

Archivos

Creación, lectura y escritura.

Cátedra 11

Otoño 2005

1 El vendedor

Un vendedor desea crear una base de datos con los productos que vende, las unidades disponibles y sus precios, con el propósito de no vender productos que no están en bodega y no vender a precios equivocados. El lugar lógico donde almacenar esta base de datos es un archivo en disco, pues esta información no se puede tener permanentemente en la memoria del computador, porque se pierde cuando se apaga el computador su tamaño es limitado (típicamente unos 64 megabytes, que equivalen a $64 \times 1024 \times 10^3$ caracteres).

En cambio, el disco permite almacenar datos en forma permanente o persistente (no se borra al apagar el computador) masiva (típicamente unos 40 gigabytes, que equivalen a 40.000 megabytes).

La desventaja es que la manipulación de datos en disco es mucho más lenta que en la memoria del computador.

2 Archivos

Los computadores almacena la información permanente en dispositivos de almacenamiento, tales como discos duros, diskettes y discos compactos. La información almacenada puede ser muy variada: por ej; textos, aplicaciones (programas), imágenes o sonidos. Sin embargo, todo esta información se guarda en elementos llamados “archivos”. El tamaño de un archivo puede ir desde 0 bytes (un archivo vacío) hasta archivos de varios gigabytes. Dependiendo del sistema operativo, estos archivos se organizan en estructuras similares a árboles llamadas directorios. Cada archivo se identifica mediante un nombre.

3 Archivos de texto

Los archivos de texto son el tipo más simple de archivo. Están compuestos por líneas de texto. Cada línea contiene palabras (o símbolos) separadas por espacios. Uno puede representar un archivo de texto como un conjunto de líneas de ancho variable. Los archivos de texto se caracterizan por ser “planos”, es decir, todas las letras tienen el mismo formato y no hay palabras subrayadas, en negrita, o letras de distinto tamaño o ancho. Sólo letras.

Los archivos binarios, a diferencia de los archivos de texto, guardan información que no se puede representar por letras: imágenes o texto con formato.

Al abrir estos archivos con un editor de texto, el editor se llena de símbolos extraños, pues trata de interpretar la información dentro del archivo como si fueran letras. Las aplicaciones (ejecutables), también se almacenan en archivos, como secuencias de unos y ceros.

Tampoco puede mirarse su contenido con un editor de texto.

3.1 Escritura de archivos

En este curso crearemos archivos mediante la clase `con.printWriter` y podremos leer su contenido mediante la clase `BufferedReader`. Por ejemplo, consideremos un programa que crea un archivo con nombres de productos que ingresa el usuario. El diálogo entre programa y usuario será:

```
producto ? jabon Lux
pasta de dientes Odontine
producto ? ...
```

El programa debe escribir el archivo “productos.txt” de tal forma que su contenido sea el siguiente:

Productos.txt

```
jabón Lux
pasta de dientes Odontine
Champú Sedal
Flan Colón
```

El programa que realiza este diálogo y escribe el archivo es el siguiente:

```

PrintWriter escr;
escr= new PrintWriter(new FileWriter("producto.txt"));
con.print("Ingrese el nombre de un producto: ");
String producto=con.readLine();
while (!producto.equals("fin")) {
    escr.println(producto);
    con.print("Ingrese el nombre de un producto: ");
    producto=con.readLine();
}
escr.close();

```

En la primera línea el programa fabrica/construye un objeto escritor del archivo "productos.txt". Este objeto pertenece a la clase `con.printWriter`. Se indica (como argumento del constructor) el nombre del archivo que se va a escribir. El objeto construido queda referenciado por la variable `escr`, y con él se pueden realizar las siguientes operaciones:

3.2 Métodos `print()` y `println()`

print(...) escribe un dato en el archivo, de la misma forma que los procedimientos `con.print(...)` y `con.println(...)` despliegan datos en la consola. El objeto `escr` no escribe los datos de inmediato en el disco, si no que espera a formar un lote grande y luego escribirlos en conjunto. Esto resulta más rápido que escribirlos de a uno, porque escribir un byte en disco toma el mismo tiempo que escribir 4 kilobytes. Es totalmente análogo a escribir en la consola.

3.3 Método `close`

escr.close() ordena a `escr` concluir la escritura del archivo (si hay un lote pendiente lo escribe). Después de invocar `close()` no es posible volver a escribir con el objeto `escr`.

La única forma de escribir más datos en "productos.txt" sería construyendo un nuevo objeto `con.printWriter`.

4 Lectura de un archivo

El programa que despliegue en pantalla los productos almacenados en el archivo "productos.txt" es:

```
BufferedReader lect= new BufferedReader(new FileReader("productos.txt"));
```

```

String prod= lect.readLine();
while (prod!=null ) {
    con.println(prod);
    prod= lect.readLine();
}
lect.close();

```

En la primera línea se fabrica un objeto lector del archivo "productos.txt". El objeto es de la clase `BufferedReader` y queda referenciado por la variable `lect`. Con él se pueden realizar las siguientes operaciones:

4.1 Método `con.readLine`

`String con.readLine()` Si es la primera vez que se le invoca, devuelve la primera línea. Si no quedan datos en el archivo, se devuelve `null`. Es análoga a `con.readInt()`, solo que en vez de leer un entero lee toda la línea y la devuelve en un `String`.

4.2 Método `close`

Después de invocar esta operación no es válido seguir leyendo datos. En realidad esta operación no es muy útil en archivos de lectura y se puede omitir si uno es choro.

AL MARGEN

Como guardar datos en archivos de texto

Existen 2 formas básicas de guardar datos en un archivo de texto. La primera es separar los distintos campos usando un delimitador especial:

```

Da Silva:Lula:Brasil
Bush:George:Estados Unidos
Lagos:Ricardo:Chile

```

La segunda forma, consiste en utilizar un ancho fijo para cada campo:

```

Da Sila      Lula      Brasil
Bush        George    Estados Unidos
Lagos       Ricardo    Chile

```

En el primer caso se tiene que usar **indexOf** y **substring** para quebrar cada línea, en el segundo, basta con usar **substring**.

Problema propuesto

Escribe un programa que lea un archivo separado por limitadores y muestre sus campos en pantalla, haz luego lo mismo, pero con un archivo de campos de ancho fijo.