# Ficheros

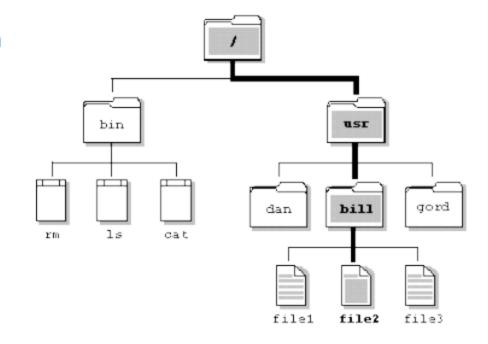
### El archivo

- Se considera un archivo (fichero) en informática a un conjunto de datos que se encuentran almacenados en un dispositivo
- Este conjunto de datos viene agrupado por un nombre, una ruta de acceso y una extensión
- El conjunto de los 3 elementos debe ser único
- Los ficheros se guardan en dispositivos de almacenamiento fijo como discos duros, pendrives, tarjetas, CD, etc.

### Sistema de ficheros

• Los ficheros se suelen organizar de forma jerárquica

- No pueden existir ficheros con el mismo:
  - Nombre
  - Ruta
  - Extensión



## Tipos de ficheros



**Fichero estándar:** Fichero que contiene cualquier tipo de datos. Documentos, imágenes, audio, vídeo, etc.



**Directorio o carpeta:** Fichero que contiene otros ficheros. Sirve para organizar de forma jerárquica los diferentes ficheros

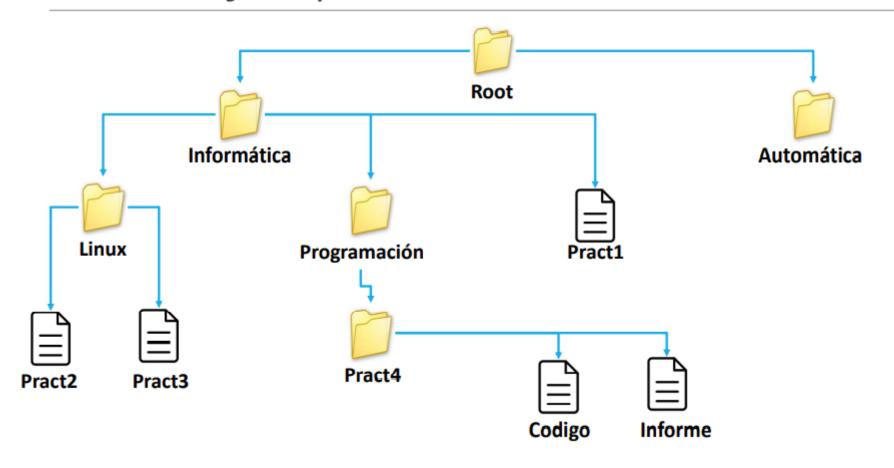


**Ficheros especiales:** Ficheros que sirven para controlar los diferentes periféricos conectados al ordenador

#### Rutas

- Para acceder a un determinado fichero, se utiliza la ruta (path)
- Una ruta indica la dirección del fichero en el sistema de archivos
- En una ruta, cada nivel de la jerarquía se representa delimitado por el símbolo /. En Windows, el símbolo separador es \
- · Además de /, existen 2 elementos especiales en la ruta
  - Representa al directorio actual
  - .. Representa al directorio padre en la jerarquía
- Existen 2 tipos de rutas:
  - Absoluta: Ruta al fichero desde el directorio principal (root). Ej: /home/Documentos/ejemplo.txt
  - Relativa: Ruta al fichero desde el directorio actual. Ej: Estando en home, ./Documentos/ejemplo.txt

# Rutas. Ejemplo de árbol de directorios



#### Extensión de un fichero

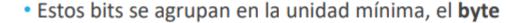
- · La extensión sirve para asociar un determinado fichero con un programa/s que lo ejecuten o interpreten
- Indica cómo debe ser interpretado el fichero



