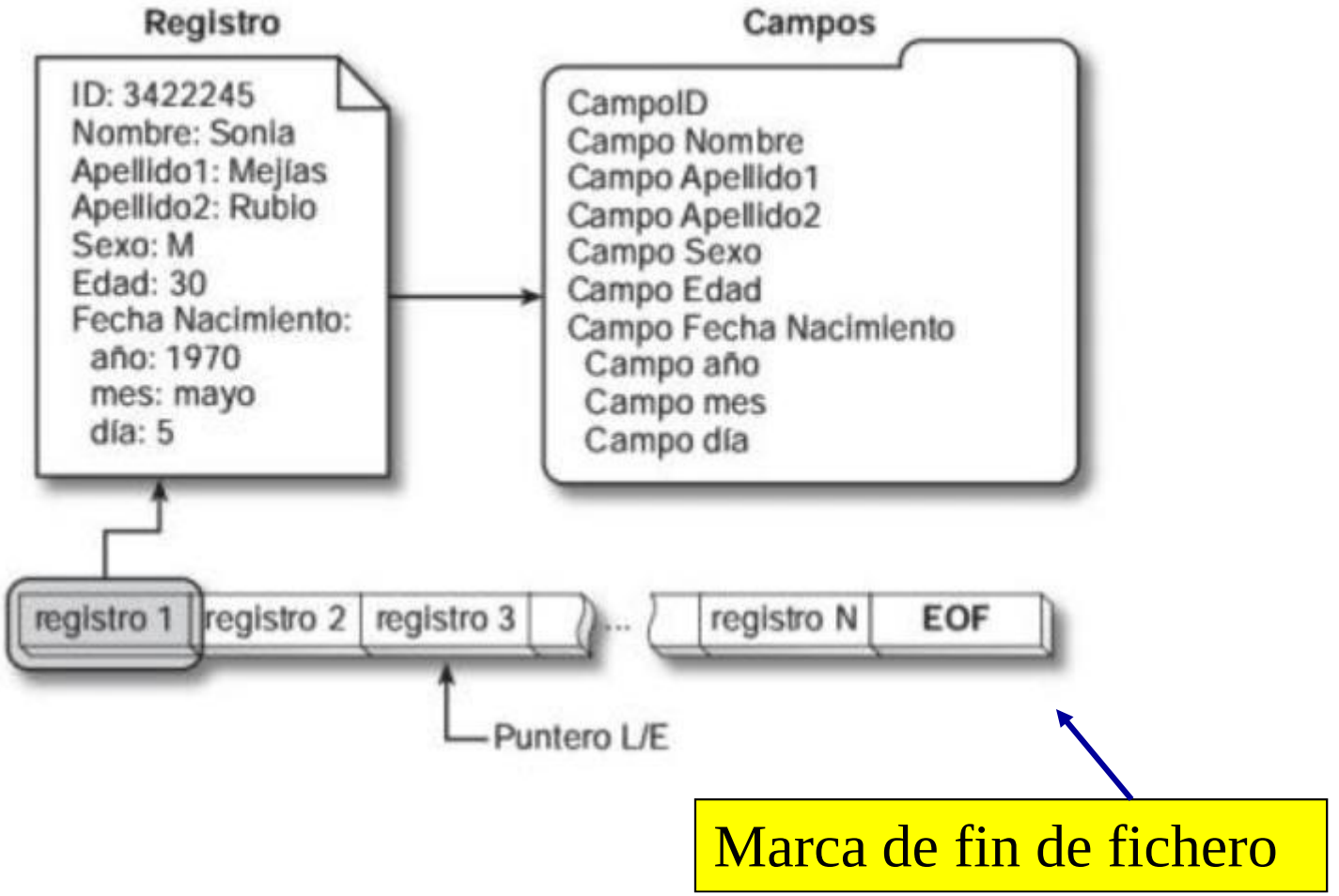
Tema 6. Ficheros

- Definición de ficheros
- Ficheros de texto
- Ficheros binarios
- Registros
- Arrays de Registros

Definición

- Secuencia homogéneas de datos de tamaño no fijado de antemano que facilitan
  - Manejo de grandes cantidades de datos
  - Persistencia de los datos, más allá del momento de ejecución de un programa
  - Uso de datos en diversos programas
- El almacenamiento de los datos se realiza en memoria externa o auxiliar.

