Desarrollo de Aplicaciones Multiplataforma

DOCENTE: Daniel López Lozano

escuelaartegranada

Tema 1. Acceso a Ficheros

Índice de contenidos

- Formas de acceso a ficheros.
- Ficheros de texto y binarios.
- Clases asociadas al acceso a ficheros.

- Un **fichero** es un conjunto de datos (bits) almacenados en un **dispositivo** (disco duro) que los mantendrá de manera **permanente**.

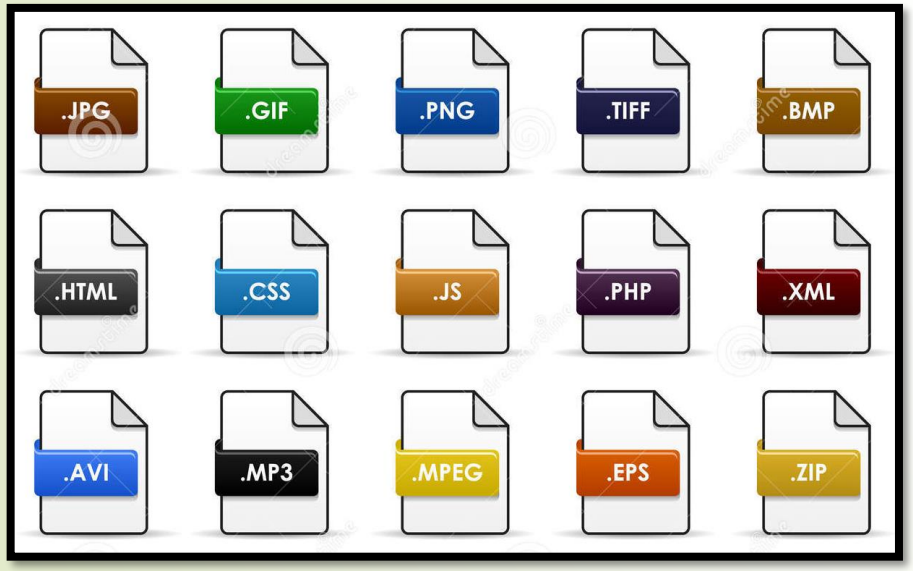
- Se guarda la información en **bloques** cuyo tamaño depende del tipo de información.

- Los datos se encuentran agrupados en **un orden concreto** y solamente son accesibles **en ese orden**.

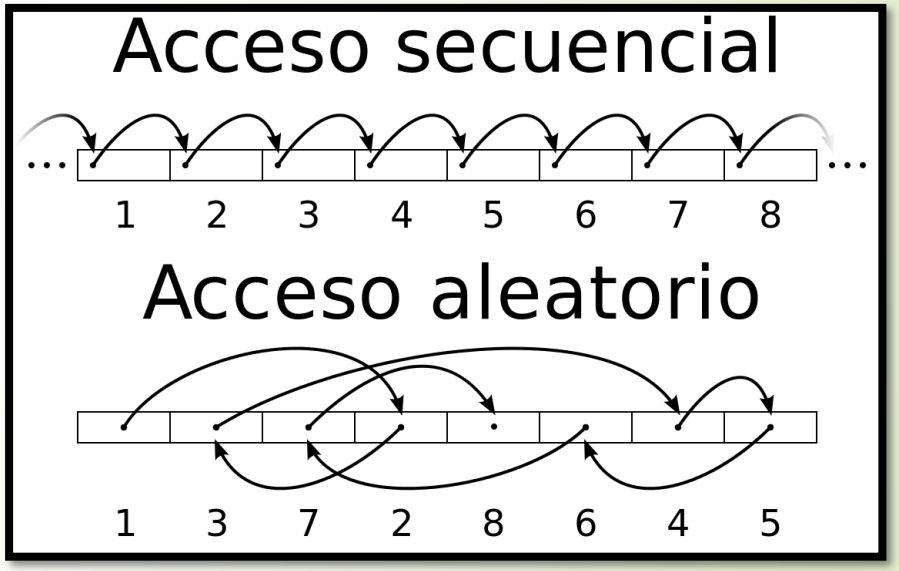
- La forma en que estos datos se agrupan depende completamente de la **persona que lo programe**.

- Existen distintas **formas de acceder a datos en un fichero** dependiendo de como queramos trabajar con ellos.

Tipo de Contenido



Tipo de acceso



- Según el tipo de contenido existen **ficheros de texto** y **ficheros binarios**.

- Los ficheros de texto tamaño del bloque es de un **carácter de tamaño**. Donde la **codificación** marca el tamaño de un carácter que depende de cada lenguaje.