- · Se considera extensión lo existente a partir del último punto del nombre del archivo
  - Hola.doc → Extensión = doc
  - Hola.doc.mp3 → Extensión = mp3

#### El archivo

• En un ordenador, lo único que se es capaz de manejar es un conjunto de bits (0, 1)





 Podemos considerar a todo fichero en nuestro ordenador como un conjunto de bytes, independientemente de su extensión

• Interpretar el significado de estos bytes es responsabilidad del programa que manipula este fichero

## Operaciones con ficheros

- Apertura: El programa abre el fichero y se prepara para leerlo o escribirlo. Suele "reservar" el fichero para sí
- Cierre: Indica que se ha finalizado con las operaciones sobre el fichero. Libera el fichero
- **Lectura**: Lee el fichero o una de sus partes
- Escritura: Permite escribir en el fichero, ya sea añadiendo datos o sobrescribiendo los ya existentes
- **Ejecución**: Similar a la lectura, pero utiliza los datos del fichero para ejecutar un software
- Creación: Crea un nuevo fichero con un nombre, extensión y ruta determinados
- Eliminación: Elimina un fichero determinado

### Permisos sobre ficheros

- •El usuario que crea el fichero tiene derecho a decidir quien y cómo accede a su fichero
- •Existen 3 grupos para los que se les puede definir permisos
  - u. Propietario
  - g. Grupo
  - o. Resto de usuarios
- •Los permisos que se pueden dar son los siguientes
  - r. Lectura
  - w. Escritura
  - x. Ejecución

### Codificación de caracteres

- · Para poder hacer la equivalencia entre byte y símbolo (número, letra, símbolo especial, etc.) se utiliza la tabla ASCII
- La tabla ASCII permite hasta 128 símbolos

