Java Files

File es una forma de referenciar de un ruta en un sistema de ficheros. Esta ruta puede no existir físicamente o podría ser la ruta correspondiente con un directorio (carpeta).

Tipos de referencias

Absoluta: empieza en la unidad de almacenamiento
Ej:"C:\\carpeta\\fichero.txt"
Relativa: Buscan archivos dentro de la carpeta/paquete donde se ejecuta el programa.
Ej: carpeta/fichero.txt
//averiguar cuál es el directorio actual de trabajo
System.getProperty("user.dir"));

Constructores

```
File(String pathname)
File(S tring direct orio, String nombre Arc hivo)
```

Inicializar File

File nameFile = new File(String path);

Métodos File

createNewFile()	si no existe, lo crea
delete()	borra
isDirectory()	true si referencia a un directorio
isFile()	true si referencia a un archivo
mkdir()	crea el directorio
mkdirs()	crea todos los directorios necesarios para crear un fichero cuya ruta coincida con la representada
length()	tamaño
listFiles()	array File[] con los archivos del directorio
canExecute(), canRead(), canWrite()	true cuando se tiene permisos de ejecución, lectura o escritura sobre un fichero

Array archivos de un directorio

```
File directorio = new File("C:/");
File[] lista;
if ((directorio.exists()) && (directorio.isDirectory())){
 lista = directorio.listFiles();
}else{
 System.out.println("El directorio no existe");
}
```

Flujo de datos/Stream

Representa cualquier fuente que proporcione datos al programa, o cualquier sumidero que tome datos del programa.
Flujo de datos binario/ de bytes: información en formato binario. más compacto. Objetos deben ser serializables.
Flujo de datos de texto: información en formato de texto. Más fácil de leer desde otros lenguajes.
Flujo de entrada/Input: flujos que proporcionan datos.
Flujo de salida/Output: el programa escribe datos

Jerarquía Entrada/Salida en binario

Output:
Datos primitivos:FileOutputStream -> DataOutputStream
Objetos:FileOutputStream -> ObjectOutputStream (Objetos serializables)
Input:
Datos primitivos:FileInputStream -> DataInputStream
Objetos:FileInputStream -> ObjectInputStream (Objetos serializables)

Métodos OutputStream y Writer

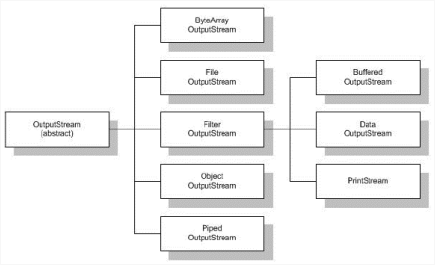
close()	cierra el flujo y libera recursos
flush()	sincroniza este flujo de datos con el dispositivo al cual se están escribiendo los datos
write(byte[] b)	escribe el array
write(byte[] b, int off, int len)	escribe len bytes del array b, empezando a escribirlos en el offset indicado por off

Métodos OutputStream y Writer (cont)

abstract void write(int b)	escribe 1 byte

Clase abstracta que representa un flujo de datos de salida binario cualquiera

Salida de bytes



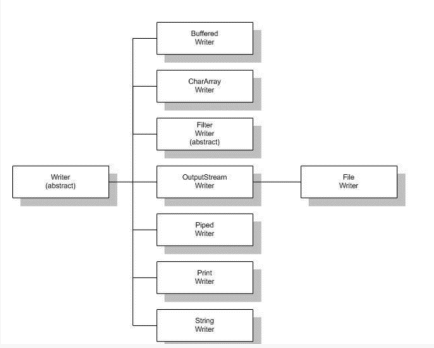
Salida de bytes

```
File f1 = new File("src/main/resources/File.csv");
FileOutputStream fout = null;
DataOutputStream dout = null;
ObjectOutputStream oout = null;
try {
 if (f1.createNewFile()) {
  System.out.println("Se ha creado el archivo");
 }
 fout = new FileOutputStream(f1);
 dout = new DataOutputStream(fout);
 oout = new ObjectOutputStream(fout);
 dout.writeChar('a');
 dout.writeBoolean(true);
 dout.writeInt(34);
 dout.writeFloat(34.4f);
 oout.writeObject(new Date());
 oout.writeObject(new ObjetoPrueba(4, "hola"));
 dout.writeBytes("hola");
} catch (IOException e) {
 throw new RuntimeException(e);
} finally{
 if(oout != null){
  try{
   oout.close();
```

Salida de bytes (cont)

```
  }catch (IOException ex)
   ex.printStackTrace();
  }
 }
 if(fout != null){
  try{
   oout.close();
  }catch (IOException ex)
   ex.printStackTrace();
  }
 }
 if(dout != null){
  try{
   oout.close();
  }catch (IOException ex)
   ex.printStackTrace();
  }
 }
}
//Clase ObjetoPrueba
class ObjetoPrueba implements Serializable {
 private int a;
 private String b;
 public ObjetoPrueba(int a, String b) {
  this.a = a;
  this.b = b;
 }
 @Override
 public String toString() {
  return a+", "+b;
 }
}
```

Salida de texto



Writer representa cualquier flujo de datos de salida donde la info. se va a representar en modo texto.