- Tienen la ventaja de que **cualquier sistema puede leer y escribir su contenido** por lo que es un formato universal y no pueden **almacenar virus**.

- La principal **desventaja** es que ocupan más espacio del **necesario**.

- Los **ficheros binarios** son aquellos que no contienen **caracteres reconocibles**.
- El **tamaño de bloque** de información depende del uso que se le dé a los datos y puede **ser variable**.
- Puede almacenar **música, imágenes ,videos**, …
- Son **más eficientes**, pero también **más incompatibles** al ser el tamaño de **bloque variable** se hacen mas difíciles de interpretar.
- Su contenido **es delicado y fácilmente corrompible**.

- Según el **modo de acceso** existen los **ficheros secuenciales** y los **ficheros de acceso aleatorios**.

- Los ficheros secuenciales contienen la información almacenada como **una secuencia de bloques**.

- Para **acceder al ultimo bloque** hay que acceder a todos los anteriores.

- Los ficheros aleatorios pueden acceder a un **bloque en concreto sin necesidad de recorrer los anteriores** por lo que se puede recorrer en cualquier orden.

- La única pega es que el **tamaño del bloque debe ser conocido** por la aplicación.
- La **conveniencia** de un tipo de acceso u otro **depende del uso** que se quiera hacer del fichero.
- Las **operaciones básicas** son las de lectura y las de escritura.
- Dentro de las de escritura existen las operaciones de **alta, baja y modificación** de bloques.
- Otras operaciones son **creación, apertura y cierre**.

- Las operaciones de escritura son las que implican **mayor complejidad** de mantenimiento.

- Cualquier modificación de los datos puede provocar una **corrupción del fichero** o la necesidad de usar **ficheros auxiliares**.

- Desde el punto de vista del programador de aplicaciones puede ser **un problema más complejo que la aplicación en si misma.**

- Esto provoca la evolución hacia los sistemas gestores de bases de datos **(SGBD).**

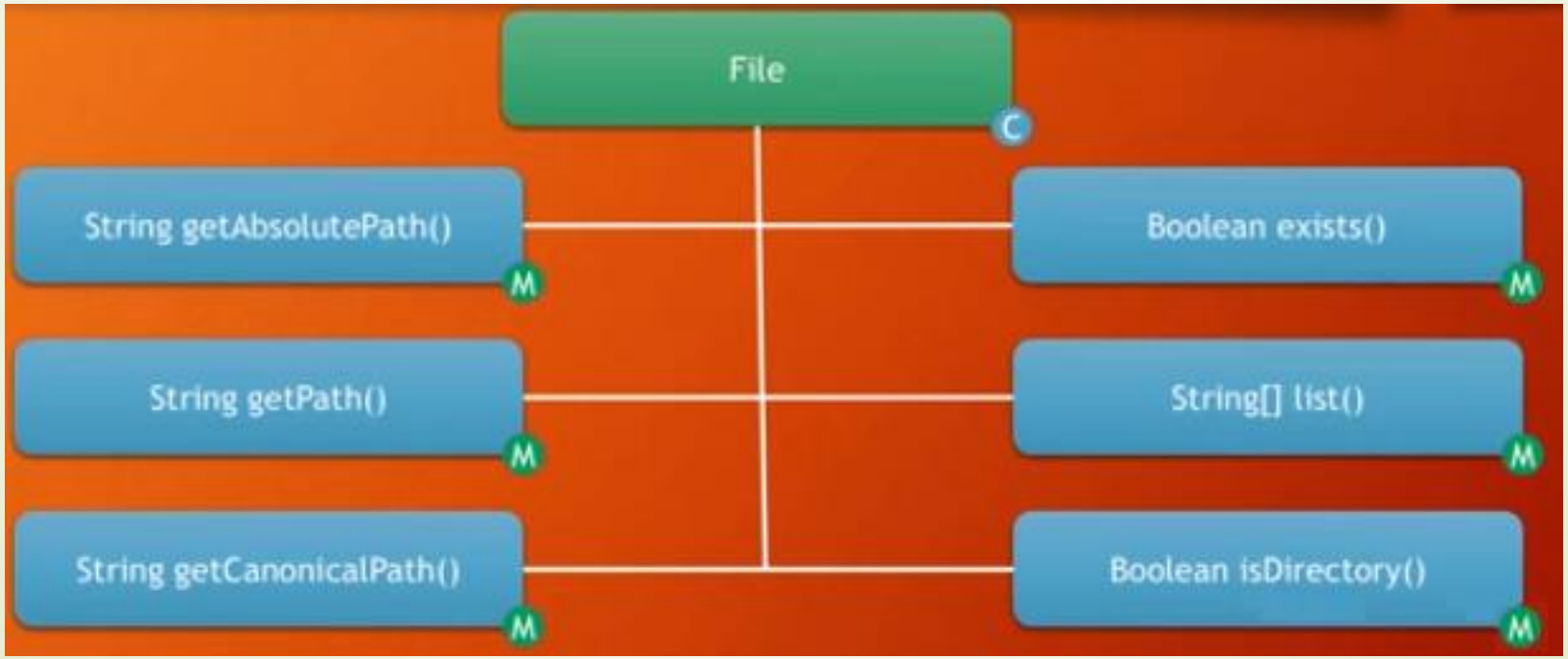
- El acceso **secuencial es eficiente** para hacer **backups y resúmenes** de datos.

