Ejercicios opcionales

1. Crear un programa que pida al usuario que introduzca una ruta de fichero, cree una instancia de *Path* a partir de la ruta y muestre el nombre del fichero, el directorio padre y si la ruta es absoluta o relativa.
2. Crear un programa que elimine un directorio y todo su contenido de manera recursiva. El programa debe aceptar la ruta del directorio a eliminar como entrada (se pasará al programa desde la línea de comandos).
3. Crear un programa que calcule y muestre estadísticas sobre un directorio dado (número de ficheros que contiene, tamaño total de los ficheros, número de directorios, extensión de fichero más común, etc.).
4. Crear un programa que lea un archivo de texto línea por línea y cree un nuevo archivo donde las líneas estén en orden inverso al que se leyeron.
5. Crear un programa que reciba tres nombres de archivos como entrada y combine sus contenidos en un nuevo archivo. Las líneas de los archivos originales deben alternarse en el archivo combinado.
6. Crear un programa que simule un bloc de notas. El programa pedirá frases al usuario y las escribirá en un archivo de texto. El archivo puede existir o no, si existe, se debe preguntar al usuario si desea sobrescribir la información o agregarla. Las líneas serán numeradas secuencialmente.
7. Crear un programa que registre las ventas de una tienda, para ello el programa debe permitir al usuario introducir los detalles de estas: nombre del producto, cantidad vendida y el precio por unidad. Esta información debe guardarse en un archivo de registro de las ventas llamado "ventas.txt" con un formato específico, por ejemplo, una línea por cada venta con los campos separados por tabulaciones.
8. Crear un programa que pida números enteros positivos al usuario hasta que introduzca el 0 que el programa terminará. El programa debe ir guardando los números introducidos por el usuario en un archivo binario y antes de finalizar leerlos del archivo y mostrarlos por pantalla.
9. Crear un programa para determinar el formato de un archivo común. Para detectar el tipo de archivo, se deben leer los primeros bytes. Tabla con los bytes de cabecera para cada formato en notación hexadecimal:

Formato de Archivo	Cabecera en Hexadecimal	Cabecera en Decimal
PNG	89 50 4E 47 0D 0A 1A 0A	137 80 78 71 13 10 26 10
JPEG	FF D8 FF	255 216 255
GIF	47 49 46 38 37 61 (GIF87a) / 47 49 46 38 39 61 (GIF89a)	71 73 70 56 55 97 / 71 73 70 56 57 97
PDF	25 50 44 46 2D	37 80 68 70 45
ZIP	50 4B 03 04	80 75 3 4
RAR	52 61 72 21 1A 07 00	82 97 114 33 26 7 0
MP3	FF FB (MPEG) / 49 44 33 (ID3)	255 251 / 73 68 51
BMP	42 4D	66 77
WAV	52 49 46 46	82 73 70 70
AVI	52 49 46 46	82 73 70 70
EXE	4D 5A	77 90
TAR	75 73 74 61 72	117 115 116 97 114
7Z	37 7A BC AF 27 1C	55 122 188 175 39 28

   Para obtener más información sobre los encabezados de los ficheros se puede consultar el siguiente enlace: https://en.wikipedia.org/wiki/List_of_file_signatures

   Conversión a Hexadecimal: La función `bytesToHex()` convierte cada byte leído en su representación hexadecimal. Usar `Integer.toHexString(bytes[i] & 0xFF)` para convertir y aplicar una máscara `& 0xFF` para tratar los bytes como valores sin signo.
10. Crear un programa que lea un archivo PNG y muestre información básica sobre el archivo, como su tipo, tamaño de la imagen, profundidad de color, tipo de compresión, filtro y método de entrelazado.

    El formato PNG (Portable Network Graphics) es un formato de imagen rasterizada que utiliza compresión sin pérdida. Los archivos PNG están organizados en una secuencia de *chunks*

(bloques) que contienen información sobre la imagen. Cada archivo PNG comienza con una cabecera específica seguida de varios chunks que contienen los datos de la imagen y su configuración.