0 1 2 3 4 5 6 7 8 9 A	000 001 002 003 004 005 006 007 010	NUL SOH STX ETX EOT ENQ ACK BEL BS	32 33 34 35 36 37 38 39	20 21 22 23 24 25 26	040 041 042 043 044 045	1. # 8	64 65 66 67 68	40 41 42 43	100 101 102 103	@ABC	96 97 98 99	60 61 62	140 141 142	a b
2 3 4 5 6 7 8 9 A	002 003 004 005 006 007 010	STX ETX EOT ENQ ACK BEL	34 35 36 37 38	22 23 24 25	042 043 044	# 8	66 67	42 43	102	A B	98	62	142	
3 4 5 6 7 8 9 A	003 004 005 006 007 010	ETX EOT ENQ ACK BEL	35 36 37 38	23 24 25	043	#	67	43						b
4 5 6 7 8 9 A	004 005 006 007 010	EOT ENQ ACK BEL	36 37 38	24 25	044	8	177.1		103	C	on		4.40	
5 6 7 8 9 A	005 006 007 010	ACK BEL	37 38	25			60				35	63	143	0
6 7 8 9 A	006 007 010	ACK BEL	38		045		98	44	104	D	100	64	144	d
7 8 9 A	007 010	BEL		28		%	69	45	105	E	101	65	145	е
8 9 A	010		39	100,000	046	8.	70	46	106	F	102	66	146	1
9 A		BS		27	047	.0	71	47	107	G	103	67	147	g
A	811		40	28	050	1	7.2	48	110	H	104	68	150	h
		TAB	41	29	051	)	73	49	111	1	105	69	151	1
	012	LF	42	2A	052	*	7.4	4.4	112	J	106	6A	152	1
B	013	VT	43	28	053		75	48	113	K	107	68	153	k
C	014	FF	4.4	2C	054	V	76	40	114	6	108	6C	154	1
D	015	CR	45	2D	055	2	7.7	4D	115	M	109	6D	155	m
E	016	80	46	2E	056	1	78	4E	116	N.	110	6E	156	n
F	017	SI	47	2F	057	1	79	4F	117	0	111	6F	157	0
10	020	DLE	48	30	060	0	80	50	120	80	112	70	160	p
11	021	DC1	49	31	061	1	81	51	121	0	113	71	161	q
12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	
13	023	DC3	51	33	063	3	83	53	123	8	115	73	163	S
14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	-t
15							85	55						્ય
16														V
17		ETB					87							W
18														M
19														y
1A						1				Z.				2
18	033	ESC	59	3B	073	1	91	58	133		123	78	173	1
1C	034	FS	60	3C	074	*	92	5C	134	1	124	7C	174	1
1D	035	GS	61	3D	075	=	93	5D	135	1	125	70	175	1
1E	036	RS	62	3E	076	20	94	SE SE	136	*	126	7E	176	DEL
	B C D E F 10 11 12 13 14 15 16 17 18 19 11 18 11 10 10	8 013 C 014 D 015 E 016 F 017 10 020 11 021 11 221 12 022 13 023 14 024 16 026 17 027 18 030 10 034 11 034 11 031 11 031 10 031	B 013 VT C 014 FF D 015 CR E 016 S0 F 017 SI 10 020 DLE 11 021 DC1 11 021 DC1 11 022 DC2 13 023 DC3 14 024 DC4 15 025 NAK 16 026 SYN 17 027 ETB 18 030 CAN 17 027 ETB 18 030 CAN 19 031 EM0 1A 032 SUB 1B 033 ESC 1C 034 FS 1E 036 RS	B 013 VT 43 C 014 FF 44 D 015 CR 45 E 016 SO 46 F 017 SI 47 10 020 DLE 48 11 021 DC1 49 12 022 DC2 50 13 023 DC3 51 14 024 DC4 52 15 025 NAK 53 16 026 SYN 54 17 027 ETB 55 18 030 CAN 56 17 027 ETB 55 18 030 CAN 56 19 031 EM) 57 1A 032 SUB 58 1B 033 ESC 59 1C 034 FS 60 1D 035 GS 61 1E 036 RS 62	B 013 VT 43 2B C 014 FF 44 2C D 015 CR 45 2D E 016 SO 46 2E F 017 SI 47 2F 10 020 DLE 48 30 11 021 DC1 49 31 12 022 DC2 50 32 13 023 DC3 51 33 14 024 DC4 52 34 15 025 NAK 93 35 16 026 SYN 54 36 17 027 ETB 55 37 18 030 CAN 56 38 17 027 ETB 55 37 18 030 CAN 56 38 19 031 EM 57 39 1A 032 SUB 58 3A 1B 033 E8C 59 3B 1C 034 FS 60 3C 1D 035 GS 61 3D 1E 036 RS 62 3E	B 013 VT 43 2B 053 C 014 FF 44 2C 054 C 056 C 015 CR 45 2D 056 E 016 SO 46 2E 056 F 017 SI 47 2F 057 10 020 DLE 48 30 060 11 021 DC1 49 31 061 12 022 DC2 50 32 062 13 023 DC3 51 33 063 14 024 DC4 52 34 064 15 025 NAK 53 35 066 17 027 ETB 55 37 067 18 030 CAN 56 38 070 18 030 