- El acceso aleatorio es eficiente para aplicaciones de usuario que solo necesitan **trabajar con un conjunto de datos** cada vez.

- Un sistema gestor de base de datos ofrece todas la **ventajas y usos de forma transparente,** siendo una solución integrada y de caja negra para el **almacenamiento de datos** masivo.

- La **clase File** nos permite obtener información sobre **ficheros y directorios**. (metadatos o atributos).

- La **clase File** puede representar **un fichero particular** o los nombres de un **conjunto de ficheros** de un directorio.



Mostrar información de un fichero

```
String resultado;
System.out.println("INFORMACIÓN SOBRE EL FICHERO:");
File f = new File("Ruta del fichero o directorio");
if(f.exists()){
      resultado+="Nombre del fichero  : "+f.getName()+"\n"+
                 "Ruta                : "+f.getPath()+"\n"+
                 "Ruta absoluta       : "+f.getAbsolutePath()+"\n"+
                 "Se puede leer       : "+f.canRead()+"\n"+
                 "Se puede escribir   : "+f.canWrite()+"\n"+
                 "Tamaño              : "+f.length()+"\n"+
                 "Es un directorio    : "+f.isDirectory()+"\n"+
                 "Es un fichero       : "+f.isFile()+"\n"+
                 "Nombre del directorio padre: "+f.getParent()+"\n";
}
System.out.println(resultado);
```

Crear/borrar directorios y ficheros

```
File d = new File("NUEVODIR"); //directorio que creo a partir del actual
File f1 = new File(d,"FICHERO1.TXT");
File f2 = new File(d,"FICHERO2.TXT");

d.mkdir();//CREAR DIRECTORIO
try
{
    f1.createNewFile());
    System.out.println("FICHERO1 creado correctamente...");
    f2.createNewFile());
    System.out.println("FICHERO2 creado correctamente...");
}catch (IOException e){
    JOptionPane.showMessageDialog(frame,"Error al crear los ficheros");
    e.printStackTrace();//Muestra la traza del error
}

if(f2.delete())
{
    System.out.println("Fichero borrado con exito");
}else{
    System.out.println("Imposible borrar el fichero");
}
```

Listar contenido de un directorio

```
String dir = ".";
String resultado="";
File f = new File(dir);
String[] archivos = f.list();
System.out.println("Ficheros en el directorio actual:"
                                        +archivos.length+"\n");
for (int i = 0; i < archivos.length; i++) {
   resultado+=archivos[i]+"\n";
}
System.out.println(resultado);
```

Más información en:

https://www.discoduroderoer.es/clase-file-y-sus-metodos/

- Para trabajar con **ficheros de texto** utilizaremos las clases **FileReader** y **FileWriter**.

- Añadiremos la **capa BufferedReader** y **PrintWriter** sobre **FileReader** y **FileWriter** respectivamente para realizar operaciones de manera **más cómoda**.

- Estas clases puede lanzar la excepción **IOException y FileNotFoundException** entre otras.

- Usaremos un bloque **try catch** para controlar posible errores de manera eficiente.

Mostrar contenido de un fichero

```
try{
      FileReader fr = new FileReader("FichTexto.txt");
      BufferedReader fichero = new BufferedReader(fr);
      String resultado="CONTENIDO DEL FICHERO:\n";

      while((linea = fichero.readLine())!=null)
      {
        resultado+=linea+"\n";
      }
      System.out.println(resultado);
      fichero.close();

}catch (FileNotFoundException fn ){
      JOptionPane.showMessageDialog(frame,"No se encuentra el fichero");
      fn.printStackTrace();
}catch (IOException io) {
      JOptionPane.showMessageDialog(frame,"Error de E/S ");
      io.printStackTrace();
}
```