Definición


Registro
ID: 3422245
Nombre: Sonia
Apellido1: Mejías
Apellido2: Rubio
Sexo: M
Edad: 30
Fecha Nacimiento:
año: 1970
mes: mayo
día: 5
Campos
CampoID
Campo Nombre
Campo Apellido1
Campo Apellido2
Campo Sexo
Campo Edad
Campo Fecha Nacimiento
Campo año
Campo mes
Campo día
registro 1
registro 2
registro 3
...
registro N
EOF
Puntero L/E
Marca de fin de fichero

Ficheros

- El tamaño de los ficheros
  - no se declara con anticipación
  - puede variar durante la ejecución del programa
  - limitado sólo por el medio de almacenamiento en el que persisten
- El espacio necesario para almacenar los datos se asigna de forma dinámica

Declaración

- Nombre físico de un fichero: el nombre que tiene asignado en memoria externa (disco, etc.).
- Nombre lógico de un fichero: identificador válido Java asociado a un archivo o nombre físico.
- Es necesario asociar los Ficheros lógicos o nombres lógicos con sus correspondientes Ficheros Físicos. (identificador en el disco, etc.)

Método de acceso a los datos

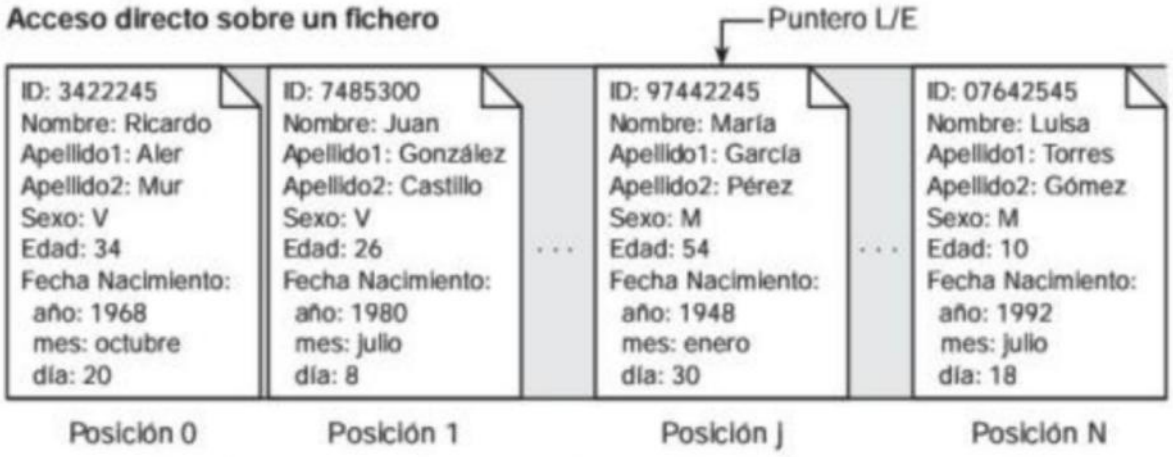
- Acceso secuencial: El acceso al registro i del fichero necesita la exploración secuencial de los (i-1) registros anteriores. Tiempo medio de acceso a un dato dependerá de la posición ocupada en la secuencia.
- Acceso directo: El acceso al registro deseado se realiza directamente indicando la posición que ocupa la secuencia. Tiempo medio de acceso a un registro es constante

Método de acceso a los datos

Acceso secuencial sobre un fichero

Buscar: García Pérez, María (ID = 97442245)
Tiempo = tiempo acceso 1 registro (t) * j (posición que ocupa) = t * j

Acceso directo sobre un fichero



Buscar: García Pérez, María (ID = 97442245)
Tiempo = tiempo acceso 1 registro (t') = t'

Operaciones sobre ficheros

- Cuando de realiza una operación de L/E sobre un fichero, existe un puntero de lectura/escritura (puntero L/E)
- Todo fichero tiene un registro especial, que indica el final del mismo. EOF (End of File) que se incluye automáticamente por el sistema

Operaciones sobre ficheros

1. Creación. Relaciona el nombre lógico con el nombre físico. El nombre lógico será utilizado dentro del programa para acceder a la información.
2. Apertura. Debe abrirse para permitir el acceso a la info. Tipos de apertura:
    1. Sólo lectura (ficheros de flujo de salida)
    2. Sólo escritura (fichero de flujo de entrada)
    3. Lectura/escritura
3. Lectura/escritura. Permiten leer o escribir nuevos datos en el fichero
4. Inserción/borrado. Permiten modificar los valores de los registros almacenados en el fichero, o borrarlos.
5. Renombrado/eliminación. Cambiar el nombre físico de un fichero o eliminarlo por completo.
6. Desplazamiento. Mover el puntero L/E a lo largo de los registros. (no disponible en acceso secuencial)
7. Cierre. Java cierra los ficheros automáticamente. Conviene hacerlo