Estructura Básica de un Archivo PNG:

- Cabecera PNG (Signature):
  - Los primeros 8 bytes son una firma constante que identifica el archivo como PNG: 89 50 4E 47 0D 0A 1A 0A (hexadecimal).
- Chunks (Bloques):
  - Después de la cabecera, el archivo PNG contiene una serie de chunks, cada uno compuesto por:
    - Longitud (4 bytes): Tamaño del bloque de datos.
    - Tipo de Chunk (4 bytes): Identificador del tipo de chunk (IHDR, PLTE, IDAT, IEND, etc.).
    - Datos del Chunk: Contenido específico del chunk.
    - CRC (4 bytes): Código de verificación del chunk.
- Chunk IHDR (Header Chunk):
  - El primer chunk después de la cabecera es siempre IHDR, que contiene información crítica sobre la imagen:
    - Ancho (4 bytes)
    - Alto (4 bytes)
    - Profundidad de Color (1 byte)
    - Tipo de Color (1 byte)
    - Método de Compresión (1 byte)
    - Método de Filtro (1 byte)
    - Método de Entrelazado (1 byte)

El programa debe mostrar la siguiente información:

- Tipo de archivo: Confirmar que es PNG.
- Dimensiones: Ancho y alto de la imagen.
- Profundidad de color: Indica la cantidad de bits por canal.
- Tipo de color: Indica si la imagen es en escala de grises, RGB, con o sin alfa, etc.
- Método de compresión: Siempre debe ser 0 para los PNG estándar.
- Método de filtro: Siempre debe ser 0.
- Método de entrelazado: Indica si la imagen está entrelazada (Adam7).

11. Crear una clase *Venta* que represente una venta con atributos como fecha (*String*), producto (*String*) y total (*double*). Escribir un programa que permita al usuario introducir datos de ventas y escriba estos datos en un archivo binario. Antes de finalizar el programa debe leer los registros de ventas guardados en el archivo y mostrarlos por pantalla. No se pueden utilizar *ObjectOutputStream/ObjectInputStream*.
12. Crear un programa que realice la copia de un archivo binario en otro. El usuario pasará el nombre del archivo origen y destino como parámetros al programa.
13. Implementar un juego de adivinanzas de números en el que el programa elija un número secreto y el usuario intente adivinarlo. Se debe utilizar la clase RandomAccessFile para guardar el historial del juego: número secreto, número de intentos del usuario y si ganó o perdió (lo adivina en 5 o menos intentos).
14. Implementar una aplicación de agenda de contactos que almacene los nombres y números de teléfono en un archivo binario utilizando la clase RandomAccessFile. La aplicación debe permitir agregar nuevos contactos, buscar contactos por nombre, actualizar números de teléfono y eliminar contactos.
15. Crear un programa que simule la gestión de archivos comprimidos en un archivo binario utilizando la clase RandomAccessFile. El programa debe permitir la compresión y descompresión de archivos individuales y la visualización del contenido comprimido (compresión simple donde se cuenten las repeticiones de bytes consecutivos en el archivo original y se almacenan en el archivo comprimido).

16. Una persona que tiene una cuenta de ahorros en un banco puede realizar ingresos y reintegros de su cuenta. Se necesita crear un programa que genere un extracto de los ingresos, reintegros y saldos obtenidos en cada transacción y se almacene en un archivo binario.
17. Crear un programa que lea el fichero XML generado en el ejercicio anterior y cree un fichero de acceso aleatorio y otro de objetos con los datos contenidos en este.
18. Crear un programa que utilice SAX para analizar el archivo XML dado y convertir su contenido en formato JSON. El programa debe mostrar el resultado obtenido en formato JSON.
19. Crear una clase llamada Persona con campos como nombre, edad y dirección. Serializar una instancia de Persona a un archivo XML y luego deserializarla para recuperar la información.

Más ejercicios opcionales

Ejercicio 1 : Muchos videojuegos necesitan calcular las operaciones matemáticas **Math.sin** y **Math.cos** sobre números entre 0 y 360 y no pueden perder tiempo durante la partida en llamar a esos métodos porque son muy lentos. Crear un programa que genere un archivo llamado **trigonometrics.bin** en el que se guarden los resultados de dichos cálculos, según este formato:

Dato a guardar	Tipo	Explicación
ángulo	int	Un número entero entre 0 y 360, que representa un ángulo medido en grados
seno	double	El resultado de aplicar el método Math.sin al ángulo pasado a radianes (usar el método Math.toRadians)
coseno	double	El resultado de aplicar el método Math.sin al ángulo pasado a radianes (usar el método Math.toRadians)
Estos tres datos se repiten para todos los números comprendidos entre 0 y 360 (ambos inclusive)		

Ejercicio 2 : Programar la siguiente clase, para ser usada en un videojuego que necesite el archivo del ejercicio anterior.

