfabetos clásicos persa el fenicio el lineal etc

Clase *File* de java

- Java SE 8
- Paquete Java.io
- Clase File :
  - Nos permite obtener información relativa a directorio y ficheros dentro del sistema de ficheros
  - Permite operaciones de CRUD
    - Crear (Create)
    - Leer (Read)
    - Actualizar (Update)
    - Eliminar (Delete)

Clase *File* de java

- Constructores:
  - public File(String nombreFichero | path);
  - public File(String path, String nombreFichero | path);
  - public File(File path, String nombreFichero | path);

Clase *File* de java

Ejemplos utilizando el **primer** constructor:

1. *Crea un Objeto File asociado al fichero personas.dat que se encuentra en el directorio de trabajo:*
   - **File f = new File("personas.dat");**
     - En este caso no se indica path. Se supone que el fichero se encuentra en el directorio actual de trabajo.
2. *Crea un Objeto File asociado al fichero personas.dat que se encuentra en el directorio ficheros dentro del directorio actual.*
   - **File f = new File("ficheros/personas.dat");**
     - En este caso se indica la ruta relativa tomando como base el directorio actual de trabajo.
       Se supone que el fichero personas.dat se encuentra en el directorio ficheros.
       A su vez el directorio ficheros se encuentra dentro del directorio actual de trabajo.
3. *Crea un Objeto File asociado al fichero personas.dat dando la ruta absoluta:*
   - **File f = new File("c:/ficheros/personas.dat");**
     - El fichero se encuentra en el directorio ficheros. A su vez el directorio ficheros se encuentra en la raíz de la unidad C:
     - Si se omite la letra de la unidad, por defecto se asume la letra de la unidad en la que se encuentra el proyecto:
     - File f = new File("/ficheros/personas.dat");

Clase *File* de java

Ejemplos utilizando el **segundo** constructor:

En este caso se crea un objeto File cuya ruta (absoluta o relativa) se indica en el primer String.

1. Crea un Objeto File asociado al fichero personas.dat que se encuentra en el directorio ficheros dentro del directorio actual.
   - **File f = new File("ficheros", "personas.dat");**
     - En este caso se indica la ruta relativa tomando como base el directorio actual de trabajo.

2. Crea un Objeto File asociado al fichero personas.dat dando la ruta absoluta:
   - **File f = new File("/ficheros", "personas.dat");**
     - En este caso se indica la ruta absoluta, indicada por la barra del principio.

Clase *File* de java

Ejemplos utilizando el **tercer** constructor:

Este constructor permite crear un objeto File cuya ruta se indica a través de otro objeto File.

1. Crea un Objeto File asociado al fichero personas.dat

que se encuentra en el directorio ficheros dentro del directorio actual.

- **File ruta = new File("ficheros");**
- **File f = new File(ruta, "personas.dat");**

2. Crea un Objeto File asociado al fichero personas.dat dando la ruta absoluta:

- File ruta = new File("/ficheros");
- **File f = new File(ruta, "personas.dat");**
- Debemos tener en cuenta que crear un objeto File no significa que deba existir el fichero o el directorio o que el path sea correcto.
- Si no existen no se lanzará ningún tipo de excepción ni tampoco serán creados.

Clase *File* de java

- Métodos

MÉTODO	DESCRIPCIÓN
boolean canRead()	Devuelve true si se puede leer el fichero
boolean canWrite()	Devuelve true si se puede escribir en el fichero
boolean createNewFile()	Crea el fichero asociado al objeto File. Devuelve true si se ha podido crear. Para poder crearlo el fichero no debe existir. Lanza una excepción del tipo IOException.
boolean delete()	Elimina el fichero o directorio. Si es un directorio debe estar vacío. Devuelve true si se ha podido eliminar.
boolean exists()	Devuelve true si el fichero o directorio existe
String getName()	Devuelve el nombre del fichero o directorio
String getAbsolutePath()	Devuelve la ruta absoluta asociada al objeto File.

Clase *File* de java

- Métodos

MÉTODO	DESCRIPCIÓN
String getCanonicalPath()	Devuelve la ruta única absoluta asociada al objeto File. Puede haber varias rutas absolutas asociadas a un File pero solo una única ruta canónica. Lanza una excepción del tipo IOException.
String getPath()	Devuelve la ruta con la que se creó el objeto File. Puede ser relativa o no.
String getParent()	Devuelve un String conteniendo el directorio padre del File. Devuelve null si no tiene directorio padre.
File getParentFile()	Devuelve un objeto File conteniendo el directorio padre del File. Devuelve null si no tiene directorio padre.
boolean isAbsolute()	Devuelve true si es una ruta absoluta
boolean isDirectory()	Devuelve true si es un directorio válido
boolean isFile()	Devuelve true si es un fichero válido
long lastModified()	Devuelve un valor en milisegundos que representa la última vez que se ha modificado (medido desde las 00:00:00 GMT, del 1 de Enero de 1970). Devuelve 0 si el fichero no existe o ha ocurrido un error.

Clase *File* de java

- Métodos

MÉTODO	DESCRIPCIÓN
long length()	Devuelve el tamaño en bytes del fichero. Devuelve 0 si no existe. Devuelve un valor indeterminado si es un directorio.
String[] list()	Devuelve un array de String con el nombre de los archivos y directorios que contiene el directorio indicado en el objeto File. Si no es un directorio devuelve null. Si el directorio está vacío devuelve un array vacío.
String[] list(FilenameFilter filtro)	Similar al anterior. Devuelve un array de String con el nombre de los archivos y directorios que contiene el directorio indicado en el objeto File que cumplen con el filtro indicado.
boolean mkdir()	Crea el directorio. Devuelve true si se ha podido crear.
boolean mkdirs()	Crea el directorio incluyendo los directorios no existentes especificados en la ruta padre del directorio a crear. Devuelve true si se ha creado el directorio y los directorios no existentes de la ruta padre.
boolean renameTo(File dest)	Cambia el nombre del fichero por el indicado en el parámetro dest. Devuelve true si se ha realizado el cambio.

Ejemplos

Ejemplo 1

- Mostrar la información de ficheros y directorios de una ruta determinada leída por teclado

Ejemplo 2

- A partir del directorio c:/ficheros y un fichero datos.txt que se encuentra en ese directorio, implementar le siguiente análisis
  - Si el directorio no existe se crea
  - Si el fichero no existe se crea
    - A continuación, se crea el fichero.
  - Si el fichero existe se muestra el tamaño del mismo.

Ejemplo 1

- Si suponemos que no existe el fichero ni el directorio la ejecución del programa produce la siguiente salida:
  - c:\ficheros\datos.txt
  - c:\ficheros
  - c:\ficheros
  - c:\
  - Fichero datos.txt no existe
  - El directorio ficheros no existe
  - Directorio creado
  - Fichero datos.txt creado

Ejemplo 1 : Clase *File*

- Si volvemos a ejecutar el programa después de crear la ruta y el fichero, se muestra:
  - c:\ficheros\datos.txt
  - c:\ficheros
  - c:\ficheros
  - c:\
  - Fichero datos.txt existe
  - Tamaño 0 bytes

Ejemplos

Ejemplo 3

- Un programa que elimina un fichero

Ejemplo 4

- Un programa que modifica el nombre de un fichero

Ejemplo 5

- Diferencia entre getPath(), getAbsolutePath() y getCanonicalPath().. Usamos la propiedad getProperty() de la clase System para obtener el directorio de trabajo