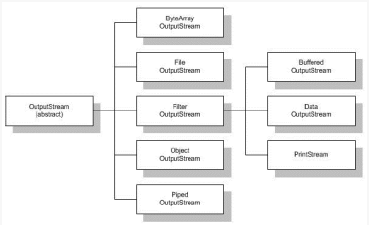
Salida Texto

```
import java.io.*;
public class Ejemplo5 {
 public static void main(String[] args) {
  File file = new File("C:/datos.txt");
  PrintWriter printWriter = null;
  try {
   printWriter = new PrintWriter(file);
   printWriter.println(25);
   printWriter.println(2.5);
   printWriter.println(true);
   printWriter.printl n(3.6F);
   printWriter.println("hola" + 2.4+ ", "+ 3);
   printWriter.close();
  } catch (IOException ex)
   System.out.printl ("Error durante el proceso");
   ex.printStackTrace();
  } finally
   ...
  }
 }
}
```

Metodos InputStream y Reader

close()	cierra el flujo
int availa-ble()	devuelve el nº de bytes que se pueden leer sin bloqueo (solo InputStream)
int read(b-yte[] b)	lee bytes y almacena en array. Devuelve el numero de bytes que se han leído. -1 si se ha alcanzado el final. Si no hay bytes, espera a que haya (bloqueante)
int read()	lee un único byte
abstract int read(byte[] b, int off, int len)	lee hasta len bytes y almacena en b, empezando a leer en el offset
skip(long n)	ignora los próximos n bytes
boolean ready()	true si hay caracteres para leer (Reader)

Entrada de bytes



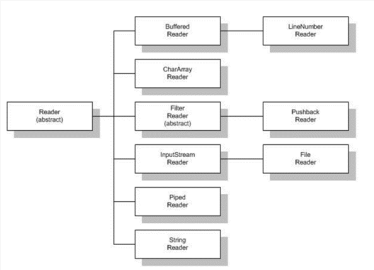
Entrada de bytes

```
File f1 = new File("src/main/resources/File.csv");
FileOutputStream fout;
DataOutputStream dout;
ObjectOutputStream oout;
try {
 if (f1.createNewFile()) {
  System.out.println("Se ha creado el archivo");
 }
 finput = new FileInputStream(f1);
 dinput = new DataInputStream(finput);
 oinput = new ObjectInputStream(finput);
 System.out.println(dinput.readChar());
```

Entrada de bytes (cont)

```
 System.out.println(dinput.readBoolean());
 System.out.println(dinput.readInt());
 System.out.println(dinput.readFloat());
 System.out.println((Date) oinput.readObject());
 System.out.println((ObjetoPrueba) oinput.readObject());
 finp.close();
} catch (IOException e) {
 throw new RuntimeException(e);
} catch (ClassNotFoundException e) {
 throw new RuntimeException(e);
}finally{
 ...
}
```

Entrada Texto



Entrada Texto

```
import java.io.*;
public class Ejemplo5 {
 public static void main(String[] args) {
  File file = new File("C:/datos.txt");
  FileInputStream fileInputStream = null;
  InputStreamReader inputStreamReader = null;
  BufferedReader bufferedReader = nulll;
  try {
   fileInputStream = new FileInputStream(file);
   inputStreamReader = new InputStreamReader(fileInputStream);
   bufferedReader = new BufferedReader(inputStreamReader);
   int entero = Integer.parseInt(bufferedReader.readLine());
   double realDoble = Double.parseDouble(bufferedReader.readL-
ine());
   boolean logico = Boolean.parseBoolean(bufferedReader.readLi-
ne());
```

Entrada Texto (cont)

```
   float real = Float.parseFloat(buffe redReader.readLine());
   System.out.println(entero + " " + realDoble + " " + logico + " " +
real);
   System.out.println(buffer.readLine());
  } catch (IOException ex)
   System.out.printl ("Error durante el proceso");
   ex.printStackTrace();
  } finally
   ...
  }
 }
}
```