Primero antes de identificar las diferencias entre las clases java.io y java.nio. Java.nio Es la elección **estándar y moderna** para la gran mayoría de escenarios de almacenamiento y red:

- **Gestión de archivos y directorios (cualquier tamaño):** Utiliza `Path` y `Files` para crear, leer, mover, eliminar o listar contenido. Reemplaza de forma definitiva a la clase antigua `java.io.File`.
- **Redes de alto rendimiento:** Ideal para servidores que necesitan gestionar miles de conexiones concurrentes manteniendo un consumo bajo de hilos mediante entrada/salida no bloqueante.
- **Procesamiento de archivos gigantes (GBs):** Permite manipular volúmenes masivos de datos con `FileChannel`, mapeando memoria directamente al sistema operativo sin saturar el *Heap* de la JVM.

Y java.io queda reservado principalmente para **interacción por consola, flujos simples o compatibilidad**:

- **Entrada y salida por consola:** Adecuado para scripts sencillos o interacción rápida en terminal utilizando `System.out.println()` o `Scanner`.
- **Integración con librerías externas:** Necesario cuando un *framework* o librería de terceros requiera explícitamente trabajar con `InputStream` u `OutputStream`.
- **Mantenimiento de sistemas heredados (*legacy*):** Proyectos construidos antes de Java 7 que no han sido migrados a las APIs modernas.

Parte A: Investigación teórica (java.nio vs java.io)

1. Diferencias principales entre java.io y java.nio

- **Orientación de operaciones:**
  - java.io está orientado a flujos de datos (*streams*), procesando la información byte a byte o carácter a carácter.
  - java.nio está orientado a bloques, utilizando canales (*Channels*) y buffers (*Buffers*), lo que optimiza el rendimiento al interactuar directamente con el sistema operativo.
- **Manejo de rutas y archivos:**
  - En java.io, la gestión de archivos se realiza principalmente mediante la clase java.io.File.
  - En java.nio, se utiliza la interfaz java.nio.file.Path (representación de la ruta) y la clase utilitaria java.nio.file.Files, que ofrece métodos estáticos atómicos y eficientes para operar con ficheros y directorios.
- **Gestión de bloqueos (*Blocking vs Non-blocking*):**
  - java.io utiliza operaciones E/S bloqueantes.
  - java.nio permite operaciones no bloqueantes (*Non-blocking I/O*) mediante selectores.

Operación	Clase y Método en java.nio
Crear fichero / directorio	Files.createFile(Path path) Files.createDirectories(Path path)
Borrar fichero / directorio	Files.delete(Path path) Files.deleteIfExists(Path path)
Renombrar / Mover	Files.move(Path source, Path target, CopyOption... options)
Abrir fichero	Files.newBufferedReader(Path path)

	`Files.newBufferedWriter(Path path)`
Juego de caracteres (*Charset*)	`StandardCharsets.UTF_8` / `StandardCharsets.ISO_8859_1`
Leer todas las líneas	`Files.readAllLines(Path path, Charset cs)`
Escribir líneas	`Files.write(Path path, Iterable lines, Charset cs, OpenOption... options)`
Añadir información (*Append*)	`Files.write(Path path, lines, StandardOpenOption.APPEND)`
Cerrar fichero	Implementa `AutoCloseable` (se cierra automáticamente con `try-with-resources`)