Escribir datos en un fichero

```
try{
    FileWriter fw = new FileWriter("FichTexto.txt"); //Borra el fichero si existe
    PrintWriter fichero = new PrintWriter(fw);

    for (int i=1; i<=10; i++){
      fichero.println("Fila numero: "+i); //escribe en la posicion actual
    }
    fichero.close();

}catch (FileNotFoundException fn ){
    JOptionPane.showMessageDialog(frame,"No se encuentra el fichero");
    fn.printStackTrace();
}catch (IOException io) {
    JOptionPane.showMessageDialog(frame,"Error de E/S ");
    io.printStackTrace();
}
```

- Las clases **FileInputStream** y **FileOutputStream** son las clases básicas para el uso de datos binarios.
- Trabajar con ficheros binarios es **más complejo** porque somos **dependientes del tamaño** de datos.
- Para facilitarnos las tarea usaremos **serialización** mediante la interfaz **Serializable** al definir los datos.
- Esto permite usar las clases **ObjectOutputStream** y **ObjectInputStream** que realizan la transformación de objeto a fichero de forma automática.

Definición de datos serializables

```
public class Persona implements Serializable{
  private static final long serialVersionUID = 8799656478674716638L;
  private String nombre;
  private int edad;

  public Persona(String nombre,int edad)  {
    this.nombre=nombre;
    this.edad=edad;
   }
  public Persona() {
    this.nombre=null;
   }
  public void setNombre(String nom){nombre=nom;}
  public void setEdad(int ed){edad=ed;}

  public String getNombre(){return nombre;}
  public int getEdad(){return edad;}
}//fin Persona
```

Escritura de datos binarios

```
Persona persona;//para contener los datos de la persona
FileOutputStream fileout = new FileOutputStream("FichPersona.dat");
ObjectOutputStream dataOS = new ObjectOutputStream(fileout);

String nombres[] = {"Ana","Luis Miguel","Alicia","Pedro",
                    "Manuel","Andrés","Julio","Antonio","María Jesús"};
int edades[] = {14,15,13,15,16,12,16,14,13};

System.out.println("GRABANDO DATOS...");
for (int i=0;i<edades.length; i++){ //recorro los arrays
    persona= new Persona(nombres[i],edades[i]);
    dataOS.writeObject(persona);
}
System.out.println("FIN DE GRABACION...");

dataOS.close();
```

Lectura de datos binarios

```
Persona persona; // defino la variable persona
File fichero = new File("FichPersona.dat");
FileInputStream fileout= new FileInputStream(fichero);
ObjectInputStream dataIS = new ObjectInputStream(fileout);

while (fileout.available()>0){
        // lectura del fichero
        persona = (Persona)dataIS.readObject()
        resultado+="Nombre: "+persona.getNombre()+"\n"+
                   "Edad: "+persona.getEdad()+"\n"+
                   "-----------------------------\n";

}
System.out.println(resultado);
dataIS.close();
```

- Java proporciona una clase para el acceso aleatorio de ficheros llamada **RandomAccessFile**.

- Proporciona **métodos para movernos a lo largo** de todos los datos organizados secuencialmente.

- Podemos movernos hacia **adelante, hacia atrás y colocarnos en cualquier punto** del fichero para leer o escribir.

- La unidad básica de bloque **es un byte** y para llegar a una **posición concreta** del fichero se tiene que calcular los **bloques delante** de dicha posición.

- Los métodos que ofrece **RandonAccesFile** son:
  - ✓ **getFilePointer():** Obtiene la posición actual del puntero.
  - ✓ **seek(int posicion):** Mueve el puntero a la posición indicada.
  - ✓ **writeTipo(Tipo):** Escribe un dato en la posición actual del puntero.
  - ✓ **readTipo(Tipo):** Lee el dato existente en la posición actual del puntero

Ejemplos de uso en:

http://puntocomnoesunlenguaje.blogspot.com/2013/06/java-ficheros-acceso-aleatorio.html

Bibliografía

- **Ramos Martín, Alicia y Ramos Martín, MªJesús:**
  "Acceso a Datos".Editorial Garceta. 2012

- **Córcoles Tendero, J.Ed. y Montero Simarro, Francisco:**
  "Acceso a Datos. CFGS". Editorial Ra-Ma. 2012

- **"Sistemas abiertos". Contenidos de la asignatura.**
  http:http://deim.urv.cat/~pedro.garcia/SOB/
  Última visita: Septiembre 2017.

- **Ejercicios Acceso a Datos 2ª CFGS DAM**
  https://github.com/andresmr/AccesoDatos
  Última visita: Septiembre 2017

- **Master en desarrollo de aplicaciones Android. Universidad Politécnica de Valencia**
  http://www.androidcurso.com/index.php/recursos/42-unidad-9-almacenamiento-de-datos/299-preferencias
  Última visita: Septiembre 2017