Operaciones sobre ficheros

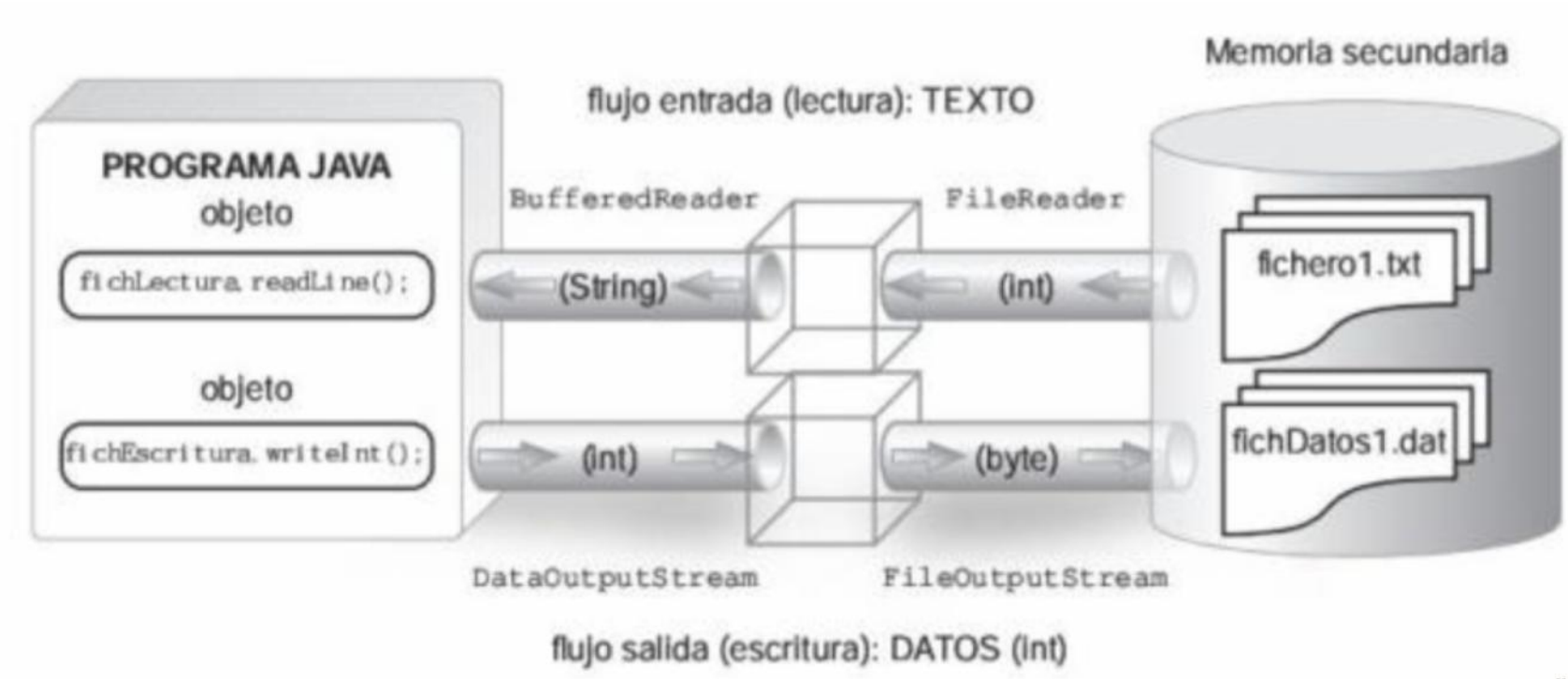
Siempre al menos estas operaciones

1. Crear o asignar un nombre lógico al fichero físico
2. Abrir el fichero:
    1. Sólo lectura (ficheros de flujo de salida)
    2. Sólo escritura (fichero de flujo de entrada)
    3. Lectura/escritura
3. Operar sobre el fichero (lectura/escritura, inserción/borrado, etc...)
4. Cerrar el fichero

