Paquete java.io: Clases Importantes para Manejo de Archivos y Flujos

Clase BufferedReader 📖

Método	Descripción
readLine()	Lee una línea completa de texto.
close()	Cierra el flujo de entrada.

BufferedReader permite leer texto de un flujo de entrada de manera eficiente, almacenando en búfer las entradas para reducir el número de llamadas de lectura en el sistema de archivo. Ideal para leer archivos grandes línea por línea.

```
BufferedReader br = new BufferedReader(new FileReader("file.txt"));
 String line;
while ((line = br.readLine()) != null) {
      System.out.println(line);
}
br.close();
```

Clase BufferedWriter 📝

Método	Descripción
write(String s)	Escribe una cadena en el archivo.
newLine()	Escribe un salto de línea en el archivo.
close()	Cierra el flujo de escritura.

BufferedWriter permite escribir texto en un archivo de manera eficiente mediante un búfer, evitando la escritura directa cada vez que se llama al método de escritura.

```
BufferedWriter bw = new BufferedWriter(new
FileWriter("file.txt")); bw.write("Hola Mundo");
bw.newLine();
bw.write("Este es un ejemplo.");
bw.close();
```

Clase File 📋

Método	Descripción
exists()	Verifica si el archivo o directorio existe.
createNewFile()	Crea un archivo si no existe.

File representa archivos y directorios en el sistema de archivos, permitiendo la creación, eliminación y verificación de propiedades de los mismos

delete()	Elimina el archivo o directorio.
isDirectory()	Verifica si es un directorio.

```
File file = new File("nuevoArchivo.txt");
 if (!file.exists()) {
      file.createNewFile();
}
```

Clase FileReader 🔍

FileReader permite leer archivos de texto, caracter por caracter.

Método	**Descripción**
read()	Lee un caracter del archivo.
close()	Cierra el flujo de lectura.

```
FileReader fr = new FileReader("file.txt"); int ch;
while ((ch = fr.read()) != -1) {
      System.out.print((char) ch);
}
fr.close();
```

Clase FileWriter 🖋

FileWriter permite escribir texto en archivos de manera directa.

Método	**Descripción**
write(String s)	Escribe una cadena en el archivo.
close()	Cierra el flujo de escritura.

```
FileWriter fw = new FileWriter("file.txt");   fw.write("Este
es un texto de ejemplo."); fw.close();
```

Clase FileInputStream 📼

FileInputStream permite leer archivos en formato binario, como imágenes y archivos multimedia.

Método	**Descripción**
read()	Lee un byte del archivo.
close()	Cierra el flujo de entrada.

```
FileInputStream fis = new FileInputStream("file.txt");  int byteData;
while ((byteData = fis.read()) != -1) {
      System.out.print((char) byteData);
}
fis.close();
```

Clase FileOutputStream

FileOutputStream permite escribir datos en formato binario en archivos.

Método	Descripción
write(int b)	Escribe un byte en el archivo.
close()	Cierra el flujo de salida.

```
FileOutputStream fos = new FileOutputStream("file.txt");
fos.write(65);
// Escribe 'A'
fos.close();
```

Clase ObjectOutputStream y ObjectInputStream

ObjectOutputStream y ObjectInputStream permiten serializar (escribir) y deserializar (leer) objetos, respectivamente.

Método	Descripción
writeObject(Object obj)	Escribe un objeto en el flujo (ObjectOutputStream).
readObject()	Lee un objeto del flujo (ObjectInputStream).
close()	Cierra el flujo.

```
// Escribiendo un objeto
ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("obj.dat"));
oos.writeObject(new String("Texto serializado"));
oos.close();
// Leyendo un objeto
ObjectInputStream ois = new ObjectInputStream(new FileInputStream("obj.dat")); String readObject
= (String) ois.readObject();
ois.close();
```

Clase PrintWriter

Método	Descripción
print(String s)	Escribe una cadena sin salto de línea.
println(String s)	Escribe una cadena con salto de línea.
close()	Cierra el flujo de escritura.

PrintWriter facilita la escritura de texto en archivos, permitiendo la impresión de cadenas y números de manera formateada.

```
PrintWriter pw = new PrintWriter(new FileWriter("file.txt")); pw.println("Línea
de texto con salto de línea"); pw.print("Texto en la misma línea");
pw.close();
```