

## Archivos en disco: motivación

	Memoria (RAM)	Disco
<b>tipo de memoria</b>	volátil (contenido se pierde)	permanente (contenido se mantiene)
<b>velocidad</b>	rápida ( $\sim 10^{-6}$ segs)	lenta ( $\sim 10^{-3}$ segs)
<b>tiempo acceso</b>	constante (a cualquier lugar)	variable (depende de cercanía a cabezal)
<b>costo</b>	caro	barato ( $\sim 100$ veces -)
<b>capacidad</b>	limitada (máx 4G)	mayor (máx 500G)

**Problema 1.** Leer líneas que terminan con la palabra “fin” y grabarlas en un archivo en disco de nombre “archivo.txt”.

```
import java.io.*;
class CrearArchivo{
static public void main(String[]args)
throws IOException
{
    PrintWriter a=new PrintWriter(
        new FileWriter("archivo.txt"));
    while(true){
        String linea=U.readLine("linea(o fin)?");
        if( linea.equals("fin") ) break;
        a.println(linea);
    }
    a.close();
}
```

### Explicaciones

#### 1. import java.io.\*;

Inserta (deja disponibles) clases predefinidas para leer y escribir archivos

- clases PrintWriter y FileWriter para escribir archivos
- clases BufferedReader y FileReader para leer archivos

#### 2. throws IOException

indica que si se produce una excepción (“error”) de lectura/escritura (falla al leer o escribir), se muestra un mensaje y se aborta (termina) el programa

#### 3. PrintWriter a=new PrintWriter(...);

- a: objeto en memoria de clase PrintWriter que representa al archivo (y contiene sus principales características: ubicación, tamaño, cursor, etc)
- “archivo.txt”: nombre externo del archivo (en el disco).
  - El sufijo .txt se usa para archivos que contienen texto (caracteres).
- “abre” (prepara, inicializa) archivo para escritura
  - si archivo no existe, se crea, y si ya existe, se regraba
  - ubica el cursor del archivo al comienzo del espacio asignado (para grabación de información)

#### 4. a.println(linea)

escribe (graba) una línea en el archivo

- graba caracteres del string
- graba una marca de fin de línea (carácter especial *newline* o \n)
- ubica cursor después de *newline*

#### Ejemplo:

```
a.println("hola como estás");
```

escribe en el archivo en disco:

```
...hola como estás\n
                        ^ (cursor de archivo)
```

**Nota.** print: método que sólo graba caracteres (y no newline)

#### 5. a.close()

- “cierra” el archivo
  - graba marca de fin de archivo (eof)
  - libera recursos (memoria) asociada al objeto a
  - objeto a queda indefinido

ejemplo:

```
a.println("chao"); a.close();
```

```
...chao\nX
```

X: marca de fin de archivo (distinta a cualquier carácter)

**Problema 2.** Mostrar el archivo "archivo.txt"

```
import java.io.*;
class LeerArchivo{
static public void main(String[]args)
throws IOException
{
    BufferedReader a = new BufferedReader(
        new FileReader("archivo.txt"));
    while(true){
        String linea=a.readLine();
        if(linea==null) break;
        U.println(linea);
    }
    a.close();
}
}
```

**Solución 2.** usando "expresión idiomática" para leer hasta el final del archivo

```
import java.io.*;
class LeerArchivo{
static public void main(String[]args)
throws IOException
{
    BufferedReader a = new BufferedReader(
        new FileReader("archivo.txt"));

    String linea;
    while((linea=a.readLine())!=null)
        U.println(linea);

    a.close();
}
}
```

**Explicaciones**

**1. BufferedReader a=new BufferedReader(...);**

- a: objeto de clase BufferedReader que representa al archivo
- "abre" (prepara, inicializa) archivo para lectura
  - si archivo no existe, aborta
  - si existe, ubica el cursor al comienzo del archivo

**2. a.close()**

- "cierra" el archivo
- si se omite, se cierra al terminar programa

**3. a.readLine()**

- lee una línea del archivo
  - entrega un string con los caracteres de la línea (sin newline)
  - avanza cursor al comienzo de la línea siguiente

**Ejemplo:**

```
a.readLine();
```

lee (y entrega) una línea del archivo en disco:

```
hola como estás\n
                        ^ (cursor de archivo)
```