Declaración Ficheros en Java

Flujo de datos:

Flujos de entrada/salida sobre ficheros en Java



Flujos de Datos

- E/S puede estar basada:
  - En texto - streams de caracteres legibles (caracteres)
    - Ejemplo: el código fuente de un programa
  - En datos - streams de datos binarios (bytes)
    - Ejemplo: imagen, objetos, sonido, ...
- Uso:
  - Los streams de caracteres se utilizan en la E/S de texto
    - Se denominan lectores (reader) y escritores (writer)
  - Los streams de bytes se utilizan en la E/S de datos
    - Se denominan streams de entrada y streams de salida

Lectura de un **fichero de texto**- Ejemplo

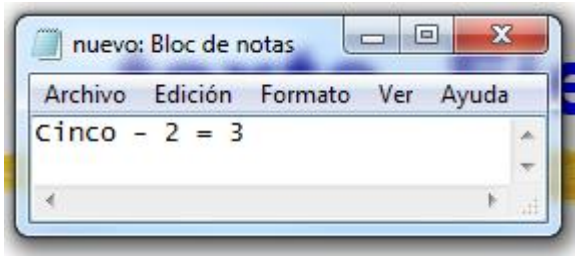
- Programa que lee todas las líneas de un fichero de texto y lo muestra por pantalla utilizando streams (flujos)

```
try {
    BufferedReader lec;
    lec = new BufferedReader(new FileReader("sal1.txt"));
    String st = lec.readLine();
    while (st != null) {
        System.out.println(st);
        st = lec.readLine();
    }
    lec.close();
} catch (FileNotFoundException e) {
    System.out.println("Fichero no encontrado");
} catch (IOException exc) {
    System.out.println("Otro tipo de excepción");
}
```

print() y println()

Escritura de un fichero de texto- Ejemplo

```
try {
    PrintStream flujo;
    flujo = new PrintStream(new FileOutputStream("nuevo.txt"));
    flujo.print("Cinco - " + 2);
    flujo.println(" = " + (5 - 2));
} catch (FileNotFoundException e) {
    System.out.println("Fichero no encontrado");
} catch (IOException exc) {
    System.out.println("Otro tipo de excepción");
}
```



Operaciones habituales con ficheros y directorios

- `exists()` : Comprueba si el fichero o directorio existe
- `getName()`: Devuelve el nombre del fichero o del directorio
- `isDirectory()` / `isFile()`: TRUE si es un directorio o fichero
- `mkdir()` / `createNewFile()`: Creación de un dir. o fichero
- `listFiles()`: Devuelve una lista de objetos 'File' de un directorio
- `delete()`: Borra un fichero/directorio (los directorios deben estar vacíos)

Creación de Registros

- Java permite guardar objetos (registros) en ficheros, siempre y cuando implementen la interfaz Serializable:
  - `public class MySerializableClass implements Serializable {…}`
- ObjectOutputStream:
  - Escribe registros enteros
  - Puede lanzar una IOException
- ObjectInputStream:
  - Lee registros enteros de un fichero
  - Los datos han sido escritos utilizando ObjectOutputStream
  - Puede lanzar una IOException y ClassNotFoundException

Creación de Registro

- Creación del Registro Alumno (en Java es una clase)

```
class Alumno implements Serializable {
    String nombre;
    int edad;
    double nota;
}
```

- Creación del Registro Profesor

```
class Profesor implements Serializable {
    String nombre;
    int codAsignatura;
    String descAsignatura;
}
```

Escritura de un registro en un fichero

REGISTRO

```
class Alumno implements Serializable {

    String nombre;
    int edad;
    double nota;
}
```

Escritura en Fichero

```
public class FicherosBinarios {

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

        try {
            FileOutputStream out = new FileOutputStream("alumnos.dat");
            ObjectOutputStream so = new ObjectOutputStream(out);
            Alumno al1;
            al1 = new Alumno();
            al1.nombre = "Diego";
            al1.edad = 22;
            al1.nota = 10;
            so.writeObject(al1);
            so.close();
        } catch (IOException exc) {
            System.out.println("Otro tipo de excepción");
        }
```