CAN 56 38 070 14 032 SUB 58 3A 072 18 033 ESC 59 3B 073 1C 034 FS 60 3C 074 1D 035 GS 61 3D 076 1E 036 RS 62 3E 076	B 013 VT 43 2B 053 * C 014 FF 44 2C 054 . D 015 CR 45 2D 055 - E 016 SO 46 2E 056 . F 017 SI 47 2F 057 / 10 020 DLE 48 30 060 0 11 021 DC1 49 31 061 1 12 022 DC2 50 32 062 2 13 023 DC3 51 33 063 3 14 024 DC4 52 34 064 4 15 U25 NAK 53 35 066 5 16 026 SYN 54 36 066 6 17 027 ETB 55 37 067 7 18 030 CAN 56 38 070 8 19 031 EM) 57 39 071 9 1A 032 SUB 58 3A 072 : 1B 033 ESC 59 3B 073 ; 1C 034 FS 60 3C 074 × 1C 034 FS 60 3C 074 × 1C 034 FS 60 3C 0775 = 1E 036 RS 62 3E 076 >	B 013 VT 43 2B 053 + 75 C 014 FF 44 2C 054 , 76 D 015 CR 45 2D 055 - 77 E 016 SD 46 2E 056 , 78 F 017 SI 47 2F 057 / 79 10 020 DLE 48 30 060 0 80 11 021 DC1 49 31 061 1 81 12 022 DC2 50 32 062 2 82 13 023 DC3 51 33 063 3 83 14 024 DC4 52 34 064 4 84 16 025 NAK 53 35 085 5 85 16 026 SYN 54 36 066 6 86 17 027 ETB 55 37 067 7 87 18 030 CAN 56 38 070 8 88 17 027 ETB 55 37 067 7 87 18 030 CAN 56 38 070 8 89 1A 032 SUB 58 3A 072 90 1B 033 E8C 59 3B 073 91 1C 034 FS 60 3C 074 92 1D 035 GS 61 3D 075 93 1E 036 RS 62 3E 076 94	B 013 VT 43 2B 053 + 75 4B C 014 FF 44 2C 054 , 76 4C D 015 CR 45 2D 055 - 77 4D 6 C 017 SI 47 2F 057 / 79 4F 10 020 DLE 48 30 060 0 80 50 11 021 DC1 49 31 061 1 81 51 12 022 DC2 50 32 062 2 82 52 13 023 DC3 51 33 063 3 83 63 14 024 DC4 52 34 064 4 84 54 15 025 NAK 53 35 066 6 86 56 17 027 ETB 55 37 067 7 87 57 18 030 CAN 56 38 070 8 88 58 17 027 ETB 55 37 067 7 87 57 18 030 CAN 56 38 070 8 88 58 18 030 CAN 56 38 070 8 89 59 14 032 SUB 58 3A 072 : 90 5A 18 033 ESC 59 3B 073 ; 91 5B 1C 034 FS 60 3C 074 9 92 5C 1D 035 GS 61 3D 075 = 93 5D 1E 035 GS 61 3D 075 = 93 5D 1E 036 RS 62 3E 076 > 94 5E	B 013 VT 43 2B 053 + 75 4B 113 C 014 FF 44 2C 054 , 76 4C 114 C 0 015 CR 45 2D 055 - 77 4D 115 CR 45 2D 055 - 77 4D 115 CR 45 2D 055 - 77 4D 115 CR 45 2D 056 . 78 4E 118 C 0 017 SI 47 2F 057 / 79 4F 117 C 0 020 DLE 48 30 060 0 80 50 120 C 120 DLE 48 30 060 0 80 50 120 C 120 DLE 48 30 060 0 80 50 120 C 120 DLE 48 30 060 0 80 50 120 C 120 DLE 48 30 063 3 83 53 123 C 023 DC3 51 33 063 3 83 53 123 C 024 DC4 52 34 064 4 84 54 124 DC4 DC4 52 34 064 4 84 54 124 DC4 DC4 DC4 52 34 064 4 84 54 124 DC4 DC4 DC4 52 34 064 6 86 56 126 DC4	B 013 VT 43 2B 053 + 75 4B 113 K C 014 FF 44 2C 054 , 76 4C 114 L D 015 CR 45 2D 055 - 77 4D 115 M E 016 SO 46 2E 056 . 78 4E 116 N F 017 SI 47 2F 057 / 79 4F 117 O 10 020 DLE 48 30 080 0 80 50 120 80 11 021 DC1 49 31 061 1 81 51 121 Q 12 022 DC2 50 32 062 2 82 52 122 R 13 023 DC3 51 33 063 3 83 53 123 S 14 024 DC4 52 34 064 4 84 54 124 T 15 025 NAK 93 35 065 0 85 55 125 U 16 026 SYN 54 36 066 6 86 56 126 V 17 027 ETB 55 37 067 7 87 57 127 W 18 030 CAN 56 38 070 8 89 59 131 Y 18 031 EM) 57 39 071 9 89 59 131 X 19 031 EM) 57 39 071 9 89 59 131 X 10 034 FS 60 3C 074 9 92 5C 134 N 11 D 035 GS 61 3D 075 = 93 5D 135 1 11 D 035 GS 61 3D 075 = 93 5D 135 1 11 D 035 GS 61 3D 075 = 93 5D 135 1 11 D 035 GS 61 3D 075 = 93 5D 135 1	B 013 VT 43 2B 053 * 75 4B 113 K 107 C 014 FF 44 2C 054 , 76 4C 114 L 108 D 015 CR 45 2D 055 - 77 4D 115 M 109 E 016 SO 46 2E 056 , 78 4E 116 N 110 CF 017 SI 47 2F 057 / 79 4F 117 O 111 10 