Flujos en Kotlin

InputStreams y OutputStreams:

- Los InputStreams y OutputStreams son flujos que trabajan a nivel de bytes, es decir, manejan datos byte a byte.
- Por ejemplo, si deseamos leer un archivo byte a byte, podemos utilizar la clase FileInputStream, y si queremos escribir en un archivo, utilizaremos FileOutputStream.
- Estas clases son utilizadas en situaciones de muy bajo nivel, como cuando se necesita modificar un único byte en un archivo. En general, se utilizan clases más cómodas y de nivel superior para operaciones de lectura y escritura de datos.

Readers y Writers:

- A diferencia de los flujos de bytes, los Readers y Writers trabajan a nivel de caracteres. Esto es importante debido a que en el contexto actual, con Unicode, un solo carácter podría ocupar más de un byte.
- Cuando deseamos leer caracteres de un archivo, podemos utilizar clases como FileReader, y para escribir caracteres, utilizamos FileWriter.
- Estas clases de Readers y Writers se basan en realidad en los InputStreams y OutputStreams subyacentes para realizar las operaciones.

InputStreamReader y OutputStreamWriter:

- A veces, es necesario combinar conceptos y trabajar con caracteres cuando Java proporciona clases que operan con bytes. Para resolver esto, existen InputStreamReader y OutputStreamWriter.
- InputStreamReader toma un objeto que lee bytes y lo

convierte en caracteres. Por otro lado, OutputStreamWriter toma caracteres y los convierte en bytes que componen esos caracteres.

BufferedReaders y PrintWriters:

- Cuando trabajamos con caracteres, generalmente no procesamos uno por uno, sino que trabajamos con líneas completas.
- Java ofrece clases que automáticamente gestionan los búfers para nosotros, lo que brinda comodidad y eficiencia en la lectura y escritura de datos.
- Por ejemplo, la clase PrintWriter tiene un método print(ln) que puede imprimir elementos más complejos, como números flotantes o cadenas largas de texto.
- Además, podemos utilizar clases como BufferedReader para leer líneas completas y BufferedWriter para escribir líneas completas de manera eficiente.
- Al utilizar estas clases de búfer, podemos trabajar con bloques de datos en lugar de procesar byte por byte, lo que mejora la eficiencia y comodidad en la manipulación de datos.