Escribe registro en fichero y cierra

Lectura de un registro de un fichero y S.O.P.

```
try {
    FileInputStream in = new FileInputStream("alumnos.dat");
    ObjectInputStream inso = new ObjectInputStream(in);
    Alumno al2;
    do {
        al2 = (Alumno) inso.readObject();
        System.out.println("El nombre es: " + al2.nombre);
        System.out.println("La edad es: " + al2.edad);
        System.out.println("la nota es: " + al2.nota);
    } while (al2 != null);
    inso.close();
} catch (IOException exc) {
    System.out.println("Otro tipo de excepción");
} catch (ClassNotFoundException ex) {
    System.out.println("Error al convertir");
}
```

Escritura de un array de registros en un fichero

```
try {
    FileOutputStream out = new FileOutputStream("fechas.dat");
    ObjectOutputStream so = new ObjectOutputStream(out);
    Alumnos[] miClase = new Alumnos[20];
    for (int i = 0; i < miClase.length; i++) {
        miClase[i] = new Alumnos();
    }
    for (int i = 0; i < miClase.length; i++) {
        System.out.println("dame el nombre");
        miClase[i].nombre = teclado.nextLine();
        System.out.println("dame la edad");
        miClase[i].edad = teclado.nextInt();
        System.out.println("dame la nota");
        miClase[i].nota = teclado.nextDouble();
        teclado.nextLine();
    }
    for (int i = 0; i < miClase.length; i++) {
        System.out.println("El nombre de " +i +"es: " + miClase[i].nombre);
        System.out.println("La edad de " +i +"es: "  + miClase[i].edad);
        System.out.println("La nota de " +i +"es: "  + miClase[i].nota);

    }
    for (int i = 0; i < miClase.length; i++) {
        so.writeObject(miClase[i]);
    }
    so.close();

} catch (IOException ex) {
    System.out.println("Alguna excepcion");
}
```

Escritura en el array

Muestra el registro por pantalla

Escribe el fichero

Salida:

```
dame el nombre
Raquel
dame la edad
25
dame la nota
9
dame el nombre
Pepe
dame la edad
19
dame la nota
8
El nombre de 0es: Raquel
La edad de 0es: 25
La nota de 0es: 9.0
El nombre de 1es: Pepe
La edad de 1es: 19
La nota de 1es: 8.0
```

Entrada de Datos - Scanner

- Clase de utilidad para "leer" diferentes entradas: teclado, ficheros, ...
- Funcionalidad:
  - Entrada: Cadena de entrada y opcionalmente, un delimitador (por defecto, el espacio en blanco)
  - Salida: Tokens a los cuales se accede por métodos nextX()

Entrada de Datos – Scanner - Uso

- Importar la clase:

```
import java.util.Scanner;
```

- Construcción de un objeto de la clase Scanner con el flujo de entrada:
  - Teclado: `Scanner s = new Scanner (System.in);`
  - Fichero de texto:

```
Scanner s = new Scanner (new File ("fich.txt"));
```

- Uso de sus métodos nextX() para obtener los tokens:
  - `nextInt()`, `nextLong()`, `nextDouble()`, `nextFloat()`: Tipos primitivos
  - `nextLine()`: Lee lo que le queda de la línea actual
  - `hasNextLine()`: Devuelve true si quedan más líneas por leer

Entrada de Datos – Scanner - Ejemplo

```
import java.util.Scanner;
class InteraccionTeclado{
   public static void main (String args[]){
        Scanner ent= new Scanner(System.in);
        S.O.P.("Operador 1: ");
        int op1= ent.nextInt();
        S.O.P.("Operador 2: ");
        int op2= ent.nextInt();
        S.O.P.("Suma: " + (op1 + op2));
       ent.close()
   }
}
```

Ejercicio – Entrada de Datos - Scanner

- Realizar un programa que lea de fichero los datos de una agenda de teléfonos y los muestre por pantalla, teniendo en cuenta que:
  - Cada línea del fichero tiene los datos de un contacto, que serán el nombre y su teléfono separados por el carácter #

  - El programa deberá mostrar los datos de esta manera:

```
Nombre: Maria Martin    Telefono: 123456
Nombre: Pepe Lopez      Telefono: 234567
Nombre: Juan Sanchez    Telefono: 89230
```

Entrada de Datos – Scanner - Solución

```
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.io.File;
public class InteraccionFichero{
   public static void main (String args[]) throws
   FileNotFoundException{
            Scanner ent= new Scanner(new File("agenda.txt"));
            while (ent.hasNextLine()){
                String[] c = ent.nextLine().split("#");
                S.O.P.("Nom: " + c[0] + "\t Tel: " + c[1]);
            }
            ent.close();
   }
}
```