020 DLE 48 30 060 0 80 50 120 80 112 11 021 DC1 49 31 061 1 81 51 121 Q 113 12 DC2 DC2 50 32 062 2 82 52 122 R 114 13 023 DC3 51 33 063 3 83 53 123 8 115 14 024 DC4 52 34 064 4 84 54 124 T 116 15 U25 NAK 53 35 065 5 85 55 125 U 117 16 025 NAK 53 35 066 6 86 56 126 V 118 17 027 ETB 55 37 067 7 87 57 127 W 119 18 030 CAN 56 38 070 8 89 59 131 V 121 1A 032 SUB 58 3A 072 : 90 5A 132 Z 122 18 033 E8C 59 3B 073 ; 91 5B 133 [ 123 1C 034 FS 60 3C 074 * 92 5C 134 V 124 T 105 035 GS 61 3D 075 = 93 5D 135 ] 125 1E 036 RS 62 3E 076 > 94 5E 136 * 126	B 013 VT 43 2B 053 + 75 4B 113 K 107 6B C 014 FF 44 2C 054 , 76 4C 114 L 108 6C D 015 CR 45 2D 055 - 77 4D 115 M 109 6D E 016 SO 46 2E 056 , 78 4E 116 N 110 6E F 017 SI 47 2F 057 / 79 4F 117 0 111 6F 10 020 DLE 48 30 060 0 80 50 120 80 112 70 110 021 DC1 49 31 061 1 81 51 121 Q 113 71 12 022 DC2 50 32 062 2 82 52 122 R 114 72 13 023 DC3 51 33 063 3 83 53 123 8 115 73 14 024 DC4 52 34 064 4 84 54 124 T 116 74 115 025 NAK 53 35 065 5 85 55 125 U 117 75 16 026 SYN 54 36 066 6 86 56 126 V 118 76 17 027 ETB 55 37 067 7 87 57 127 W 119 77 18 030 CAN 56 38 070 2 90 59 131 Y 121 79 18 030 CAN 56 38 070 8 89 59 131 Y 121 79 18 033 EBC 59 3B 073 ; 91 5B 133 [ 123 7B 16 034 FS 60 3C 074 × 92 5C 134 1 124 7C 115 034 FS 60 3C 074 × 92 5C 134 1 124 7C 115 034 FS 60 3C 074 × 92 5C 134 1 124 7C 115 035 CS 68 61 3D 075 = 93 5D 135 1 125 7D 16 035 CS 68 61 3D 075 = 93 5D 135 1 125 7D 16 036 CS 68 62 3E 076 > 94 5E 136 4 126 7E	B 013 VT 43 2B 053 + 75 4B 113 K 107 6B 153 C 014 FF 44 2C 054 , 76 4C 114 L 108 6C 154 D 015 CR 45 2D 055 - 77 4D 115 M 109 6D 155 E 168 SO 46 2E 056 . 78 4E 116 N 110 6E 156 F 017 SI 47 2F 057 / 79 4F 117 O 111 6F 157 10 020 DLE 48 30 060 0 80 50 120 80 112 70 160 112 021 DC1 49 31 061 1 81 51 121 Q 113 71 161 122 DC2 DC2 50 32 062 2 82 52 122 R 114 72 162 13 023 DC3 51 33 063 3 83 53 123 S 115 73 163 14 024 DC4 52 34 064 4 84 54 124 T 116 74 164 150 025 NAK 93 35 065 9 80 50 125 U 117 75 185 160 025 SYN 54 36 066 6 86 56 126 V 118 76 166 17 027 ETB 55 37 067 7 87 57 127 W 119 77 167 180 030 CAN 56 38 070 2 90 5A 132 Z 122 7A 172 180 031 EM) 57 39 071 9 89 59 131 Y 121 79 171 1A 032 SUB 58 3A 072 90 5A 132 Z 122 7A 172 180 034 FS 60 3C 074 × 92 5C 134 1 124 7C 174 150 034 FS 60 3C 074 × 92 5C 134 1 124 7C 174 150 035 GS 61 3D 075 = 93 5D 135 1 125 7B 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 125 7B 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 125 7B 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 125 7B 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 125 7B 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 125 7B 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 125 7B 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 125 7B 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 175 175 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 175 175 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 175 175 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 175 175 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 175 175 175 185 185 036 RS 62 3E 076 > 94 5E 135 A 125 T 175 175 175 175 175 175 175 175 175 175