Trigonometrics
- double[] senos - double[] cosenos
+ Trigonometrics() throws IOException + double getSeno(int n) + double getCoseno(int n)

- Constructor: carga el archivo trigonometrics.bin y rellena los arrays de senos y cosenos para todos los números entre 0 y 360.
- getSeno/getCoseno: devuelve el valor del array de senos o cosenos correspondiente al resto de dividir el parámetro entre 360. Si el parámetro es negativo, IllegalArgumentException.

Ejercicio 3 : Primero programar la siguiente clase, donde se tendrá que proteger adecuadamente la nota para que esté comprendida entre 0 y 10.

Nota
- String asignatura - int nota
+ Nota (String asignatura, int nota) + getters

A continuación, hacer un programa que cree un List<Nota> vacío y muestre al usuario:

1. Añadir la nota de un examen
2. Guardar
3. Salir

- Si se pulsa 1, el programa preguntará el nombre de una asignatura y la nota que has sacado en un examen de esa asignatura. Se creará un objeto Nota y se añadirá a la lista de notas.
- Si se pulsa 2, el programa creará (sobrescribiendo si existe) un archivo llamado **notas.bin** y escribirá sobre él la lista de notas, de acuerdo a este formato:

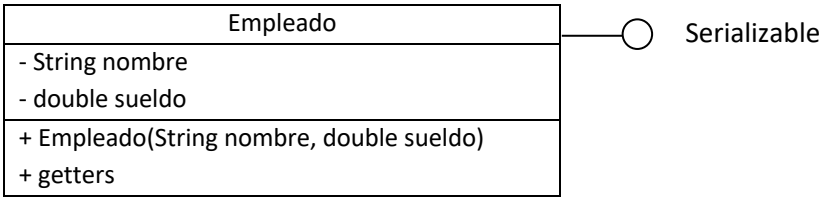
Dato a guardar	Tipo	Explicación
Total	int	El total de notas que hay en la lista de calificaciones
Asignatura	String	El nombre de una asignatura
Nota	int	La nota obtenida en el examen de la asignatura
Hay tantas parejas de datos asignatura-nota como indica el total		

Ejercicio 4 : Realizar las siguientes modificaciones sobre el programa del ejercicio anterior:

- Hacer que, al iniciarse el programa, justo después de crear el List<Nota> vacío, se mire si existe el archivo **notas.bin**. Si existe, se abre el archivo y se leen sus datos de acuerdo al formato de la tabla anterior, recuperando primero el total de notas que hay en el archivo y luego por cada pareja asignatura-nota se crea un objeto Nota y se añade al List<Nota>.
- Añadir una opción al menú llamada "Calcular media". Al pulsarla, se preguntará el nombre de una asignatura y se mostrará la nota media de dicha asignatura.

Ejercicio 5 : Repetir los ejercicios anteriores de forma que la clase Nota sea serializable y el archivo **notas.bin** guarde la lista de notas serializada.

Ejercicio 6 : Crear la siguiente clase:



A continuación, crear un programa que cree un Map<String,List<Empleado>> con los datos de la siguiente tabla y lo guarde directamente, usando serialización, en un archivo llamado **empresa.bin**

Departamento	Empleados
Informática	Antonio, 1200€ Pedro, 1800€ Ana, 2000€
Recursos humanos	Hermenegildo, 1500€ Lucía, 2000€

Ejercicio 7 : Crear un programa que cargue el archivo **empresa.bin,** pregunte el nombre de un departamento y muestre los nombres de sus empleados y lo que gana cada uno.

Ejercicio 8 : Crear un programa que pregunte por teclado al usuario la ruta de un archivo y le haga un duplicado. El duplicado se llamará **duplicado.bin** y estará en la misma carpeta que el original.