**Nota.** si detecta fin del archivo, es decir, si trata de leer cuando el cursor está apuntando a la marca de fin de archivo, entonces entrega el valor **null** (no el string "null" ni el string "")

**Problema 3.** Copiar un archivo en otro, obteniendo del usuario los nombres de los archivos

```
BufferedReader a=new BufferedReader(
    new FileReader(U.readLine("input?")));

PrintWriter b=new PrintWriter(
    new FileWriter(U.readLine("output?")));

String linea;

while((linea=a.readLine())!=null)
    b.println(linea);

b.close();
a.close();
```

**Problema 4.** mostrar las líneas de un archivo que contengan un string

```
BufferedReader a=new BufferedReader(
    new FileReader(U.readLine("archivo?")));

String linea, s=U.readLine("string?");

while((linea=a.readLine())!=null)

    if(linea.indexOf(s)>=0)
        U.println(linea);

a.close();
```

#### Ejercicio

//nº de líneas de archivo de nombre x

static public int líneas(String x)

...

}

//dividir el archivo "CC1001.txt" en dos mitades que deben grabarse

//en los archivos "CC1001A.txt" y "CC1001B.txt"

static public void main(String[] args) throws IOException{

...

}

**Nota.** Si el archivo tiene una cantidad impar de líneas, entonces la primera mitad debe contener una línea más que la segunda mitad.

//nº de líneas de archivo de nombre x

static public int líneas(String x)

throws IOException

{

BufferedReader a=new BufferedReader(  
new FileReader(x));

int n=0;

while((linea=a.readLine())!=null)

++n;

}

a.close();

return n;

}

static public void main(String[] args) throws IOException{

//obtener nº de líneas

int n=lines("CC1001");

//separar en dos mitades

BufferedReader a=new BufferedReader(  
new FileReader("CC1001.txt"));

PrintWriter

b=new PrintWriter(new FileWriter("CC1001A.txt")),

c=new PrintWriter(new FileWriter("CC1001B.txt"));

for(int i=1; (s=a.readLine())!=null; ++i)

if(i<=(n+1)/2)

b.println(s);

else

c.println(s);

b.close(); c.close(); a.close();

}

**abreviando:**

for(int i=1; (s=a.readLine())!=null; ++i)

(i<=(n+1)/2 ? b : c).println(s);

//separar en dos mitades (solución 2)

BufferedReader a=new BufferedReader(  
new FileReader("CC1001.txt"));

//copiar primera mitad

PrintWriter b=new PrintWriter(  
new FileWriter("CC1001A.txt"));

for(int i=1; i<=(n+1)/2; ++i)

b.println(a.readLine());

b.close();

//copiar segunda mitad

b=new PrintWriter(new FileWriter("CC1001B.txt"));

for(int i=1; i<=n/2; ++i)

b.println(a.readLine());

b.close();

a.close();

#### Lectura (desde teclado) y escritura (en pantalla)

##### ejemplos de uso:

String s = U.readLine("string?");

int i = U.readInt("Nº entero?");

double d = U.readDouble("Nº real?");

char c = U.readChar("caracter?");

U.println(s+" "+i+" "+d+" "+c);

##### //mostrar archivo

BufferedReader a=new BufferedReader(  
new FileReader(U.readLine("archivo?")));

String s;

while((s=a.readLine())!=null)

U.println(s);

a.close();

#### Métodos para leer strings, números y caracteres desde el teclado

import java.io.\*;

class U{

static public BufferedReader teclado =

new BufferedReader(new InputStreamReader(System.in));

//System.in: objeto predefinido (entrada estándar)

static public String readLine(String x) throws IOException{

print(x); return teclado.readLine();

}

static public int readInt(String x) throws IOException{

return Integer.parseInt(readLine(x)); //String a int

}

static public double readDouble(String x) throws IOException{

return Double.parseDouble(readLine(x)); //String a double

}

static public char readChar(String x) throws IOException{

return readLine(x).charAt(0); //primer caracter

}

static public void print(String x){System.out.print(x);}

static public void println(String x){System.out.println(x);}

//System.out: objeto predefinido (salida estándar)

...

}