### Codificación de caracteres

- En muchos casos, 128 símbolos es muy limitado (por ejemplo, en ASCII no existe la ñ o las tildes)
- Es por ello por lo que los sistemas utilizan diferentes codificaciones de caracteres, donde se utiliza mas de un byte para representar caracteres
- La elección de la configuración correcta indica que símbolos se pueden representar
- Algunas comunes:
  - UTF-8, 16, 32
  - ANSI
  - Estándar por idiomas ISO-XXXX

#### Fichero en Java

- En Java se sigue el concepto de un fichero como un conjunto de bytes
- Por lo tanto, Java puede operar con ficheros de cualquier tipo
- Para encontrarlos, su ruta relativa parte de la ubicación del fichero ejecutable
  - · Se permiten también rutas absolutas
- En Java se permiten hacer todas las operaciones con ficheros vistas anteriormente:
  - · Apertura y cierre
  - Lectura y escritura
  - Creación y eliminación

# Apertura y cierre de un fichero

- · Cuando se abre un fichero en Java, se "reserva" el fichero para operar con el
- Se establece un flujo de datos desde el fichero a una variable en Java, que representa al fichero
- A partir de esa variable, se pueden realizar todas las operaciones sobre fichero que se quieran
- Cuando se quiere dejar de usar el fichero, se debe cerrar el mismo, cortando el flujo de datos y liberando la variable



#### La clase File

- La clase que manipula los ficheros en Java se llama File
- Con esta clase se pueden hacer un gran número de operaciones sobre un fichero y sus propiedades, pero no se permite leer ni escribir
- También permite obtener datos del fichero, como rutas, nombres, permisos e incluso si existe
- El resto de clases que manipulan ficheros parten de la existencia de una clase File, por lo que es la base de cualquier operación de manipulación de ficheros

#### La clase File

• El constructor de la clase File tiene el siguiente esquema:

```
File variableFichero = new File("ruta fichero");
```

- Al crear el constructor, la variable variableFichero es un objeto con los datos del fichero que se encuentra en la ruta pasada por parámetro, si es que existe
  - La ruta puede ser absoluta o relativa
- Para cerrar un fichero, se usa la siguiente sentencia:

```
variableFichero.close();
```

•Con esto se cierra el fichero y la variable variableFichero pasará a ser nula (null)

#### La clase File

- Además de la apertura y cierre, la clase File permite realizar un gran número de operaciones:
  - · Comprobar si el fichero existe
  - · Crear un fichero
  - · Borrar el fichero
  - · Obtener nombre, rutas (absolutas y relativas) y extensión del fichero
  - Decir si el fichero es un fichero o un directorio
  - · Obtener el tamaño en bytes del fichero
  - · Consultar y cambiar los permisos del fichero
  - Si es un directorio, obtener la lista de ficheros que contiene el mismo
  - Crear nuevos directorios
  - Renombrar fichero
  - · Etc.

# Ejemplo de uso de File

```
public static void main(String args[]) {
    File fichero = new File("FicheroEjemplo.txt");

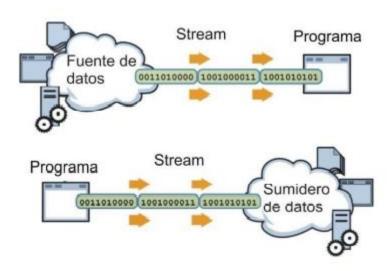
if (fichero.exists()) {
    System.out.println("Nombre del archivo "+ fichero.getName());
    System.out.println("Ruta "+ fichero.getPath());
    System.out.println("Ruta absoluta "+ fichero.getAbsolutePath());
    System.out.println("Se puede escribir "+fichero.canRead());
    System.out.println("Se puede leer "+fichero.canWrite());
    System.out.println("Tamaño "+fichero.length());
}

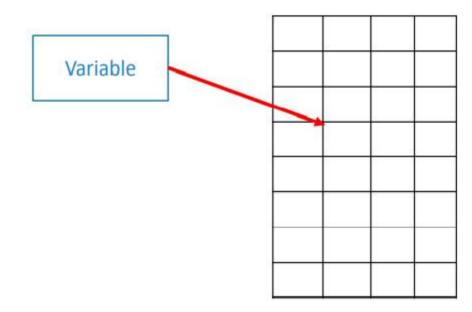
fichero.close();
}
```

Nombre del archivo: FicheroEjemplo.txt
Ruta: FicheroEjemplo.txt
Ruta absoluta: C:\DirectorioEjemplo\FicheroEjemplo.txt
Se puede escribir: false
Se puede leer: true
Tamaño: 1366

#### Stream de datos

- Para poder realizar las operaciones de lectura y escritura de ficheros, Java establece lo que se conoce como un Stream de datos
- · Crea una vía de comunicación entre programa y fichero que permite "moverse" por las distintas partes del fichero
- Existe un puntero que apunta a las partes del fichero





#### Lectura de datos de un fichero

- Para leer datos de un fichero de texto, utilizaremos las siguientes clases:
- · Clase File: Para representar el fichero que se quiere leer

```
File fichero = new File("ruta fichero");
```

• Clase FileReader: Establece el stream de datos de lectura del fichero. Tiene una serie de métodos para leer carácter a carácter. Al constructor del FileReader recibe el objeto File

```
FileReader reader = new FileReader(fichero);
```

 Clase BufferedReader: Crea un buffer a través del FileReader, que permite leer mas de un carácter. El constructor recibe el FileReader como parámetro

```
BufferedReader buffer = new BufferedReader (reader);
```

#### Lectura de datos de un fichero

- Utiliza la función del BufferedReader llamada readLine(), la cual:
  - Devuelve la siguiente línea de texto si existe
  - · Si no existe, devuelve null

```
String linea = buffer.readLine();
```

• Teniendo en cuenta el funcionamiento de readLine(), se puede leer todo el fichero utilizando un bucle while

```
String linea;
while((linea=buffer.readLine()) != null) {
   System.out.println(linea);
}
```

#### Lectura de datos de un fichero

```
import java.io.*;
class LeeFichero {
  public static void main(String [] arg) {
     File archivo = null;
     FileReader reader = null;
     BufferedReader buffer = null;
     try {
        archivo = new File("C:\\directorioArchivo\\archivo.txt");
        reader = new FileReader (archivo);
        buffer = new BufferedReader(reader);
        String linea;
        while( (linea=buffer.readLine()) != null) {
           System.out.println(linea);
      catch(Exception e){
        e.printStackTrace();
     }finally{
        try{
           if( null != fr ){
              fr.close();
        }catch (Exception e2){
           e2.printStackTrace();
```

#### Escritura de datos de un fichero

- Para escribir datos en un fichero de texto, utilizaremos las siguientes clases:
- · Clase File: Para representar el fichero que se quiere leer

```
File fichero = new File("ruta fichero");
```

• Clase FileWriter: Establece el stream de datos de escritura del fichero. Tiene una serie de métodos para escribir en ficheros. Al constructor del FileWriter recibe el objeto File

```
FileWriter writer = new FileWriter(fichero);
```

• Clase PrintWriter: Crea un buffer a través del FileWriter, que permite extender los métodos del FileWriter por otros similares a los que tenemos en la salida de pantalla. El constructor recibe el FileWriter como parámetro

```
PrintWriter pw = new PrintWriter(writer);
```

#### Escritura de datos de un fichero

- Entre las funciones que tenemos en el PrintWriter, las mas comunes son:
  - print("texto"). Imprime el texto pasado por parámetro
  - println("texto"). Imprime el texto pasado por parámetro y hace un salto de línea
- El constructor del FileWriter puede recibir un segundo parámetro **boolean** que indica si queremos escribir el fichero desde cero (false) o si queremos añadir texto al existente (true)

```
FileWriter writer = new FileWriter(fichero);
```

FileWriter writer = new FileWriter(fichero, true);

```
FileWriter writer = new FileWriter(fichero, false);
```

Desde cero Añadir texto

### Escritura de datos de un fichero

```
import java.io.*;
public class EscribirFichero
    public static void main(String[] args)
       File fichero = null;
       FileWriter writer = null;
       PrintWriter pw = null;
        try
            fichero = new File("C:\\directorioArchivo\\archivo.txt");
            writer = new FileWriter(fichero);
            pw = new PrintWriter(writer);
            for (int i = 0; i < 10; i++) {
               pw.println("Linea " + i);
       } catch (Exception e) {
            e.printStackTrace();
       } finally {
          try {
              if (null != fichero) {
                 fichero.close();
          } catch (Exception e2) {
              e2.printStackTrace();
```

#### Borrado de un fichero

- · Para borrar un fichero, se utiliza la clase File
- Existe la función delete() que elimina el fichero

```
public static void main(String args[]) {
    File fichero = new File("FicheroEjemplo.txt");

if (fichero.exists()) {
    System.out.println("Nombre del archivo "+ fichero.getName());
    System.out.println("Ruta "+ fichero.getPath());
    System.out.println("Ruta absoluta "+ fichero.getAbsolutePath());
    System.out.println("Se puede escribir "+fichero.canRead());
    System.out.println("Se puede leer "+fichero.canWrite());
    System.out.println("Tamaño "+fichero.length());
}

fichero.delete();
}
```

# Posibles problemas

- •En las operaciones con ficheros pueden surgir una serie de problemas
- Algunos son:
  - Intentar leer/borrar/escribir sobre un fichero que no existe
  - Intentar leer/borrar/escribir sobre un fichero que ya está siendo modificado por otro programa
  - Intentar escribir o leer un fichero cuando el puntero haya llegado al final del fichero (EOF)

# Control de excepciones

```
import java.io.*;
public class EscribirFichero
    public static void main(String[] args)
       File fichero = null;
       FileWriter writer = null;
       PrintWriter pw = null;
       try
            fichero = new File("C:\\directorioArchivo\\archivo.txt");
           writer = new FileWriter(fichero);
            pw = new PrintWriter(writer);
            for (int i = 0; i < 10; i++) {
               pw.println("Linea " + i);
       } catch (Exception e) {
            e.printStackTrace();
       } finally {
          try {
              if (null != fichero) {
                 fichero.close();
          } catch (Exception e2) {
             e2.printStackTrace();
```